

IBM Visual Performance Analyzer

User Guide

Version 5.0

Issue Date: 06/06/2007

Revision Status: Final

About this Document

This document describes how to install and use the Visual Performance Analyzer tool. This document will help you install the tool, learn how to collect performance data on your platform and later analyze the data, using the VPA plug-ins.

Table of Contents

1. INTRODUCTION	5
1.1 VPA on Alphaworks	6
1.2 Release History	6
2 VPA BASICS	7
	,
2.1 Design Objectives.	8
2.2 Deployment	<u>8</u> 9
3. INSTALLATION	
	11
<u>3.1 Windows</u>	
<u>3.1.1 Download from Alpha works</u>	
<u>3.1.2 Extract the compressed file.</u>	
2.2 Linux	
3.2 Linux.	
<u>5.2.1 Download from Alphaworks</u>	
5.2.2 Extract the compressed file	
2.2.1 Download from Alphaworka	
2.2.2 Eutrate the commenced file	
5.5.2 Extract the compressed file	
4. COLLECTING PERFORMANCE DATA	
4.1 Using Platform Tools	
4.2 Setting up Windows to collect Profiling data	
4.2.1 Verify that your Java Runtime is installed on your system	
4.2.2 Verify that the Windows performance tools are installed	
4.2.3 Verify PI Tprof	
4.2.4 Copying data files.	
4.3 Setup up AIX to collect Profiling data.	
4.3.1 Verify that your Java Runtime is installed on your system	
4.3.2 Verify that the AIX performance tools are installed	
4.3.3 Verify AIX Tprof	
4.3.4 Copying data files	
4.3.5 Using Remote System Explorer	
4.4 Collecting Profiling Data on Linux platform	
4.4.1 Linux CELL/B.E	<u>30</u>
4.4.2 Linux PowerPC	<u>31</u>
4.4.3 Linux X86	<u>31</u>
4.5 Collecting Pipeline data on PowerPC	
4.6 Collecting Counter Data on PowerPC	
5. USING THE VPA ANALYSIS TOOLS	
5.1 Profile Analyzer	
5.1.1 Create a Profiling Configuration.	
5.1.2 Run Profiling Configuration	
5.1.3 Load an Existing Profile.	
5.1.4 Profile Navigation	
5.1.5 Profile Comparison	
5.1.6 Profile Merge	
5.1.7 Symbol Analysis	

5.1.8 Configure database connections and manage cached database files	<u></u>
5.2 Code Analyzer	104
5.2.1 Load an executable for analysis.	
5.2.2 Adding profiling information.	
5.2.3 Navigate the Executable.	<u>108</u>
5.2.4 Instruction Properties Analysis	
5.2.5 Statistic Analysis	<u>141</u>
5.3 Pipeline Analyzer	
5.3.1 Load an existing pipeline file	<u>149</u>
5.3.2 Navigating the scroll pipe view	<u>153</u>
5.3.3 Navigating the resource view	<u>156</u>
5.4 Counter Analyzer	<u>161</u>
5.4.1 Basic concepts for Counter Analyzer	<u>162</u>
5.4.2 Load an existing counter data file	<u>163</u>
5.4.3 Navigate the Counter Analyzer Perspective	<u>168</u>
5.5 Trace Analyzer	<u>194</u>
5.5.1 Basic concepts	<u>194</u>
5.5.2 Load an existing trace file	<u>195</u>
5.5.3 Navigate the Trace Analyzer Perspective	<u>196</u>
6. APPENDIX A - SAMPLE PROFILING SESSION	
7. APPENDIX B – RUN.TPROF_XML.SH,	217
8. APPENDIX C – RUN.TPROF_E.CMD.	

1.Introduction

What is Visual Performance Analyzer?

Visual Performance Analyzer (VPA) is an Eclipse-based performance visualization toolkit. It consists of five major components: Profile Analyzer, Code Analyzer, Pipeline Analyzer, Counter Analyzer and Trace Analyzer.

Profile Analyzer provides a powerful set of graphical and text-based views that allow users to narrow down performance problems to a particular process, thread, module, symbol, offset, instruction, or source line. Profile Analyzer supports time-based system profiles (Tprofs) collected from a number of IBM platforms.

Code Analyzer examines executable files and displays detailed information about functions, basic blocks, and assembly instructions. It is built on top of <u>FDPR-Pro</u> (Feedback Directed Program Restructuring) technology and allows adding of FDPR-Pro and Tprof profile information. (The Linux® version of FDPR-Pro is available here at AlphaWorks.) Code Analyzer is able to show statistics; navigate disassembled instructions; and display performance comments, instruction grouping information, and map instructions back to source code.

Pipeline Analyzer is a port of the <u>IBM Performance Simulator for Linux on POWER™</u>, another AlphaWorks technology. Pipeline joins the VPA toolkit to provide VPA users with the means of examining how code is executed on various IBM POWER processors. Pipeline Analyzer displays the pipeline execution of instruction traces generated by a POWER series processor. It does so by providing a scroll view and a resource view of the instruction execution.

Counter Analyzer is a common tool to analyze hardware performance counter data among many IBM eServer platforms, which includes systems running on AIX, i5OS, zOS, Linux on POWER, Linux on CELL/B.E.. Counter Analyzer accepts hardware performance counter data generated by AIX tools hpmc and tcount in the form of a cross-platform XML file format. The tool uses either build-in hsqldb database engine or external DB2 instance to store the raw performance counter data. The tool provides multiple views to help users identify and eliminate performance bottlenecks by examine the hardware performance counter values, computed performance metrics and also CPI breakdown models.

Trace Analyzer visualizes Cell/B.E. traces containing information such as DMA communication, locking/unlocking activities, mailbox messages, etc. Trace Analyzer shows this data organized by core, along a common timeline. Extra details are available for each kind of events, for example, lock identifier for lock operations, accessed address for DMA transfers, etc.

Visual Performance Analyzer is available as an IBM internal use tool for any IBMer who wants to try it out. Support is provided on a best-effort basis.

How does it work?

Profile Analyzer parses system profiles into an internal profiling data model that supports the profile hierarchy, offset locations, tick counts, CPU counter data, source line information, and disassembly. The plug-in then displays this data model, using various Eclipse views. The system profiles are those produced by Performance Inspector and AIX® Tprof. However, Visual Performance Analyzer can be extended to support almost any platform by converting a system profile to an XML schema that it understands.

Code Analyzer is able to read profiling information generated by AIX Tprof or FDPR-Pro performance tools. It reads in executable files and shared libraries and analyzes them using FDPR-Pro. FDPR-Pro is a post-link analyzer and performance optimization tool that can perform accurate static and dynamic analysis of executable files.

Pipeline Analyzer reads the .pipe and .config input files that are produced by the IBM Performance Simulator for Linux on POWER. An instruction trace is first collected and analyzed by a processor model. The two output files are produced for viewing with either the Performance Simulator or Visual Performance Analyzer.

Counter Analyzer reads the counter data files as its input, parses these files into an internal counter data model, and then displays this data model using various Eclipse views. The counter data files are generated by hpmc or tcount, having a suffix ".pmf".

Trace Analyzer reads in traces generated by the Performance Debugging Tool for Cell, and displays time-based graphical visualization of the program execution as well as a list of trace contents and the event details for selection.

1.1VPA on Alphaworks

Visual Performance Analyzer was released on Alpha works to explore the use of Eclipse-based performance tools with IBM customers. VPA is built as an Eclipse Rich Client Platform (RCP) package and there are versions for AIX and Windows. An RCP release contains Eclipse runtime files, all required plug-ins and VPA plug-ins.

1.2Release History

Date	Description
09/14/2006	Initial release of VPA to Alphaworks
06/08/2007	VPA 5.0 Release

2.VPA Basics

Visual Performance Analyzer is an Eclipse-based tool set that includes: Profile Analyzer, Code Analyzer, Pipeline Analyzer, Counter Analyzer, and Trace Analyzer. All of these tools are Eclipse plug-ins.



Figure 1 System Architecture of Visual Performance Analyzer

Profile Analyzer

Profile Analyzer is a system profile analysis tool. This plug-in obtains profile information from various platform specific tools, and provide analysis views for user to identify performance bottle necks.

Pipeline Analyzer

Pipeline Analyzer provides almost the same features as ScrollPipeViewer, a standalone Java application. It gets pipeline information of Power processors from the Sim-GX tool, and provides two analysis views; scroll mode and resource mode.

Code Analyzer

Code Analyzer reads XCOFF (AIX binary file format) files or ELF files running on Linux on Power, and displays program structure with block information. With related profile information, it can provide analysis views on hottest program block as well as some optimization suggestions.

Counter Analyzer

Counter Analyzer reads counter data files generated by hpmc or tcount running on AIX, and it provides multiple views to help users identify and eliminate performance bottlenecks by examine the hardware performance counter values, computed performance metrics and also CPI breakdown models.

Trace Analyzer

Trace Analyzer reads in traces generated by the Performance Debugging Tool for Cell, and displays time-based graphical visualization of the program execution as well as a list of trace contents and the event details for selection.

2.1Design Objectives

The base object of Visual Performance Analyzer is to extend the capabilities of Eclipse by adding plug-in support for: system profile, code, pipeline, and counter analysis. VPA is a collection of performance data analysis tools that can be used to identify performance bottlenecks. VPA does not supply performance data collection tools. Instead, it relies on platform specific tools, like AIX Tprof, to collect the performance data. When necessary, multiplatform support is provided by converting data into XML. The XML schema is understood by VPA and is parsed and loaded for analysis. The VPA tool must be extensible and it achieves this by allowing for additional plug-ins to be added and also by adding integration between plug-ins, e.g. shared internal data models and linked views.

Information about VPA data files:

- The .etm is the XML file for Profile Analyzer
- .etz is the zipped XML profile data
- .opm is the XML file for Profile Analyzer
- .opz is the zipped XML file for Profile Analyzer
- Java profile data from IBM JRE Java profiling tools are merged by TProf tools into a single .etm file. No additional post processing is needed.
- The pipeline files are: .pipe data file and .config file is the default configuration file.
- The .pmf is the XML file for Counter Analyzer.
- The .pe is the file for Trace Analyzer

2.2Deployment

As a performance analysis tool, Visual Performance Analyzer typically runs on User's ThinkPad or desktop as a client application. Visual Performance Analyzer can get performance-related data from servers via Remote System Explorer (RSE) or by copying the files with FTP or some other means.





2.3Software Stack Information



Figure 3 Product Stack of Visual Performance Analyzer

VPA runs on the following Operating Systems:

- (1) Windows XP with SP2 or later
- (2) IBM AIX 5.3 in latest Maintenance Level

(3) Linux/x86--Fedora Core 5/6

Profile Analyzer, Pipeline Analyzer, Trace Analyzer and Counter Analyzer are Eclipse plug-ins and are 100% JAVA code. They can run on all above supported platforms.

Code Analyzer is also an Eclipse plug-in, but it depends on FDFR-Pro libraries that are platform-dependent libraries. Code Analyzer can only run on Windows in this release.

Although VPA only runs on the above operating systems, it's important to realize that it can analyze data collected from any platform, providing the data is provided in a format understood by VPA.

VPA supports only IBM J9 JRE 5.

There is an IBM J9 JRE5 in VPA

VPA supports the following Eclipse platforms:

(1) Eclipse 3.2 with latest fixpack, such as Eclipse 3.2.2

3.Installation

No installer is required for VPA installation. The VPA installation is as simple as:

- 1. Download a newest VPA RCP release, usually it should be a zip archive or compressed tar archive
- 2. Extract the archive
- 3. Run the application by executing the Eclipse launcher script.

The RCP application will not include the following products: Performance Inspector for Windows and DB2 UDB. If you want to use these capabilities, they must install the corresponding product manually.

Configuration

No configuration is required for the VPA application installation.

Advance configuration information is provided in online-help. These configurations address some special requirements, such as setting bigger heap size of JVM for Eclipse when a user analyzes large profile.

Uninstallation

No special uninstallation action is required. If a user wants to uninstall a VPA RCP application, they can simply delete the application directory that VPA was installed to.

3.1Windows

These steps will walk you through the installation of VPA on your Windows workstation.

3.1.1Download from Alpha works

Download the latest VPA (Visual Performance Analyzer) from here:

http://www.alphaworks.ibm.com/tech/vpa

Save vpa-rcp-\${version}-win32.zip to your favorite download directory.

3.1.2Extract the compressed file.

Right Click on the file and select Extract All to open the Extraction Wizard.





As time advances, new versions of VPA will come out. In order to save yourself a lot of headaches with new versions, create the new folder with a name containing the version number and install VPA to that directory. If each version is installed this way, you'll have multiple working versions. When there is a problem, you can go back to the old version.

Select a Destination Files inside the Zi choose.	on P archive will be extracted to the locat	Select a root directory and folder to extract files to. You will need to create a folder yourself since it will
	Select a folder to extract files to	not create it automatically
	Files will be extracted to this directo	yy
	C:WPA	
		Browse
		Password
4	Extracting	
	- Back Next	> Cancel





3.1.3Create a Shortcut

A window with the folder and its contents will open if you selected "Show extracted files".



Click on the VPA executable to start VPA.



Otherwise you will see this:



If you see this screen when you start up VPA ...

S Visual Performance Analyzer	
<u>File Edit I</u> ools <u>Wi</u> ndow <u>H</u> elp	
Se Welcome ×	
Welcome to Visual Performance Analyzer 5.0	
Image: state stat	
i 📭	

It means that Eclipse is not running one of the VPA tools and you will need to switch to one of the VPA tool perspective by following this procedure.

Select tools	
Image: Control of Analyzer Im	♥welcome ♥ ■
Samp X Coun Basic Resol Symb Coun	B B S C P J [≫] 4 □ B B P C P J [≫] 4 □ Findude <al< td=""></al<>
	main(in (printl()
: •	

🗟 Profile Analyzer - Visual Performance Analyzer	
<u>F</u> ile <u>E</u> dit Profile Analyzer <u>T</u> ools <u>W</u> indow <u>H</u> elp	Welcome window
👫 Profiling Reso 🗙 Database Con Navigator 🖵 🗖	
ho 🗞 b 🕘 🗏 😋 🎾 🎽	
LOCALHOST (Local)	Moleomo to Vie
Samp 🛛 Coun Basic Resol Symb 🖵 🗖	
<u> </u> 🌒 🏹	₽ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	main(int
	printf(
i □ *	

If you have existing profiles, and do not see them after a VPA upgrade, follow these steps to restore them to the project.













N2507850.ETM N50605J7.etz S50407e1.etm S50407E1.etz Sample1.etm Sample1.etz

3.2Linux

These steps will walk you through the installation of VPA on your Linux workstation.

Supported Linux platforms are: Linux/x86: Fedora Core 5/6

3.2.1Download from Alphaworks

Download the latest VPA (Visual Performance Analyzer) from here:

http://www.alphaworks.ibm.com/tech/vpa

3.2.2Extract the compressed file

Go to the directory where gz file iscd /favdir Change file attributeschmod 755 vpa-rcp-\${version}-linux-x86.tgz Decompress the filetar –xvfz vpa-rcp-\${version}-linux-x86.tgz

3.3AIX

These steps will walk you through the installation of VPA on your AIX workstation.

3.3.1Download from Alphaworks

Download the latest VPA (Visual Performance Analyzer) from here:

http://www.alphaworks.ibm.com/tech/vpa

Save vpa-rcp-\${version}-aix-ppc.zip to your favorite download directory.

3.3.2Extract the compressed file

Go to your favorite download directory and follow these steps to extract the VPA tool:

Go to the directory where gz file is	cd /favdir
Change file attributes	chmod 755 vpa-rcp-\${version}-aix-ppc.tgz
Decompress the file	gzip -dc vpa-rcp-\${version}-aix-ppc.tgz tar xvf -

4.Collecting Performance Data

VPA is a collection of performance data analysis tools. It relies on platforms to provide the necessary tools for collecting data and converting the data into a format that is understood by VPA.

4.1Using Platform Tools

Visual Performance Analyzer works with following tools for collecting profile data.

- AIX Tprof
- Performance Inspector for Windows Tprof
- IBM JRE Java profiling tools
- Linux oProfile

Profile data from AIX tprof is converted into XML by using the –X flag. The .etm is the XML file for Profile Analyzer; .etz is the zipped XML profile data. Profile data from PI Tprof is in a .out format, which profile analyzer supports directly. Java profile data form IBM JRE Java profiling tools are merged to above tools.

Pipeline data is generated from tools found in the <u>IBM Performance Simulator for Linux on POWER™</u> project on Alphaworks. The .pipe file is pipeline data file and .config file is the default configuration file.



4.2Setting up Windows to collect Profiling data

4.2.1Verify that your Java Runtime is installed on your system

Run the following command:

java -version

You should see something similar to the following:

java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cn142-20050609 (JIT enabled: jitc))

Note: You need version 1.4.1 or higher

4.2.2Verify that the Windows performance tools are installed

VPA runs with the Performance Inspector for Windows performance tools. Run the following command: Swtrace -?

You should see something similar to the following:

D:\>swtrace -? SWTRACE Version: 7.1.1 Valid SWTRACE commands: ...

The Performance Inspector for Windows package can be downloaded from here: http://www.alphaworks.ibm.com/tech/pi

4.2.3Verify PI Tprof

Using the PI tools themselves, you can verify their operation by capturing a system trace using these steps:

Swtrace init

Swtrace enable Tprof

Swtrace on

Swtrace off

Swtrace get

Swtrace post

Post

At this point you should have a PI profile (.out file) in your working directory that you can look at. Refer to PI documentation for details on PI tools.

You can capture traces yourself or you can configure VPA to collect traces. Refer to the Profile Analyzer plug-in section in this document.

4.2.4Copying data files

Running Performance Inspector for Windows Tprof, produces an ascii profile (.out) file. You can simply use FTP to transfer the file to your system running VPA or open the profile locally if you have VPA installed on the same system. See section 4.4 about using Remote System Explorer.

4.3Setup up AIX to collect Profiling data

4.3.1Verify that your Java Runtime is installed on your system

Run the following command:

java -version

You should see something similar to the following:

java version "1.4.1" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1) Classic VM (build 1.4.1, J2RE 1.4.1 IBM build cxppc321411-20040301 (JIT enabled: jitc))

Note: You need version 1.4.1 or higher

4.3.2Verify that the AIX performance tools are installed

Recent versions of AIX Tprof can generate XML profiles. AIX 5.3.TL5 or higher is required. The utility that produces a VPA profile from the Tprof output is bundled with the bos.perf.tools package. It includes an updated versions of Tprof, Symlib and the added tprof2xml utility.

Verify installation of bos.perf.tools package:

lslpp -L bos.perf.tools | grep "bos.perf"

If not installed, you can use smitty or installp

4.3.3Verify AIX Tprof

Using the AIX tools, you can verify their operation by capturing a system trace using these steps:

tprof -eukj -X -A -F -r vpa_test -x sleep 5

At this point you should have a Tprof profile (vpa_test.etm file) in your working directory that you can look at.

You can capture traces yourself or you can configure VPA to collect traces. Refer to the Profile Analyzer plug-in section in this document.

4.3.4Copying data files

All versions of AIX support FTP. So, once AIX Tprof has produced the XML profile (.etm) file, you can simply use FTP to transfer the file to your system running VPA. If VPA has been installed on the same AIX system you can open the profile locally. See section 4.3.5 about using Remote System Explorer.

4.3.5Using Remote System Explorer

You can configure VPA to use Remote System Explorer (RSE) to remotely collect data and transfer files. However, VPA does not distribute an RSE server component. The below steps illustrate how to configure an AIX remote resource but the steps are similar to configuring a Windows remote resource as well.

Open Remote Connection View	Choose Window \rightarrow Show view \rightarrow Other \rightarrow Remote Systems \rightarrow Remote Systems
Open Remote AIX System Connection wizard	Under Remote Systems window, double click New Connection \rightarrow AIX



Follow wizard to specify information about the remote System

Sew New			
Remote AIX Sy: Define connection i	stem Connection	_	
Parent profile:	IBM-92373328618	•	
Connection name:	Select if RSE daemon was sta manually on AIX Server	rted	
Host name:	9.3.104.80 Select if you want RSE daemo only when a connection is	n to start made	
Description:			
☑ <u>V</u> erify host nan	e		
	Click Next Click to connecti Finish	·	
	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel	
😪 New			
Files Define subsystem inform	ation	=	
Server Launcher Settings Indicate how the remote server should be launched • Remote daemon Daemon Port 4035 • REXEC Path to installed server on host /opt/rseserver/ Server launch command perl./server.linux Port 512			
C Connect to runnin) server		
DLM	Alphaworks		

See New	X
Files Define subsystem information	
Server Launcher Settings Indicate how the remote server should be launched Remote daemon Daemon Port 4035 REXEC Path to installed server on host /opt/rseserver/ Server launch command perl ./server.linux Pgrt 512 Connect to running server	
< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel	

Note:

If RSE daemon was started manually on the AIX server choose Remote daemon option. The port 4035 is selected by default.

If you want RSE daemon to be started automatically when a connection is made, select REXEC option and specify where the server launch command is found. This is typically a perl script.

Connect to remote server

Right click the new connection and select Connect

	🕂 Move Down	×
🔏 Remote Syste	Connect	8
🖃 🚅 New Conr	Disconnect	
🗄 🏘 Windd	Clear Passwords	
📃 🕀 🔬 Linux.		-
🕀 🔬 Power	Run As	
	Debug As	
H	Profile As	
	Properties(H)	-
⊞ -G _x 9.3.104.8	0	-
■ 〒…尾… 9.3.104.3	Right Click and select	

Type username / password

🗟 Enter Password 🛛 🛛		
System type:	AIX	
Host name:	9.3.104.80	
<u>U</u> ser ID:	root	
<u>P</u> assword:	****	
	🔲 Save user ID	
	Save password	
Select OK		
	<u> </u>	Cancel

Go to Profiling Resources, click refresh to see the new connection



4.4Collecting Profiling Data on Linux platform

4.4.1Linux CELL/B.E.

- > Hardware: CELL/B.E. blade
- Software Requirement: fedora core 6 and CELL/B.E. SDK 2.0 installation
- Verify oprofile: opcontrol/opreport –X
- Tool Usage:

After verifying that oprofile has been installed successfully, users should first use "opcontrol --init" to initialize oprofile module; and then, use "opcontrol –event=event:count" to add an event to measure for the hardware performance counters (users can refer to event names and minimal counters by using "opcontrol –l").

Next, use "opcontrol --separate=all" to separate samples based on the given separator. It is not an optional step, user must process it to meet VPA requirement.

Users can use "opcontrol --start" to start collecting profiling data, and start one user application: then, use "opcontrol --stop" to stop collecting profiling data; and use "opcontrol --dump" to force a flush of the collected profiling data to the daemon; Finally, use "opreport -X -g - I - d - o xxx.opm" to generate a specified XML output, which can be imported to Profile Analyzer. The xml output file must be suffixed with the extension '.opm', which identifies an acceptable file.

Users can further use "opcontrol --reset" to clear out data, and choose "opcontrol --deinit" to unload the oprofile module.

Some important command usages :

opcontrol --init :

loads the oprofile module and oprofilefs

opcontrol -event=event_name:count:unit_mask:kernel-space_count:user-space_count :

choose an event with specified event_name, count, unit_mask, kernel-space counting, user-space counting. Here, the unit_mask, kernel-space counting, user-space counting are optional.

A default event can be specified with the command " *opcontrol –event="default"* ". Generally, the default event is the system timer of the OS and hardware.

opcontrol -I :

list event types and unit masks

opcontrol --start/--stop/--reset/--deinit :

start running the oprofile, stop oprofile, reset the profile data in default session. Unload oprofile module.

opreport -X –g –d –l xxx.opm :

Generate a specified XML output

(Here: -X : specifies the output file in XML format.

-g : show source file and line for each symbol.

-I : list per-symbol information instead of a binary image summary.

-d : show per-instruction details for all selected symbols.)

4.4.2Linux PowerPC

- > Hardware: System p servers or POWER blade
- Software Requirement: Linux, oprofile (oprofile Download Link: <u>http://oprofile.sourceforge.net</u>)

Verify oprofile: the same as CELL/B.E.

> Tool Usage: the same as CELL/B.E.

4.4.3Linux X86

- > Hardware: X86 based machine
- Software Requirement: Linux, oprofile (oprofile Download Link: <u>http://oprofile.sourceforge.net</u>)
- Verify oprofile: the same as CELL/B.E.
- > Tool Usage: the same as CELL/B.E.

4.5Collecting Pipeline data on PowerPC

Pipeline Analyzer is a port of the <u>IBM Performance Simulator for Linux on POWER™</u>, another alphaWorks technology. Please refer to the directions given by this project for collecting pipeline data. While VPA provides the Pipeline data analysis tool, the project provides the tools necessary for collecting and generating Pipeline data files.

4.6Collecting Counter Data on PowerPC

Before using Counter Analyzer to view and operate counter data, you should first prepare data from data source.

- 1. Our Counter Analyzer supports opening the counter data file generated by both **hpmc** and **tcount**. The following are two instances:
 - hpmc -s 0.1 -mv -G 1,2,3 -m output.pmf -x sleep 5
 - o tcount -g 4,5,6,7,8 -X output.pmf sleep 5
- 2. Make sure that the counter data file has the suffix ".pmf".

5. Using the VPA analysis tools

This section describes the use of each plug-in. structure of the system by first focusing on some typical usage scenarios where various tasks are performed, then outlining the major components of the system and their interactions. You can find this information by selecting Help - Help Contents within VPA. To get context sensitive help, press F1 for Windows and AIX or press Ctrl+F1 for Linux.

5.1Profile Analyzer

Profile Analyzer is a tool that allows you to navigate through a system profile, looking for performance bottlenecks. It provides a powerful set of graphical and text-based views to allow users to narrow down performance problems to a particular process, thread, module, symbol, offset, instruction or source line. It supports profiles generated by Performance Inspector (tprof) and AIX tprof. It also merges IBM JRE Java profile data when it is merged into the above profiles. To load huge profile data files and reduce memory footprint, Profile Analyzer now uses database to cache profile files. The current version supports DB2 and an embedded database.

You can also find the Profile Analyzer User Guide from within VPA. Select **Help - Help Contents** within VPA. To get context sensitive help, press **F1** for Windows and AIX or press **Ctrl+F1** for Linux.

5.1.1Create a Profiling Configuration

You can configure a system profile and have VPA run a workload and collect the profile. The steps are mostly the same for all supported systems. So, the example below will provide the steps necessary for creating a profiling configuration for a Windows system but are much the same as for any other system.

In the Profiling Resources view, right-click over a connection and choose New Profiling Configuration:

On the first page of the Profiling Configuration wizard, enter a name for the configuration (to remind you of what purpose the configuration serves).Currently, there are two kinds of tprof tools: Performance Inspector tprof and AIX tprof. Since you want to profile on Windows, you should choose Performance Inspector tprof and specify the profiling tools location. Then select the CPU type from the dropdown list. Click **Next**.

🚳 Profiling Configuration Wizard 🛛 🔀
Configure target profiling system You need to specify the location of the profiling tools on the target system.
Configuration name:
New Configuration
System:
LOCALHOST
Profiling tool type:
Performance Inspector tprof
Profiling tools location:
C:\ibmperf\bin
CPU:
x86 - Pentium4
< Back Next > Finish Cancel

On the second page, choose an application to launch, and enter its command line options and working directory. If the application is a Java application on IBM Virtual Machine for Java, select the Enable Java profiling checkbox, which will define the IBM_JAVA_OPTIONS environment variable for the Java process being started, so that JIT-compiled Java methods are profiled. Note: If you want to profile a system without launching an application, for instance because the application is already running, leave these fields blank. Click Next when this page is complete.

🖨 Profiling Configuration Vizard 🛛 🔀
Configure target profiling system If you want to launch application along with the profiling, please fill in the fields on this dialog form.
Select application to launch (leave empty for system wide profiling):
"C:\Program Files\IBM\Java142\bin\java.exe"
Application command line options (if any):
C:\profiles\Hello.class
Working directory for the launched application:
C:\profiles
✓ Enable Java profiling
< Back Next > Finish Cancel

On the third page you can define the way to profile, when and how to start and end the profiling.

Choose whether to profile manually or with the application. (Only manual profiling is available if you did not select an application to launch.) Fully automatic profiling is the simplest: it involves a single click of the Run icon, which will launch the profiler and your application, run the application to completion, stop the profiler, and load the profile into Profile Analyzer. Fully manual profiling requires you to start the profiler, start and stop the application (if one was entered on the second page), and stop the profiler.

For fully automatic profiling, choose **With application** and leave the entry fields to their default values of 0. If you want to run your application automatically but give it a predetermined time to "warm up" before profiling begins, choose **With application** and enter the warm-up time in the Delay profiling start by (seconds) entry field.

For fully manual profiling, choose 'Manual'. You can define the time for profiling in Profile for entry field.

🖶 Profiling Configuration Vizard 🛛 🔀
Configure the profiling options Note: You can always click Stop profiling before the scheduled end of a profile run.
<pre>Start profiler options: Manual Profile for (seconds, 0 means indefinite): With application Delay profiling start by (seconds): End profiler options: Profile for (seconds, 0 means indefinite): V Terminate profiling when application exits Kill application when profiling ends Number of successive run(s): </pre>
< Back Mext Finish Cancel

On the final page of the wizard, you can choose a supported CPU event to profile, or leave the default value of System timer. You can also define number of cycles that the processor is not halted or in sleep. Once you have chosen one of these, select **Finish** to create the profiling configuration.
🖨 Profiling Configuration Vizard
Configure the profiling options Choose either the timer or event based sampling option based on the CPV selected on page 1.
C System timer • CPU event: Wumber of guales that the processor is not holted our is also we
▼ Number of cycles that the processor is not halted nor in sleep 1000000
<u>Eack</u> <u>Next</u> <u>F</u> inish Cancel

The new profiling configuration should be visible when the host it was created for is expanded:



You can define as many configurations for local systems as you require. If you want to create two configurations that are largely similar but contain slightly different settings (for example, which CPU counter is used or a change in command line arguments), you can make a copy of a configuration as follows:

- 1. Right-click over the configuration and choose **Make Configuration Copy**. A new configuration is created with the name Copy of original configuration name.
- 2. Right-click over the copy and choose **Modify Profiling Configuration**.... This starts the Profiling Configuration Wizard. From here you can change any settings in the copied configuration.

5.1.2Run Profiling Configuration

To run a profiling configuration, select the configuration in the Profiling Resources view. You can then use the popup menu or the toolbar buttons to start or stop the profiler or the application. Some choices are greyed out from the toolbar, or not shown on the pop-up menu, depending on whether you chose manual or automatic application start.

To start the profiler manually, right-click and choose Start Profiler, or click the Start Profiler button from the

toolbar: M. To start the application manually, or to start both the profiler and the application if you have set the application up to start automatically, right-click and choose Launch Application, or click the Launch Application

button from the toolbar: If you have set up automatic profiling, the profile will run according to the configuration settings, and when it ends the profile will be loaded into Profile Analyzer. There is a delay between the end of profiling and the load into Profile Analyzer, which may vary from under a second to several minutes depending on:

- The amount of profile data (a very long-running profile will have a large trace buffer, which may take a long time to post-process by TPROF before Profile Analyzer can load it)
- For remote profiling, the size of the generated profile and the connection speed.

While the profiling takes place, the Profiling Resources view is greyed out to prevent you from starting multiple profiles at the same time. (This is because the toolbar buttons can only apply to a single running profile.)

You can stop profiling, for a manual configuration or for an automatic configuration where you want to override the

automatic settings, by clicking the **Stop Profiler** toolbar button: **•**.You can stop the profiled application at any

time by clicking the **Terminate Application** toolbar button: **•**. While your application runs, its output is displayed in the Console window (if it produces any output to stdout or stderr). If this window is not visible you can display it by selecting the lower right pane and choosing **Windows -> Show View -> Other -> Profile Analyzer -> Console**.

5.1.3Load an Existing Profile

When you first start Visual Performance Analyzer, press the Tools button and then select Profile Analyzer to load the plug in.

🥱 Profile Analyzer - Visual Performance Analyzer	
File Edit Segrch Frofile Analyzer Iools Yindow Help	
🕂 Profil Vurces X Database Connections Navigator 🗖 🗖	
LOCALHOST	
Click on Tools, Select Profile Analyzer	
Samples Di 23 Counters Basic Blocks Resolved C Symbol Dis	
<u> </u>	

You can also load Profile Analyzer perspective by choosing **Window - Open Perspective - Other - Profile Analyzer**.

If you already have profiles generated by TPROF or a Profile Analyzer compatible XML-based profile generator, you can open them by following the steps below.

- 1. Your Eclipse window should look like the one above. Select Profiling Resources view. You can open this view by selecting **Window -> Show View -> Profile Analyzer-> Profiling Resources**.
- 2. Profile Analyzer profiles must have one of the following extensions:
 - \circ An extension of .out , .etn or an extension of .etm
 - .opm and .opz

SVisual Performance Analyzer				
<u>F</u> ile <u>E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> elp				
] 📰 📊 堅 🍫 🔗				
Profiling Resources 🛛 🎽 🗖 🗖	💾 D:/code_analyze:	r/sort/sort.etm 🗙	E	
🐜 🗣 🕨 🖨 🗳 🎭 🏹	🖃 🚹 D:/code anal	yzer/sort/sort.etm	% Process	<u>^</u>
	🕂 🕂 🗜 Process 🔾	> Thread > Module	50.05 java [PID 14802	2]
S DCALADSI (Local)	🗄 📙 Process 🕽	> Module	25.68 /sort [FID 165 23.55 /sort [FID 165	20 505
	🕂 👫 Modules		0.14 /sort [PID 165	528
			0.09 ./sort [PID 165	528
			0.04 java [PID 14802	2]
			0.03 syncd [PID 8026	5]
			0.03 java [FID 14802	
	<	>	Process > Thread > Modu	le (f
🌖 Samples Distribut 🛛 🎽 🗖 🗖	🔚 Disassembly/Off	sets 🔀 Source Code	» ₅	
📙 🌒 🗸 🔤	Offsets for: .xxxsh	low (./sort)		
	🛎 🎭 🌩 🕇			
./sort TID 40639	Offset	Bytes	Disassembly	^ -
	0x00000000	7c0802a6	mfspr r0,LR	-
	0x00000004	93elfffc	stw r31,-4(r1)	
	0x0000008	90010008	stw r0,+8(r1)	
100.00%	0x000000c	9421ffb0	stwu r1,-80(r1)	
100.00%	0x00000010	83e20044	lwz r31,+68(r2)	
symbol: .xxxshow = Total ticks: 2963, t	0x00000014	90610068	stw r3,+104(r1)	v
symbol: All Symbol under .xxxshow	X		>	

You can load any of these profiles by click $rac{1}{2}$ and the profile opens as follows:

In VPA a profile data file loading process is able to run as a background runnable job. When VPA is loading a file, you can click a button to put the loading job to run in the background. While the loading job is running in the background, you can use Profile Analyzer to view already loaded profile data files, or event start another loading job at the same time.

As already stated, profile data files are loaded into database tables and kept in database tables until the user deletes them. Once a profile data file is successfully loaded into a database, further attempts to load the same data file will result in the data being reloaded directly from the database tables. Profile Analyzer does not need to read and parse the original file again. This allows for much faster loading of profile data into VPA after the initial database caching.

Note: although further use of a profile data file results in loading from the database, the original file is still required for Profile Analyzer to work properly. This is because not all of the content of the original file is loaded into database tables. For example, time data is kept in original file and we only store the offset and length information in database tables. When needed, this data is read form they original file on-demand.

5.1.4Profile Navigation

The following are tasks that you can perform to navigate around profiles within Profile Analyzer.

5.1.4.1Navigate process hierarchy

The **Process hierarchy view** appears by default in the top center pane. It shows an expandable list of all processes within the current profile. You can expand a process to view its module, later thread and etc. You can also view the profile in the form of thread or module and etc. Actually, you can define the hierarchy view by right-click profile and choose **Hierarchy Management**. Thread data is not available in merged profiles (.etm extension).

For more information about Hierarchy Management, you can refer to Navigate Generic Hierarchy Model

The following screen capture shows a process hierarchy in its unexpanded state:

🖨 Visual Performance Navigator - line-numbers-Hell	o_java.out - E	clipse SDK		
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject Run Tests <u>R</u> un <u>W</u> indow <u>H</u> elp	e e e e e e e e e e e e e e e e e e e			
] 📸 🕶 🔄 💁 🛛 🔗 🗸 🖓] 🆘 🗇 🗸 🔿 🗸			E2	🚹 Visual Per 🂙
Navigator 📲 Profiling Resources 🛛 📃 🗖	💾 line-numbers-H	ello_java.out 🛛		
	line-numbers-Hello	_java.out		
ELCALHOST (Local)		<pre>exe_dlcC541 ticks,</pre>	(50.6%) Ticks % Symbol 81 ticks/3 331 98.2 Hello.foo(I)V 81 ticks/27 0.59 java/io/Buffer 10.29 sum/io/CharToB 10.29 sum/io/FileOu 10.29 java/io/FileOu 10.29 java/io/FileOu 38 ticks/0 10.29 java/io/FileOu 9 ticks/0 java/io/Buffer 18 ticks/0 java/io/FileOu 9 ticks/0 java/io/Buffer 18 ticks/0 java/io/Buffer 18 ticks/0 java/io/Buffer 18 ticks/0 java/io/Buffer 18 ticks/0 java/io/Buffer 19 ticks/0 java/io/Buffer 10.29 java/io/Buffer java/io/Buffer 10.20 java/io/Buffer java/io/Buffer 10.20 java/io/Buffer java/io/Buffer <th>edWriter.write(Ljaway yteConverter.convert/ tream.print(C)V trustream.write((BEI) edWriter.flushBuffer</th>	edWriter.write(Ljaway yteConverter.convert/ tream.print(C)V trustream.write((BEI) edWriter.flushBuffer
Resolved Calls 🛛 Counters Basic Blocks 🎽 🗖	Disassembly/Of	fsets 🗙 Call-Gr	aph Caller Console Profile Comparison	»₁ □
Method: Hello.foo(I)V	Offsets for: Hello	o.foo(I)V(Process 33	356) 🖺	🎭 📥 🍋 🏹
Module: JITCODE	Offset	Bytes	Disassembly	÷ T 🔨 🗖
rocess: java.exe_dlc	0x00000000	83EC08	SUB ESP,8H	
Known callers (calls to here; ticks in caller)	0x0000003	3B6518	CMP ESP, DWORD PTR [EBP+18H]	6.64
Simbal	🗦 0x0000006	0F8694000000	JBE OAOH	
Symbol Call.	0x000000C	53	PUSH BBX	
< > >	0x000000D	8B7C2410	MOV EDI, DWORD PTR [ESP+10H]	0.60
	0x0000011	B8D34D6210	MOV EAX,10624DD3H	6.94
Known descendants (calls to descendant from all sources; ticks is	0x00000016	F7EF	IMUL EDI	
Symbol Call:	0x00000018	C1FA06	SAR EDX,6	14.8
	0x0000001B	SBDA	MOV EBX, EDX	9.06
	0x000001D	CIEBIE	SHR BBX,31	8.45
	0x00000020	69D2¥8030000	TMUL EDX EDX 3E8H	9.97
	0x00000022	2BFA	SUB EDI.EDX	24.7
	<			>
	P			

As in most Profile Analyzer views, objects are sorted from most to fewest ticks. In this view you can see that the **IdleProcess** was the process with the most ticks, indicating either I/O delays or actual idle time during the process (for example, if the application being profiled ran on one CPU and the system had a second, mostly idle CPU).

You can expand a process to view the threads or modules beneath it. As you select a process, thread, or module, the Symbols view updates to display the list of symbols that belong to that process, thread, or module. The Samples Distribution Chart also changes, as you select different processes or threads, to display the proportion of ticks used by the most important modules within the selected process or thread.

The following two views show part of the above process in **Process>Thread>Module** hierarchy:

🚏 D:/test files/line-numbers-Hello_java.out 🗙	💾 D:/test files/line-numbers-Hello_java.out 🗙
🖃 💦 line-numbers-Hello_java.out (5020.0 ticks) [default counter: Ticks]	🖃 🛼 line-numbers-Hello_java.out (5020.0 ticks) [defaul 🔨
🖶 🗜 Process > Thread > Module	🖶 🗜 Process > Thread > Module
🕀 💽 java.exe [PID 3356] (2541 ticks/50.62%)	🖻 💽 java. exe [PID 3356] (2541 ticks/50.62%)
🕀 💽 javaw.exe [PID 2908] (1407 ticks/28.03%)	🖃 🖗 tid_main_Oc94 [TID 3220] (2355 ticks/92.68%)
🕀 💽 IdleProcess [PID 0] (605 ticks/12.05%)	
🕀 💽 vsmon.exe [PID 1104] (55 ticks/1.10%)	
🕀 💽 NotesBuddy. exe [PID 1956] (54 ticks/1.08%)	
🕀 💽 agent.exe [PID 3096] (48 ticks/0.96%)	
🕀 💽 System [PID 8] (45 ticks/0.90%)	
🕀 💽 csrss.exe [PID 180] (33 ticks/0.66%)	
🕀 💽 CMD. EXE [PID 2876] (31 ticks/0.62%)	
🕀 💽 emacs.exe [PID 3076] (22 ticks/0.44%)	
🕀 💽 emacs.exe [PID 2748] (19 ticks/0.38%)	
🕀 💽 emacs.exe [PID 2404] (18 ticks/0.36%)	
🕀 💽 iclient. exe [PID 1916] (13 ticks/0.26%)	
🕀 💽 javaw.exe [PID 2724] (11 ticks/0.22%)	
🕀 💽 NILNOTES. EXE [PID 1300] (10 ticks/0.20%)	
🕀 💽 xterm.exe [PID 824] (10 ticks/0.20%)	
🕀 💽 rxvt.exe [PID 892] (9 ticks/0.18%)	
🕀 💽 rxvt.exe [PID 2036] (7 ticks/0.14%)	
🕀 💽 rxvt.exe [PID 2244] (7 ticks/0.14%)	
🕀 💽 Rtvscan. exe [PID 1044] (6 ticks/0. 12%)	
24 left	
🗄 🗜 Process > Module	
±	🗄 🧪 tid_Ob40 [TID 2880] (142 ticks/5.59%) 🛛 💌

Navigate generic hierarchy model

A process may have some threads, and each thread can visit some modules (for instance, DLLs) and call procedures/methods (symbols) in these modules. In default condition, you can observe systems in the hierarchy of **Process > Core > Thread > Module**, **Process > Thread > Module**, **Process -> Module** and **Module**. With the function of generic hierarchy model, you can create your own hierarchy view. For example, if you want to group threads which use a common module, you can display the hierarchy Process > Module > Thread by creating it in the **Hierarchy Management**.

Attach hierarchy file to profile file, please do the following steps:

- 1. Right-click in the process hierarchy view and choose Hierarchy Management
- 2. Click the ... button to open a new hierarchy file for attaching to the profile file
- 3. select the check-box to attach the new hierarchy file to the profile file

Here are some pictures to show how to attach a new hierarchy file -"testing.xml":

🚭 Hierarchy Management			
Hierarchy Management			
You can change the display order o	f Hierarchies,		
	Open a n	ew hierarchy fil	e
Viennels Riller		¥	
hierarchy file: C:\wen.xml [Defa	ult]	. <mark>,</mark> 🗠	Save as
Attach to D:/test files/line-nu	mbers-Hello_java	a. out	Set to Default
-Hierarchy Definition:			
Hierarchy Name: Process > Thread	i > Module		
Current Hierarchies:			
Process > Thread > Module			Move <u>Up</u>
Process > Module			Move Down
Handule Module			New
			Lopy
			<u>E</u> dit
			Delete
			Reset
	ОК	Cancel	Apply
😼 Hierarchy Ianagement			×
Hierarchy Management			
You can change the display order of	f Hierarchies,		
Winnersher Willer			
D:\test files\te	sting.xml		Save as
Attach to D:/test files/line-nu	umbers-Hello_java	. out	Set to Default
Hierarchy Definition:			
Hierarchy Name: Process > Thread	l > Module		
Current Hierarchies:			
Process > Thread > Module			Move Up
Module Module			Move <u>D</u> own
			New
			Copy
			Edit
			neTere
			Reset
			Reset
	07		Reset

🚭 Hierarchy Tanagement	×
Hierarchy Management	
You can change the display order of Hierarchies,	
Hierarchy File: D:\test files\testing.xml	Save as
Attach to D:/test files/line-numbers-Hello_java.out	Set to Default
So Tarning	
Are you sure to attach the selecte file to the current profi	le?
<u>Tes</u>	<u>No</u>
	Conv
	Edi+
	Balata
	DeTece
	Keset
OK Cancel	Apply



To create you own hierarchy view, follow these steps:

- 1. Right-click in the process hierarchy view and choose Hierarchy Management
- 2. In the Hierarchy Management Wizard, click New
- 3. Give your hierarchy a specific name if you like, or the system will generate a name for you.
- 4. Select the element you want to have in your view. You may reorder your hierarchy by choosing up or down
- 5. Click **Apply** or **Ok**

Here is a new hierarchy view we create to see the threads under each module:

😼 Hierarchy Management		
Hierarchy Management You can change the display order o	f Hierarchies,	
Hierarchy File: [Plugin]/config/ Attach to D:/test files/line-nu Hierarchy Definition:	hierarchies.xml [Default] ▼ mbers-Hello_java.out	Save as Set to Default
Hierarchy Name: Process > Thread	l > Module	
Current Aierarchies:	R	Move <u>Up</u> Move <u>Down</u> <u>N</u> ew <u>C</u> opy <u>E</u> dit De <u>l</u> ete Reset
	OK Cancel	Apply

Click the **New** button to create a new hierarchy view:

🔏 Customize Hierarch	ıy
New Hierarchy You can create a nem b Ty Hierarchy Name:	/pe the name of the hierarchy
Available levels:	Current levels:
Process ≁ Thread System Attribute Module € Core	Move Up Move Down Add or remove the available levels Add-> <-Remove
Hide when missing l	evels
	OK Cancel Apply

When you click Apply or OK, you can see the change in Process Hierarchy View

💾 17235867. out 🗙 💾 line-numbers-Hello_java. o	at						
17235867. out							
🖃 🐂 Profile 17235867.out (2186 ticks)	Ticks	%	Process	Module	Thread	Symbol	^
🗐 🕂 🔓 Process > Module > Thread	2029	92.8	IdleProcess_0	C:\\intelppm.sys	tid_0000	NoSymbols	
🗄 💽 IdleProcess_0(2035 ticks/93.0%)	47	2.15	cmd. exe_1304	C:\\VETEFILE.SYS	tid_199c	NoSymbols	
🛨 🚺 cmd. exe 1304(83 ticks/3.79%)	15	0.68	cmd. exe_1304	C:\\ntoskrnl.exe	tid_199c	NoSymbols	
+ javaw. exe 1444 (33 ticks/1.51%)	11	0.50	cmd. exe_1304	C:\\hal. dll	tid_199c	NoSymbols	=
t csrss_exe_2c8(8_ticks/0_36%)	4	0.18	javaw.exe_1444	C:\\ntoskrnl.exe	tid_1cc4	NoSymbols	
$\mathbf{E} = \mathbf{E} \mathbf{E} \mathbf{E} \mathbf{E} \mathbf{E} \mathbf{E} \mathbf{E} \mathbf{E}$	4	0.18	cmd. exe_1304	C:\\ntdl1.dl1	tid_199c	NoSymbols	
Explorer. EXE_110(0 (TCRS)0.21%)	4	0.18	ask. exe_e64	C:\\VETEFILE.SYS	tid_12f8	NoSymbols	
emule.exe_134(5 ticks/0.22%)	4	0.18	IdleProcess_0	C: \ \hal. dll	tid_0000	NoSymbols	
+ ask. exe_e64(4 ticks/0.18%)	3	0.13	javaw.exe_1444	C:\\hal.dll	tid_1cc4	NoSymbols	
⊕ 💽 cmd. exe_1cdc(4 ticks/0.18%)	3	0.13	javaw.exe_1444	C: \ \win32k. sys	tid_1cc4	NoSymbols	
🗄 [javaw. exe_1a04 (2 ticks/0. 09%)	3	0.13	javaw.exe_1444	C: \ \jvm. dll	tid_lcc4	**MMI**	
主 💽 ctfmon. exe_e4(1 ticks/0.04%)	3	0.13	eMule.exe_154	C:\\ntoskrnl.exe	tid_0158	NoSymbols	
🕂 🦲 flashget, exe de0(1 ticks/0.04%)	3	0.13	cmd.exe_lcdc	C: \ \VETEFILE. SYS	tid_1818	NoSymbols	
+	3	0.13	cmd.exe_13U4	C: A AKERNEL32. dll	tid_199c	NoSymbols	
\mathbb{H} Regite3 are 1dbg(1 tight)(0.04%)	2	0.09	javaw.exe_1444	U: A Ajite. dll	tid_lcc4	dataflow_	
$\frac{1}{2} = \frac{1}{2} \left[\frac{1}{2} \left[$	2	0.09	javaw.exe_1444	U: \ \jite. dll	tid_lcc4	qsort	
± System_4(I ticks/0.04%)	2	0.09	cmd. exe_1304	U: \ \UnsMinKP. sys	tid_199c	NoSymbols	
	2	0.09	Idlefrocess_U	U: \ \tepip. sys	tid_0000	NoSymbols	
	2	0.09	Explorer.EXE_710	U: \ \hal. dll	tid_0724	NoSymbols	
	1	0.04	msnmsgr.exe_798	U: \ \ntoskrn1. exe	tid_1890	NoSymbols	Y
	<	пи	LOTTON OT C LOUA	1113 Aptorioral ore	+14 1458	NoYrmbolr	_
		\ .				-	
· · · · · ·	frocess	> Mod	ule > Thread				

You may add other hierarchy views in the Process Hierarchy View as you like.

💾 17235867. out 🗙 💾 line-numbers-Hello_java. out					- 8
17235867. out					
🖃 🐂 Profile 17235867. out (2186 ticks)	Ticks	%	Thread	Module	Symbol 🔨
🕀 🏪 Process > Module > Thread	2029	92.8	tid_0000	C:\\intelppm.sys	NoSymbols
\doteq $\mathbb{H}^{\mathbb{C}}$ Thread > Module	47	2.15	tid_199c	C:\\VETEFILE.SYS	NoSymbols
🗄 💾 test3 (2186 ticks/100.0%)	15	0.68	tid_199c	C:\\ntoskrnl.exe	NoSymbols
	11	0.50	tid_199c	C:\\hal.dll	NoSymbols 📃
	4	0.18	tid_1cc4	C:\\ntoskrnl.exe	NoSymbols
	4	0.18	tid_199c	C:\\ntdl1.dl1	NoSymbols
	4	0.18	tid_12f8	C:\\VETEFILE.SYS	NoSymbols
	4	0.18	tid_0000	C:\\hal.dll	NoSymbols
	3	0.13	tid_1cc4	C:\\hal.dll	NoSymbols
	3	0.13	tid_1cc4	C:\\jvm.dll	** ## # I **
	3	0.13	tid_1cc4	C:\\win32k.sys	NoSymbols
	3	0.13	tid_199c	C:\\KERNEL32. dll	NoSymbols
	3	0.13	tid_1818	C:\\VETEFILE.SYS	NoSymbols
	3	0.13	tid_0158	C:\\ntoskrnl.exe	NoSymbols
	2	0.09	tid_1cc4	C:\\jite.dll	qsort
	2	0.09	tid_1cc4	C:\\jite.dll	dataflow_Q_copy
	2	0.09	tid_199c	C:\\CnsMinKP. sys	NoSymbols
	2	0.09	tid_0724	C:\\hal. dll	NoSymbols
	2	0.09	tid_0000	C:\\tepip.sys	NoSymbols
	1	0.04	tid_1e98	C:\\USER32.dll	NoSymbols 🧓
	1	0.04	tid loof	COV Assessment	rog info roors
	1				
	Thread	> Modu	ıle		

To lines of editor symbol table, user has to set the symbol threshold of the hierarchy. Symbol table contains symbols of the selected hierarchy node, but it does not list all the symbols in default. It often lists no more than a number of them. This is called threshold. After user sets the threshold to another value, the symbol table is refreshed and the listed symbols is no more than the new threshold. The default threshold of editor symbol table is 100, that is, no more than 100 symbols is listed in the table whatever hierarchy node is selected.

D:/profile/A50421C3.etm ×			- E	E
🖃 💁 A50421C3.etm (3462091.0 ticks) [default counter: T	Ticks	%]	Module 🔨	•
🖨 🏪 Process \succ Module	17594	17.39 /	opt/java/devline/J50413t/bin/lib/	
	L 9379	9.27 J	ITCODE	
😠 💽 PID 🄁 Resolve calls	6292	6.22 T	ITCODE	
🖬 💽 PID 🛛 Select default counter	5 574	5.51 J	ITCODE	
🖶 💽 PID 💾 Bucket Management	5232	5.17 J	ITCODE	
🖶 💽 PID 📲 Hierarchy Management	4739	4.68 J	ITCODE	
🖶 💽 PID 🚽	4329	4.28 J	ITCODE	
😐 💽 PID 嘗 Compare this profile with another	3878	3.83 J	ITCODE	
🖶 💽 PID	2532	2.50 J	ITCODE	
😨 💽 PID 😽 Populate CodeMiner Database	2293	2.27 J	ITCODE	
🙃 💽 PID 📝 Change Profile Symbol Threshold	2151	2.13 J	ITCODE	
🖲 🔝 PID 🔁 Load inlining information	2082	2.06 /	opt/java/devline/J50413t/bin/libj	
🗄 🖸 PID 🔜 Import line numbers from listing	1940	1.92 J	ITCODE	
	1876	1.85 J	ITCODE	
🗄 🖸 PID 2557 [PID 2557] (97821 ticks/2.83%)	1337	1.32 /	opt/java/devline/J50413t/bin/libj	
😨 🖸 PID 2574 [PID 2574] (97602 ticks/2.82%)	1239	1.22 J	ITCODE	
🖶 🖸 PID 2563 [PID 2563] (97590 ticks/2.82%)		1.12 J	ITCODE	
🖶 🧕 PID 2571 [PID 2571] (97297 ticks/2.81%)	1118	1.10 J	ITCODE	
🔒 💽 PID 2555 [PID 2555] (97230 ticks/2.81%)	1096	1.08 N	oModule	
🗊 💽 PID 2569 [PID 2569] (97124 ticks/2.81%)	939	0.93 J	ITCODE	
🗊 💽 PID 2564 [PID 2564] (97066 ticks/2.80%)	⊕ 923	0.91 J	ITCODE	
49 left		0.82 🕻		
🗈 🚰 Modules	807	0.80	symbols not more than default th	nrest
	750	0 74 4	111.1111	
	<			
	Total: ticks	(101200), t	ime share (2.92%) (first 100 rows	

If user selects the **Change Profile Symbol Threshold** item in the context menu, then it pops up the threshold box.

🚳 Change Profile Symbol Threshold 📃 🔲 🔀						
Select Thr Please sele	eshold ect symbol threshold of the profile.					
Threshold:						
	OK Cancel					

The default symbol threshold is 100. In the symbol table, there are no more than 100 rows listed.

🚳 Change P	rofile Symbol Threshold	
Select Three Please sele	eshold ct symbol threshold of the profile.	
Threshold:	200	
	OK	Cancel

If user sets the threshold to new value 200, the editor symbol table is refreshed and the number of the rows is no more than 200.

P:/profile/A50421C3.etm ×			
🖃 🐕 A50421C3.etm (3462091.0 ticks) [default counter: T	Ticks	%	Module
😑 🛱 Process \succ Module	17594	17.39	/opt/java/devline/J50413t/bin/libj
🗊 💽 PID 2577 [PID 2577] (101200 ticks/2.92%)	9379	9.27	JITCODE
🖬 💽 PID 2551 [PID 2551] (100698 ticks/2.91%)	6292	6.22	JITCODE —
💼 💽 PID 2579 [PID 2579] (100670 ticks/2.91%)	⊞ 5574	5.51	JITCODE
🖮 💽 PID 2582 [PID 2582] (100238 ticks/2.90%)	5232	5.17	JITCODE
😨 💽 PID 2554 [PID 2554] (99755 ticks/2.88%)	4739	4.68	JITCODE
😠 💽 PID 2580 [PID 2580] (99153 ticks/2.86%)	4329	4.28	JITCODE
🖻 [PID 2573 [PID 2573] (98704 ticks/2.85%)	3878	3.83	JITCODE
🖻 [PID 2575 [PID 2575] (98687 ticks/2.85%)		2.50	JITCODE
🖬 💽 PID 2568 [PID 2568] (98534 ticks/2.85%)	2293	2.27	JITCODE
🖬 💽 PID 2581 [PID 2581] (98326 ticks/2.84%)	2151	2.13	JITCODE
🖬 💽 PID 2576 [PID 2576] (98066 ticks/2.83%)	2082	2.06	/opt/java/devline/J50413t/bin/libj
🖬 💽 PID 2556 [PID 2556] (97986 ticks/2.83%)	⊞ 1940	1.92	JITCODE
🖬 [PID 2552 [PID 2552] (97833 ticks/2.83%)	1876	1.85	JITCODE
🖻 [PID 2557 [PID 2557] (97821 ticks/2.83%)	1337	1.32	/opt/java/devline/J50413t/bin/libj
🖻 💽 PID 2574 [PID 2574] (97602 ticks/2.82%)	1239	1.22	JITCODE
🖻 [PID 2563 [PID 2563] (97590 ticks/2.82%)		1.12	JITCODE
∎ [] PID 2571 [PID 2571] (97297 ticks/2.81%)	1118	1.10	JITCODE
⊕ [] PID 2555 [PID 2555] (97230 ticks/2.81%)	1096	1.08	NoModule
🖻 [PID 2569 [PID 2569] (97124 ticks/2.81%)	939	0.93	JITCODE
🖬 [PID 2564 [PID 2564] (97066 ticks/2.80%)	⊕ 923	0.91	JITCODE
49 left	⊞ 825	0.82	TITCODE
🖮 🚰 Modules	807	0.80	symbol rows no more than new thre
	750	0.74	11111111
	<		
	Total: ticks	(101200),	time share (2.92%) (first 200 rows

If user set the threshold to 'All', symbol table is refreshed and it lists all symbols of the threshold.

🚳 Change Profile Symbol Threshold					
Select Thr Please sele	eshold ect symbol threshold of the profile.				
Threshold:	A11				
	OK Cancel				

5.1.4.2Bucket Management

You can management your buckets settings by doing the followings:

- 1. Right-click in the process hierarchy view and choose Buckets-> Bucket Management
- 2. Attach a new bucket management file to the profile file
- Create a new bucket or edit/remove an existing bucket by clicking the corresponding buttons 3.

Sucket Lanagement						
Bucket Management Create, modify, delete Bucket Group and Buckets. attach a bucket file to the profile file						
Bucket File [Plugin]/config/bucket.xml [Default]	Save As					
🦳 Attach to D:/test files/line-numbers-Hello_java.out	Set As Default					
Buckets/ Buckets Groups:						
General Buckets	<u>N</u> ew Bucket					
test (enabled)	<u>E</u> dit bucket					
create a new bucket	Delete Bucket					
eidt/remove a bucket/bucket-group	Move to Group					
move down/up the buckets/buckets-group copy, rename a bucket/bucket-group	New <u>G</u> roup Delete Group					
Move <u>Up</u> Move <u>D</u> own <u>C</u> opy <u>R</u> ename						
Automatic/Enabled Enable	Disable					
The "Enable" and "Disable" buttons are only clickable when the current bucket file is associated with the profile of the active editor.						
OK Cancel	Apply					

If you have changed bucket definition, the current opened profile will be automatically reloaded. If there are any other opened profiles that are also affected by the new bucket definition, they will not be automatically reloaded. User must reload them manually.

5.1.4.2.1Add new bucket to existing one

If there are existing buckets to group some components, you can add one to another so as to further filter those components and get the views of those you really want.

🗜 D:/profile/line-numbers-Hello_java.etm × 🗧 🗖							
😑 훴 line-numbers-Hello_java.etm (8020.0 ticks) [default co	Ticks	% Module	^				
📮 🛱 Process > Module	331	12.71 JITCODE					
🖬 🧾 java. exe [PID 3356] (2605 +icke/32 49%)	264	10.13 D:\tr jit\latest.dev\jre\bin\j9jit23.	d				
🖬 💽 javaw.ex 🎏 Resolve calls	239	9.17 D:\tr_jit\latest.dev\jre\bin\j9thr23.	d 🔳				
🖬 💽 IdleProc 🛛 Select default counter	▶ 198	7.60 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
🙃 💽 System [💾 Bucket Management	141	5.41 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	±1				
🗊 🧾 vsmon.ex 📲 Hierarchy Management	116	4.45 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	11				
🖬 🖸 CMD. EXE		4.38 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
🗈 🧕 NotesBud 🚰 Compare this profile with another	94	3.61 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	11				
🗈 🧾 agent. ex	87	3.34 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	11				
🗈 🧾 csrss.ex 🔕 Populate CodeMiner Database	80	3.07 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
🗈 🧾 iclient. 📝 Change Profile Symbol Threshold	75	2.88 D:\tr_jit\latest.dev\jre\bin\j9vm23.c	11				
E C emacs. exe [FID SU(0] (22 (ICKS)0. 2(%)	74	2.84 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	11				
· · · · · · · · · · · · · · · · · · ·	62	2.38 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
⊕ [emacs.exe [PID 2748] (19 ticks/0.24%)	53	2.03 C:\WINNT\System32\ntoskrnl.exe					
	48	1.84 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
⊕ 💽 javaw.exe [PID 2724] (11 ticks/0.14%)	42	 1.61 D:\tr_jit\latest.dev\jre\bin\j9jit23. 	d				
NLNOTES. EXE [PID 1300] (10 ticks/0.12%)	40	1.54 D:\tr_jit\latest.dev\jre\bin\j9jit23.	d				
	37	1.42 D:\tr_jit\latest.dev\jre\bin\j9thr23.	d				
Explorer.EXE [PID 1084] (9 ticks/0.11%)	33	1.27 D:\tr_jit\latest.dev\jre\bin\j9vm23.0	11				
	30	1.15 D:\tr_jit\latest.dev\jre\bin\j9vm23.d	11				
	27	 1.04 D:\tr_jit\latest.dev\jre\bin\j9thr23. 	d				
26 left	24	0.92 C:\WINNT\system32\MSVCRT.dll					
	23	0.88 D:\tr_jit\latest.dev\jre\bin\j9thr23.	d				
	- 21	0 81 C·\WINNT\Svstem32\bal_dll					
• • • • • • • • • • • • • • • • • • •	Total: tick:	s (2605), time share (32.48%) (first 100 rows)				

If clicked, the bucket selection dialog will open.

🖨 Add to bucket		
General Buckets test2 test3 test4 test5		
	OK	Cancel

After selecting a bucket, the bucket property dialog will open for users to modify the filters as follows:

Bucket Properties	\mathbf{X}
Bucket Properties Edit bucket properties.	
Bucket Name test3	Type Thread 💌
Process filter	
Thread filter	tid* tid_12f8
System Attribute filter	
Module filter	
Object File filter	
Platform filter	
Profile filter	
Startup mode 💽 Automatic	C Manual
Bucket Group General Buc	kets 💌
	OK Cancel

You can add the filter requirement in this wizard and give this bucket a new name. In order to view this bucket in **Process Hierarchy View**, you may move it up in the **Bucket Management**. You can also edit, delete, enable, disable bucket or create new bucket group in **Bucket Management**.

5.1.4.2.2Create new bucket to group common components

In a system, certain groups of modules, threads, or symbols share common components. For example, in a WebSphere java process, threads can be logically grouped by name, with one group containing threads with names like tid_WebContainer*, another with names like tid_Gc_Slave_Thread_*, and another with names like _tid_Alarm_*. Buckets function in Profile Analyzer provides mechanisms for users to group different components, such as objects, together into buckets. It acts as a new layer in the **Hierarchy Process View**.

To create a new bucket, please follow these steps:

- 1. Right-click in the process hierarchy view and choose Buckets-> Create new bucket
- 2. Give you **bucket Name** and choose the type of components you want to group from thread, process, module
- 3. Give the components' name you hope to filter in the corresponding blank, such as tid* thread in thread filter
- 4. Choose the **bucket group** you want to put your new bucket into.

The following is a example of creating a bucket filtering tid* thread

8	
Bucket P Sets bucke	roperties ets' properties.
Bucket Na	me: test2
Scope	
Type:	Thread Croup: General Buckets 💌
Filters	
Process:	
Thread:	tid*
Module:	
Symbol:	
	(* for any string, ? for any character, and separate patterns by ' \mid '.)
Notes: To	apply the bucket definition, related editor(s) should be reloaded.

Now you can define multiple filters for a bucket.

You can see a new layer called test2 appeared above the thread hierarchy view. All the threads with the title beginning **tid** remain as follows:

🚏 17235867. out 🗙 💾 line-numbers-Hello_java. out						- 8			
17235867. out									
🖃 🐂 Profile 17235867.out (2186 ticks)	~	Ticks	%	Thread	Module	Symbol 🔨			
🕀 🏪 Process > Module > Thread		2029	92.8	tid_0000	C:\\intelppm.sys	NoSymbols			
🖻 📲 Thread > Module		47	2.15	tid_199c	C:\\VETEFILE.SYS	NoSymbols			
😑 💾 test2 (2186 ticks/100.0%)		15	0.68	tid_199c	C:\\ntoskrnl.exe	NoSymbols			
🕂 🧭 tid 0000 (2035 ticks/93.0%)		11	0.50	tid_199c	C:\\hal. dll	NoSymbols 🗧			
+ 1 d 199c (83 ticks/3 79%)		4	0.18	tid_1cc4	C:\\ntoskrnl.exe	NoSymbols			
≈ 2 tid lood (32 ticks) (1.46%)		4	0.18	tid_199c	C:\\ntdl1.dl1	NoSymbols			
$= \frac{1}{2} \left(\frac{1}{12} \left(\frac{1}{2} \left(\frac{1}{12} \right) \right) \right) \right) \right) \right) \right)$		4	0.18	tid_12f8	C: \ \VETEFILE. SYS	NoSymbols			
$+ \frac{1}{2} \left(\frac{1}{2} \frac{1}{2}$		4	0.18	tid_0000	C:\\hal.dll	NoSymbols			
+ 11d_0158(5_ticks/0.22%)		3	0.13	tid_1cc4	C:\\hal.dll	NoSymbols			
		3	0.13	tid_lcc4	C:\\jvm.dll	*****			
🕀 🚀 tid_12f8(4 ticks/0.18%)		3	0.13	tid_lcc4	U: \ \win32k. sys	NoSymbols			
🕀 🧭 tid_1818 (4_ticks/0.18%)		3	0.13	tid_199c	C: \ \KEKNEL32. dll	NoSymbols			
🗄 🧪 tid_1c58(2_ticks/0.09%)						3	0.13	tid_1818	C: V VELEFILE, STS
🕂 🧭 tid 18f4(1 ticks/0.04%)			0.13	tid_0150	C.V Vntoskrni, exe	NoSymbols			
+ / tid 01b4(1 ticks/0 04%)		2	0.09	tid_lcc4	C.V. Vite all	qsort			
\Rightarrow $id_0330(1+i_0)r_0(0.04\%)$		2	0.09	t1d_1cc4		datariow_Q_copy			
		2	0.09	tid_1990	C. L. L. ULISMINAR, SYS	NoSymbols N.SL.L.			
\pm γ tid_U334(I ticks/0.04%)		2	0.09	110_0124	C. V Vinal. dil	NoSymbols N.SL.L.			
		1	0.09	tia_0000	C.\.\.\	NoSymbols N-S-mbols			
🕀 🐓 tid_00e8(1 ticks/0.04%)		1	0.04	tid_les0	C.V. Viite 411	NoSymbols			
庄 🧪 tid_06b8(1 ticks/0.04%)		<				>			
	~	Thread	> Modu	le					

5.1.4.3Navigate Java package hierarchy

You can view the Java package hierarchy for a process, thread, or module that contains JITted methods using the **Java/Hierarchy** view. The process or thread must contain a JITCODE module, and the module must be the JITCODE module.

Note: The Java/Classes hierarchy view is normally displayed at the bottom right, along with the Disassembly/Offsets view, the Temporal Profiling view, and the Profiling Configurations Console view. If you cannot see it displayed, you can open it with Windows -> Show View -> Other -> Profile Analyzer-> Java/Classes hierarchy.

To view the Java package hierarchy for a process, thread, or JITCODE module, click on the **Java/Classes hierarchy** tab, then navigate the **Process hierarchy** view. As you select different processes, threads, or modules, the Java/classes hierarchy is updated to show you the Java package hierarchy for any active methods. The following screen shot shows an initial view of the Java package hierarchy for a JITCODE module on a AIX profile:

Disassembly/Offsets Source Code Compiler List	ing Prof	ile Detai	ls 🚹 J	ava/Hierarchy 🗙 Profile Comparison Temporal Profiling 🎇 🖵 🗖				
Class Hierarchy for: JITCODE module								
		%	% R	Method/Function Name				
		6.59	100.00	Hello. foo(I)V				
⊡… <mark>⊘</mark> io								
🔤 🛃 CharToByteConverter								
				A				
				0				
	<							
Selected {Hello.}	(+) Tot:	al: ticks	(331), t	ime share (6.59) %				

You can select a top-level name to view all methods in all packages that match that name (for example java). Or you can expand a top-level name to display the packages and classes beneath it. By selecting a package or class in the hiearchy, you can limit the list of displayed methods to those in that class or hierarchy. The following shows the active methods for the BufferedWriter class in the java.io package:

Disassembly/Offsets Source Code Compiler List	ing Profi	le Detai	1s 🚺 J	ava/Hierarchy 🗙 Profile Comparison Temporal Profiling 🎇 🗁 🗖			
Class Hierarchy for: JITCODE module							
r	Ticks	%	% R	Method/Function Name			
🚊 🚺 java	2	0.04	66.67	java/io/BufferedWriter.write(Ljava/lang/String;II)V			
🖻 🚺 io	1	0.02	33.33	java/io/BufferedWriter.flushBuffer()V			
🚽 🗊 FileOutputStreag							
🕒 🗈 PrintStream							
🗄 🚺 sun							
Ė <mark>0</mark> 7 io							
🔄 🚹 CharToByteConverter							
	<						
Selected {java/io/BufferedWriter.}	(+) Total	.: ticks	(3), tim	e share (0.06) %			

When you double-click on a method in the table at the right, the following views are updated to display information for that method:

- The OffsetAsm Information view
- The Disassembly Resovled Call Information view

5.1.4.3.1Notes on appropriate use of this view

The Java/Classes Hierarchy view is useful if you are working on tuning the code for specific classses under your control and are trying to determine what bottlenecks exist in those classes. However, you should avoid the pitfall of focusing simply on the classes you have control over (classes that you can make source code changes to). For example, trying to tune the hottest method in a class you own may provide some benefit, but if that method takes only 1% of total ticks, while java/io/BufferedWrite.write takes 20%, you should look at what methods are calling java/io/BufferedWrite.write (using the Disassembly Resovled Call Information view on Windows, Linux-IA32, and Linux-x86-64, or using a call graph profiling tool such as ITRACE on other platforms).

Conversely, just because the methods in your packages do not show significant CPU usage does not mean they are efficiently written or have no impact on performance. For instance, if a significant percentage of profile time is spent in the JVM garbage collection library (e.g. libj9gc23.so or j9gc23.dll), this may indicate that you are making inefficient use of memory by allocating too many objects or failing to make them available to garbage collection when they are no longer needed.

5.1.4.4View counters

In **Counters** view, you can view the ticks of process, thread, module or bucket you selected in the **Process Hierarchy View**. To open this view, choose **Window -> Show View -> Others->Profile Analyzer -> Counters** or just find it in left pane.

For example, if you select a process in the **Process Hierarchy View**, you can see the total ticks of this process in the **Counters** view

Bits Start Sugark (but Spinster Bols) ************************************	🚭 Profile Analyzer - D:/test files/line-numbers-	Hello_java.out - Visual Performance Analyzer			
Image: Status Conscience Image: Status C	<u>F</u> ile <u>E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> elp				
Profiling Reserves: S2 Provide Version (Phile] 🔣 📊 🔯] 🍫] 🖋				
Subschlaft Source Trick Normality Normality <thnormality< th=""> Normality <</thnormality<>	Profiling Resources 🛛 Database Connections 🎽 🗖	🚰 D:/test_files/line=numbers=Hello_java.out 🗙			- 8
Suplat Mistrikut	bo 🗣 🕨 🗂 😂 📬 😼	🖃 🐂 line-numbers-Hello_java.out (5020.0 ticks) [default counter: Ticks]	Ticks	% Thread	Module
Seeples Distribut De Counters 22 Baie Blocks "? Symbol: Bosymbols Counter: Count Tick: 604 Diserremblay/Offsets Source Code Compiler Listing Profile Details Jour/Hierarchy 22 Profile Comparison Temporal Profiling "? Diserremblay/Offsets Source Code Compiler Listing Profile Details Jour/Hierarchy 23 Profile Comparison Temporal Profiling "? Diserremblay/Offsets Source Code Compiler Listing Profile Details Jour/Hierarchy 23 Profile Comparison Temporal Profiling "? Diserremblay/Offsets Source Code Compiler Listing Profile Details Jour/Hierarchy 23 Profile Comparison Temporal Profiling "? Diserremblay/Offsets Source Code Compiler Listing Profile Details Jour/Hierarchy 23 Profile Comparison Temporal Profiling "? Comment salection does not contain a JITCODE module Tick: \$\% \$\% R Hethod/Function Hase Comment salection does not contain a JITCODE module Comment salection does not contain a JITCODE module Comment salection does not contain a JITCODE module Comment salection does not contain a JITCODE module	- DICALHOST (Local)			99.83 (iid_0000 [TTD 0] 0.17 (iid_0000 [TTD 0]	C:\WINDT\System32\hal.dll C:\WINDT\System32\Drivers\
Counter Count Ticks 604 Disassembly/Offsets Source Code Compiler Listing Profile Details Jave/Hierarchy X Profile Comparison Temporal Profiling Pt Current selection does not contain a JITCOBE module. Ticks % % R Method/Function Name Item in the intervence of the interven	Samples Distribut Counters 🕸 Basic Blocks 🎽 🗖 🗖 Symbol: WoSymbols	Rtvrcan.exe [FID 1044] (6 ticks/0.12%) 24 left Er Frocess > Module E Module	<	s (605) time share (12 05%)	(2 ross)
Ticks 604 Current selection does not contain a JITCOBE module. Ticks \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$	Counter Count	Disassambly/Offsats Source Code Compiler Listing Profile Details I Tawa/Hie	varaby S?	Profile Comparison Tempore	Profiling », -
Ticks % <td>Ticks 604</td> <td>Current selection does not contain a .IIICODE module.</td> <td></td> <td>Tiorrie comparison Tempora</td> <td></td>	Ticks 604	Current selection does not contain a .IIICODE module.		Tiorrie comparison Tempora	
		Tic	ks %	% R Method/Function Na	ne

5.1.4.5Select default counter

Whenever you have a profile that contains more than one counter, Profile Analyzer will allow you to choose which is the default counter, used for sorting. Profile Analyzer supports the "sort by counter" feature. A user can select any available counter as the default "sort" counter. Once the default counter is selected, all editors and views understand this selection and will sort/format their outputs according to the current active "sort" counter. Following picture shows this feature:

💾 D:/test files/line-numbers-Hello_java.out 🦳 📴 D:/test files/A50421C3.etm 🗙					- 8
🖃 🔧 A50421C3.etm (3462091.0 ticks) [default counter: Ticks]		Ticks	%	Module	~
F F Process > Module		17594	17.39	/opt/java/devlin	.e/J5041
🕀 🚺 PTD 2577 [PTD 2577] (101200 ticks/2 92%)		9379	9.27	JITCODE	
PTD 2551 [PTD 2551] (100 - p		6292	6.22	JITCODE	
TID 2551 [IID 2551] (100 2 Kesolve calls		∓ 5574	5.51	JITCODE	
The rib 2519 [rib 2519] (100 Select default counter	🗸 ticks	232	5.17	JITCODE	
HID 2562 [FID 2562] (100 Bucket	IFC	739	4.68	JITCODE	
H [] PID 2554 [PID 2554] (997	AGT B1 1-D	329	4.28	JITCODE	
🕀 💽 PID 2580 [PID 2580] (991 💗 Mierarchy Management	AGIDIKD	· 878	3.83	JITCODE	
⊕ 💽 PID 2573 [PID 2573] (987	USC	532	2.50	JIICODE	
🕀 💽 PID 2575 [PID 2575] (986 🗄 Compare this profile with another	CachMs	293	2.21	JIICODE	
🗄 💽 PID 2568 [PID 2568] (985	DCDirMs	151	2.13	JIICODE /ant/iawa/dawliw	×/TE041
🕀 🚺 PID 2581 [PID 2581] (983 🍜 Populate CodeMiner Database	DLATMs	002	1.92	TTTCODR	16/3041
F O PID 2576 [PID 2576] (980	TCD	876	1.85	TTTCODE	
PTD 2556 [PTD 2556] (97986 ticks/2 83%)	ICDITMS	337	1.32	/opt/java/devlir	w/T5041
PTD 2552 [PTD 2552] (07833 +i.ke/2 83%)	PSC	239	1.22	TITCODE	
	CPI	132	1.12	TITCODE	
TID 2331 [TID 2331] (91021 (1CKS/2.03%)		1118	1.10	JITCODE	
H [] FID 2514 [FID 2514] (91602 ticks/2.82%)		1096	1.08	NoModule	
(4) [1] 2563 [PID 2563] (97590 ticks/2.82%)		939	0.93	JITCODE	
		+ 923	0.91	JITCODE	
		+ 825	0.82	JITCODE	
🕀 🔁 PID 2569 [PID 2569] (97124 ticks/2.81%)		807	0.80	JITCODE	
🔁 💽 PID 2564 [PID 2564] (97066 ticks/2.80%)		750	0.74	JITCODE	
49 left		742	0.73	JITCODE	
		666	0.66	/opt/java/devlir	ie/J5041
		+ 651	0.64	JITCODE	
		+ 643	0.64	JITCODE	
		531	0.52	JITCODE	
		530	0.52	JIICODE	
		524	0.52	JIICODE	×
		<			>
P		Total: ticks	(101200),	time share (2.92)	6) (first

💾 D:/test files/line-numbers-Hello_java.out 💦 👫 D:/test files	/A50421C3.etm 🗙					
🖃 🔧 A50421C3.etm (1307019.0 ticks) [default counter: IFC]	*	~	IFC	%	Symbol/Functions	~
□ 🗜 Process > Module			3478	11.15	<pre>spec/jbb/infra/Coll</pre>	ectio
E T PTD 2551 [PTD 2551] (38953 ticks/2 98%)	The default counter		3203	10.27	<pre>spec/jbb/infra/Util;</pre>	/Disp
TTTCODE (31182 +i dec /80 05%)	The default counter		2736	8.77	spec/jbb/infra/Util,	/Disp
$\prod_{i=1}^{n} (1 + 1) = (1$	change to IFC		2665	8.55	spec/jbb/infra/Coll	ectio
((31) (01) (01) (01) (01) (01) (01) (01) (0			1681	5.39	spec/jbb/infra/Util,	/Disp 📄
1 //libj9jit23.so (2/2 ticks/0.(0%)			1109	3.56	java/util/Calendar.;	get (I
Mucleus (90 ticks/0.23%)			1109	3.56	spec/jbb/NewUrderTr	ansac
MoModule (88 ticks/0.23%)			1053	3.38	spec/jbb/Urder.proc	essLi
			+ 936	3.00	spec/jbb/Urderline.j	proce
			+ 767	2.46	java/util/Hashtable.	. put l
			+ 675	2.16	spec/jbb/StockLevel.	Irans
[] / /libi9prt23. so (2 ticks/0.01%)			668	2.14	spec/jbb/raymentira	nsact
/ /libi9tbr23 so (1 ticks/0 00%)			632	2.03	spec/jbb/infra/coll spec/jbb/DeliwerwTr	ectio
(1 (librthroad=0.60 rs)(0 (librthroad=0.60 rs))			+ 003	1.55	<pre>spec/jbb/beliveryll</pre>	ansac
T T PT 2570 [PTD 2570] (20000 A: -1- (2 000)			323	1.59	spec/jbb/infra/l[ti]	/Disp
			473	1.52	spec/jbb/infra/l[ti]	/Disp
H [] FID 2582 [FID 2582] (38869 ticks/2.9(%)			408	1.31	<pre>spec/jbb/infra/Util</pre>	/Disp
			328	1.05	java/util/Gregorian	Calen
⊞ 💽 PID 2554 [PID 2554] (38394 ticks/2.94%)			326	1.05	spec/jbb/infra/Util	/Disp
🕀 💽 PID 2568 [PID 2568] (38078 ticks/2.91%)			+ 297	0.95	java/util/Gregorian	Calen
🕀 💽 PID 2576 [PID 2576] (38066 ticks/2.91%)			285	0.91	spec/jbb/infra/Util,	/Disp
🕀 💽 PID 2573 [PID 2573] (38001 ticks/2.91%)			+ 273	0.88	spec/jbb/infra/Fact	ory/F
🕀 💽 PID 2575 [PID 2575] (37933 ticks/2.90%)			271	0.87	java/util/Hashtable	\$Hash
🕀 💽 PID 2574 [PID 2574] (37901 ticks/2.90%)			265	0.85	java/util/Hashtable	\$Hash
F PTD 2581 [PTD 2581] (37874 ticks/2 90%)			254	0.81	spec/jbb/infra/Coll	ectio
TID 2001 [TID 2001] (01014 (TCR3) 2.000)		-	+ 251	0.80	java/util/Hashtable	reha
[1] = [1] = 2000 [1] = 2000 [0000 (0000 (0000))]			+ 246	0.79	<pre>spec/jbb/infra/Coll</pre>	ectio
TI 2002 [TI 2002] (01000 ticks/2.90%)			+ 243	0.78	spec/jbb/infra/Coll	ectio
H 10 2564 [FID 2564] (3(761 ticks/2.89%)			221	0.71	sun/util/calendar/Z	oneIn
[] PID 2559 [PID 2559] (37724 ticks/2.89%)			207	0.66	spec/jbb/infra/Fact	ory/C 💙
🕀 [PID 2557 [PID 2557] (37693 ticks/2.88%)			<			>
🗊 👘 PID 2556 [PID 2556] (37688 ticks/2.88%)		$\mathbf{\mathbf{v}}$	Total: ticks	(31182),	time share (2.39%) (first 1

5.1.4.6View module sample distribution

The **Samples distribution chart** shows a tick distribution for modules in the currently selected process or thread in the process hierarchy.

Note: If you cannot see this view from within the Profile Analyzer perspective, select Windows -> Open view -> Other -> Profile Analyzer -> Samples Distribution Chart.

This view provides a starting point for determining where you should focus your attention. For example, the following screenshot shows the graph for a java process using 23.4% of total profile time:



This graph shows that the JITCODE module (the module containing JIT-compiled Java methods) was the busiest module for this process, which suggests that some tuning of Java methods may be advisable.

The following graph (for a different Java program on a different system) shows heavy activity both in j9gc22 and in JITCODE. j9gc22 is the Garbage Collection library of the IBM Virtual Machine for Java (J9) indicating that the application may be memory constrained, or may be creating new objects too frequently. The third column (j9jit22) is the JIT compiler library, indicating that the profile may not have run for very long, since long-running applications typically have a small percentage of time used by the JIT. (A high percentage of time in the JIT may also indicate excessive use of invoke_interface calls, which require JIT library runtime support even when executing JITted methods).



5.1.4.7View profile details

You can see the detail information of a profile file when you select it in the **Process Hierarchy View**. We can see it from the following example:

🚰 D:/test files/line-numbers-Hello_java.out 🛛 🚰 D:/test files/A	5042163.etm 23	🚰 D:/test files/tprof_e.out	
	IFC	% Symbol/Functions	~
Process > Hodyle	3478	11.15 spec/jbb/infra/Collections/Str	ingBTre
	3203	10.27 spec/jbb/infra/Util/DisplayScr	een. put
- [] [I] 2551 [[I] 2551] (30955 (16K5/2.90%)	2736	8.77 spec/jbb/infra/Util/DisplayScr	een. put
TILUDE (31182 ticks/80.05%)	2665	8.55 spec/jbb/infra/Collections/lor	gStatic
1 //libj9gc23.so (7297 ticks/18.73%)	1681	5.39 spec/jbb/infra/Util/DisplayScr	een. put 😑
//libj9jit23.so (272 ticks/0.70%)	1109	3.56 java/util/Calendar.get(I)I[sco	rching]
Mucleus (90 ticks/0.23%)	1109	3.56 spec/jbb/NewOrderTransaction.s	econdDi
	1053	3.38 spec/jbb/Order.processLines(Ls	spec/jbt
	+ 936	3. UU spec/jbb/Urderline. process (Lsp	ec/jbb/
	+ 767	2.46 java/util/Hashtable.put(Ljava/ 2.46 marchill(Stable.put)Transaction	lang/Ut
	+ 015	2.10 spec/jbb/StockLevellransaction 2.14 spec/jbb/PermontTropsection sc	. proces
	632	2.03 spec/jbb/infra/Collections/lor	BTreek
	■ 603	1.93 spec/jbb/DeliveryTransaction.	ueue OV
	525	1.68 spec/jbb/infra/Collections/lor	zBTreeN
FIN PID 2579 [PID 2579] (38886 ticks/2.98%)	+ 497	1.59 spec/jbb/infra/Util/DisplayScr	een. put
F T 2582 [PTD 2582] (38869 ticks/2 97%)	473	1.52 spec/jbb/infra/Util/DisplayScr	een. put
TID 2002 [TID 2002] (30000 (TCR3)2: 51%)	408	1.31 spec/jbb/infra/Util/DisplayScr	een. put
TID 2511 [TID 2511] (30105 (TCKs/2, 50%)	328	1.05 java/util/GregorianCalendar.co	mputeFi
TID 2004 [TID 2004] (00054 (TCKS/2.54%)	326	1.05 spec/jbb/infra/Util/DisplayScr	een. put
	+ 297	0.95 java/util/GregorianCalendar.co	mputeFi
+ O PID 2576 [PID 2576] (38066 ticks/2.91%)	285	0.91 spec/jbb/infra/Util/DisplayScr	een. put
	+ 273	0.88 spec/jbb/infra/Factory/Factory	r. create
🕀 💽 PID 2575 [PID 2575] (37933 ticks/2.90%)	271	U.87 java/util/Hashtable\$HashEnumer	ator.hs
🗄 💽 PID 2574 [PID 2574] (37901 ticks/2.90%)	265	0.05 java/util/Hashtable>HashEnumer	ator. ne
🕀 💽 PID 2581 [PID 2581] (37874 ticks/2.90%)	254	0.80 jawa/util/Harbtable weberb()/	igdiree.
🕀 💽 PID 2563 [PID 2563] (37861 ticks/2.90%)	+ 251	0.79 spec/ibb/infre/Collections/los	BTreek
🗈 💽 PID 2552 [PID 2552] (37850 ticks/2.90%)	+ 240	0.78 spec/jbb/infra/Collections/Str	ingStat
🕀 💽 PID 2564 [PID 2564] (37761 ticks/2.89%)	221	0.71 sun/util/calendar/ZoneInfo.get	Transit
🕀 🔽 PID 2559 [PID 2559] (37724 ticks/2.89%)	207	0.66 spec/jbb/infra/Factory/Contain	er.allo
F 7 PID 2557 [PID 2557] (37693 ticks/2.88%)	<		>
E T 2556 [PID 2556] (37688 ticks/2 88%)	Total: ticks	s (31182) time share (2 39%) (first 100	rows
Disassembly/Offsets Source Code Compiler Listing 🎇 Profile Detail:	s 🗙 🛛 Jawa/Hie	erarchy Profile Comparison "3	
Tax as	13 I		
Field Value			
File D:/test files/A	50421C3.etm	101.00	
Uriginal file Elletune_profil Currented en Tue Tul 10 01/2	es\My Files\AbU M.EO CST 2005	42163.etm	
Available CPUs 16	4.00 051 2000		
Active CPUs 16			
Active CPUs 16 CP Vtilization % 89.464			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Lateration (Target Content of the second			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.566 Cycle Target 1.986			
Active CPUs 15 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 18404.250 ETR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MFS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184040.250 ETR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.586 Cycles/Instruction 1.988 Total MIPS 9668.148			
Active CPUs 16 CP Utilization % 69,464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 5257.566 Cycles/Instruction 1.988 Total MIPS 9668.148			
Active CPUs 16 CP Utilization % 69,464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 5257.566 Cycles/Instruction 1.988 Total MIPS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 FTR 128095.500 Instructions/Tran 52537.586 Cycles/Instruction 1.988 Total MIFS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.586 Cycles/Instruction 1.988 Total MIFS 9688.148			
Active CPUs 16 CP Utilization % 69,464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9688.148			
Active CPUs 16 CP Utilization % 69,464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ETR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9668.148			
Active CPUs 16 CP Utilization % 69,464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ITR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9688.148			
Active CPUs 16 CP Utilization % 69.464 Cycle Time 0.831 Elapsed Seconds 600.202 ITR 184404.250 ITR 128095.500 Instructions/Tran 52537.566 Cycles/Instruction 1.988 Total MIPS 9688.148			>

A file A50421C3.out is opened, and its corresponding information shows up in the **Profile Details** View.

5.1.4.8View resolved call information

When you open an IA32 profile, Profile Analyzer can analyze the disassembly in it to identify all call sites that have an immediate address as a target, and can attempt to connect those call sites to target symbols. This is done automatically for you if the **Resolved Call Information** view is visible. The following shows the **Resolved Call Information** view for a method selected from the jvm.dll module:

	wigator - tp	rof_e.out	- Eclip	se SDK				- P
ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>I</u>	<u>P</u> roject Run Test	s <u>R</u> un <u>W</u> indov	v <u>H</u> elp					
📬 • 📄 🗁] 💁 -] 🛷]	22214	• \$ • \$ •					E I	Visual Per
📲 Profiling Resources 🖾				💾 line-number	s-Hello_java.out	💾 listingl.etm	🟴 tprof_e. out 🗙	- [
 LOCALHOST (Local) New Configuration Second Configuration OS-8-5 下午1:35 - Third Configuration Copy of New Configuration DBM-0C016B0E884 (Window New Folder New Folder C:\profiles\listingl C:\profiles\listingl C:\profiles\listingl C:\profiles\listingl C:\profiles\listingl C:\profiles\listingl Secolved Calls S Symbol: ***MMI** 	C:\runtime-work sation s) etm out shers-Hello java	space\Performs	nce I	tprof_e.out	<pre>tprof_e. out (3494 ess > Module > Thr ad > Module est5 (3450 ticks/98 c:\\signal (345) c:\\sign</pre>	ticks) ead .7%)) ticks/20.4%) ticks/13.6%) (6 ticks/13.6%) (6 ticks/13.6%) exe(6 ticks/13.6%) ticks/4.54%) DLL(2 ticks/4.54%) (1 ticks/2.27%) 1 ticks/2.27%) SYS(1 ticks/2.27%)	Ticks % Symbol 2 22.2 **MMI** 1 11.1 resolvel 1 11.1 clkesolvel 1 11.1 clkesolvel 1 11.1 clkedur	ClastonstantFromPC Alloc enceCounter 8Entry veConstantPoolMethod lassFromClassLoader F6String
Module: C:\Java\j2sdk142\jre\	bin\classic\jvm.	111			C: \ \MSVCRT. dl	l(1 ticks/2.27%)	Total: ticks (9), tim	e share (20.4%)
Module: C:\Java\j2sdk142\jre\ rocess: javaw.exe_cb8, java.e: Known callers (calls to here;	bin\classic\jvm. xe_b40, javaw.ex ticks in caller)	411 e_c7c		Disassembly	C:\\MSVCRT.dl	l(l ticks/2.27%)	Total: ticks (9), tim	e share (20.4%) → ★ ↓ → ▽ □ □
Module: C:\Java\j2sdki42\jre\ rocess: javaw.exe_cb8, java.e: Known callers (calls to here; Symbol	bin/classic/jvm. xe_b40, javaw.ex ticks in caller) Calls	dll e_c7c) Ticks		Disassembly Offsets for: ** Offset	C:\\MSVCRT.dl /Offsets & MMI**(all processe Bytes	l(1 ticks/2.27%) es, no loop detection Disassembly	Total: ticks (9), tim	e share (20.4%)
Module: C:\Java\j2sdki42\jre\ rocess: javaw.exe_cb8, java.ex Known callers (calls to here; Symbol Known descendants (calls to d Symbol clIsInstanceOf resolveClassConstantFromPC xeDeleteNativeFrame	bin\classic\jwm. xe_b40, javaw.ex ticks in caller) Calls escendant from al Calls 9 7 1	dll e_c7c) Ticks Ll sources: tin Ticks 1 1 1	cks in d	Disassembly Offsets for: ** Offset	C: \ \MSVCRT. dl	l (1 ticks/2.27%)	Total: ticks (3), tim	e share (20.4%)

5.1.4.9View basic blocks

A basic block is a block of instructions that contain a single entry point and at most two exit points. Basic blocks are a concept used by compilers to perform dataflow analysis and to perform effective optimizations. Profile Analyzer attempts to detect basic blocks by analyzing the targets of all branch instructions within the disassembly for a symbol. Note that the basic blocks detected by Profile Analyzer may not match the basic blocks indicated in a compiler listing, as the compiler may use a higher-level basic block structure that includes internal branches. For example, a single source or intermediate-language instruction would likely not span multiple basic blocks from a compiler perspective. However, some source or intermediate-language instructions may result in multiple basic blocks at the disassembly level. An array assignment operation in Java is one such instance: the assignment is a single source statement, but may require both a null check and an array bounds check, each of which are intermediate-language instructions that may result in multiple conditional branches in the resulting disassembly.

You can see basic block information by choosing **Windows-> Show View -> Other... -> Profile Analyzer -> Basic Block** or just find it in the left panes.

When you open a process, you can see its basic block information as follows:



Each basic block has a number (BB1, BB2 etc.), a tick count, zero or more incoming edges, and one or two outgoing edges (a terminating basic block does not have any outgoing edges). Each block with ticks is colored red, magenta or blue according to the same rules used to determine symbol tick color, and shaded according to the relative tick count of the basic block as compared to the symbol as a whole.

You can click on a basic block to highlight its incoming and outgoing edges in red:



In this view, BB2 was selected, and its outgoing edges to BB3 (the "fallthrough" basic block) and BB4 (the target basic block) are highlighted.

5.1.5Profile Comparison

The following are tasks you can do to compare profiles within Profile Analyzer:

5.1.5.1Compare two profiles

- 1. Click the **Compare two profiles** button in the **Profile Comparison** view toolbar. Alternatively, for any open profile, right-click in the process hierarchy view and select **Compare this profile with another**.
- 2. In the Profile Comparison Wizard, select two profiles you want to compare. The wizard supports compressed (.etz) profiles. Comparison of tprof profiles is still limited in that it does not expose compilation levels.
- 3. Click Next. Enter the following values for each profile:
 - Transaction rate: The transaction rate is the number of transactions completed per elapsed second. What we call a transaction varies from workload to workload, but it is a consistent indicator of the amount of work we are doing per second. If we are running background jobs, the number of transactions might be the number of jobs. If we are running an HTTP server, the number of transactions might be the number of HTTP requests we have served. For each type of workload, the transaction is clearly defined.
 - **CPU utilization:** CPU utilization is a percentage that describes how busy servers are. It is defined as the average utilization of all CPUs in a server.
 - Number of CPUs: This is the number of CPUs available to the system in each of the profile runs being compared. It is very common that we compare runs with different number of CPUs. For example, we may compare a 1-CPU run against a 4-CPU run to determine how well a workload scales up as we add CPUs to the configuration.
- 4. Click **Finish**. Two files are loaded and opened.

5. Right-click the process, thread or module of the first file in the process hierarchy view which you want to compare and select **Mark for comparison.**

🗜 E:/TOOLS/CodeAnalyzer/ppc/sort.etm 🗙 👫 //9.186.13.72/	data/sample	profile/GY/winxp_20051026.out	
E:/TOOLS/CodeAnalyzer/ppc/sort.etm (5329.0 ticks) [Freeses > Thread > Module Sort [PID 380956 Sort TID 11346 Sort (30) (Index) (1 time) (Index) (Index) (In	ticks 1712 1991	% Thread 55.33 sort TID 1134669 44.63 sort TID 1134669 0.03 sort TID 1134669 • 0.03	Mo ^ [TID 1134669] ./s [TID 1134669] ./s [TID 1134669] /un
vpn_tprof [PID 78060] (1 ticks/0.02%) sh [PID 380956] (1 ticks/0.02%) ⊕	<		

6. Go to the other profile, right-click the process, thread of module of the second file which you want to be compared with and select **Compare with <filename>.**

💾 E:/TOOLS/CodeAnalyzer/ppc/sort.etm 💦 👫 //9.186.13.72/0	data/sample	profile/GY/	winxp_20051026.out 🗙	
🖃 🔧 //9.186.13.72/data/sample_profile/GY/winxp_2005102 📈	ticks	%	Module	~
	6608	99.59	C:\WINDOWS\System32\D	RIVERS\intelp]
E Process > Module	12	0.18	C:\WINDOWS\system32\h	ւցլ. վլլ 🗧
H TileProcess [PTD 0] (6635 ticks/86 10%)	6	0.09	C:\WINDOWS\system32\n	toskrnl.exe
E Clock eve [PTD 496] (651 ticks Select defaul	t counter		+	RIVERS USBPOR
t jevnlore eve [PID 2864] (110 t Bucket			•	XIVERS (eluubs)
E wampione ene [PID 1420] (68 ticks - u				RIVERS\tenin :
Swetan [PTD 4] (63 tisler(0.82%)	agement			STVERS\ushhub
E System [IID 4] (05 ticks)0.02%	c: 2			RIVERS\hidusb.
Merge this pr	ofile with	another		RIVERS\HIDCLAS
Explorer EAE [IID 1040] (25 th				
Javaw.exe [rij 2260] (10 ticks mark for comp	arison n. (moote.(a)			
1 iexplore.exe [FID 4056] (If the Compare with	E:/TUULS/Co)deAnalyzer/j	ppc/sort.etm/./sort	
the compare this compare this true to the the this the	§rofile wit	h another		
+ [] cmd.exe [PID 728] (12 ticks/0.				
🕀 💽 NLNOTES EXE [PID 940] (11 tick 👆 Populate Code	Miner Datab	ase		
🛨 💽 ntaskldr. EXE [PID 2416] (8 tich.,				
🕀 💽 svchost.exe [PID 1032] (6 ticks/0.08%)				
🗄 💽 eclipse.exe [PID 2252] (4 ticks/0.05%)				
🕀 💽 iexplore.exe [PID 3068] (3 ticks/0.04%)				
🕀 💽 Rtvscan. exe [PID 1556] (2 ticks/0.03%) 🛛 🌄				×
n 🖸 D.1 IDTD 464] (2.4:-1(0.02W)	<			>
	Total: tick	ts (6635), t	ime share (86.19%) (1	Orows)

7. In the **Profile Comparison** view, the detailed information of modules you have selected to compare is listed as follows:

🚰 Pro	ofile C	Comparis	on 🗙							71	🖫 <	¢ ۳	
First Secon	Node: d Node:	E:/TOC : //9.1)LS/CodeAr .86. 13. 72/	nalyzer/pj /data/samj	pc/sort.etm-so ple profile/G	ort-sort 1 K/winxp_20	CID 113460 2051026. סי	39/sor 1t-IdlePi	t rocess				
%CP	V_1	%CPU_2	us/Tx_1	us/Tx_2	us/Tx_delta	%change	%total	%accum	Module	Symbol			
[58.	04] [86.19]	[0.58]	[0.86]	[0.28]	[48.50]				[Totals]			
		85.84		0.86	0.86		304.94	304.94	C:\\int	NoSymbols			
32	. 13		0.32		-0.32		-114.13	190.82	./sort	.xxxxsort			
25	. 91		0.26		-0.26		-92.06	98.76	./sort	. xxx show			
		0.16		0.00	0.00		0.55	99.31	C:\\hal	NoSymbols			
		0.08		0.00	0.00		0.28	99.59	C:\\nto	NoSymbols			
		0.04		0.00	0.00		0.14	99.72	C:\\USB	NoSymbols			
		0.01		0.00	0.00		0.05	99.77	C:\\e10	NoSymbols			
		0.01		0.00	0.00		0.05	99.82	С:\\НДД	NoSymbols			
		0.01		0.00	0.00		0.05	99.86	C:\\hid	NoSymbols			
		0.01		0.00	0.00		0.05	99.91	C:\\ial	NoSymbols			
		0.01		0.00	0.00		0.05	99.95	C:\\tep	NoSymbols			
		0.01		0.00	0.00		0.05	100.00	C:\\usb	NoSymbols			

5.1.5.2Understand the calculations

The comparison tool uses the normalization factors entered in the Profile Comparison Wizard to calculate the microseconds of CPU consumed per transaction (us/Tx). Since the us/Tx values are computed on a per transaction basis, they can be compared directly from profile to profile.

The us/Tx values are calculated as follows:

- Calculate percentage of total ticks in the specific symbol: CPU% = Ticks in the specific symbol / Total ticks in the profile run
- 2. Calculate transactions per busy second: ITR = Transaction rate / CPU utilization
- 3. Calculate total CPU microseconds per transaction: Total microseconds per transaction = 1,000,000 / ITR * Number of CPUs
- 4. Calculate average CPU microseconds per transaction in the specific symbol: us/Tx = Total microseconds per transaction * CPU%

5.1.5.3Save a profile comparison

- 1. Click the **Save Comparison** button in the Comparison view toolbar. Alternatively, right-click anywhere in the view and select **Save Comparison** from the pop-up menu.
- 2. The **Save As** dialog opens. Browse to the desired directory and enter a file name.
- 3. Click **Save**. The comparison will be saved as a Profile Analyzer comparison (.etc) file. This file contains both compared profiles (zipped) and the normalization factors used in the comparison.

5.1.5.40pen a profile comparison

- 1. In the Navigator view, double-click the Profile Analyzer comparison (.etc) file.
- 2. Both compared profiles will open automatically in the Profile Analyzer editor as temporary files, and the Comparison view populated.

5.1.6Profile Merge

If user has profiled a benchmark multiple times using TPROF, he can merge the .out files for these runs using the Merge Wizard in Profile Analyzer. This can be useful for several types of situations:

- If user is going to measure a short-run application (or a short-run phase such as startup of a JVM during a benchmark), each individual profile may have too few ticks to draw meaningful results, but a with a merged profile patterns may begin to emerge
- If user is going to measure different CPU events (ticks, data cache misses, branch mispredictions) on different runs of a benchmark, he can merge these runs and see the data for all counters in a single profile
- If user wants to compare two runs, he can use profile merging to see which processes, modules, symbols, and symbol offsets were active in both runs.

To merge several profiles, at least one of them must be opened. Then follow these steps:

1. Right-click in the process hierarchy view and choose Merge this profile with another.

2. In the **Merge Profiles Wizard**, select one or more profiles from the current project and click **Add** >, or use the **Browse** button to open a file dialog to select profiles from other locations. Note that Profile Analyzer will only let user to add profiles whose platform matches the profile already added to the list. Click **Next**.

3. On the **Select processor type from platform family** page, select a CPU type (this option is only available on platforms where different CPU types support different counters). Click **Next**.

4. On the **Counter options** page, user can select each profile in turn and chooses what counter to attribute its events to. One of the profiles must be the "primary" profile; this is the profile used to merge other profiles into. Click **Next**.

5. Select a file name and click Finish.

At any time the **Finish** button is not grayed out user can click it to merge immediately.

When user merges profiles, Profile Analyzer creates a new file with a .etm extension (ETM=e-Tune Merged). This file is in a Profile Analyzer-supported XML format. Profile Analyzer also opens the file immediately after the merge. This profile looks much like ordinary TPROF profiles when viewed inside Profile Analyzer, with three differences:

- Processes, modules, symbols, and offsets that had data from more than one source profile are colored in green
- Multiple counter columns may appear in the Offsets view as well as ticks, if user chooses different counters for each source profile.
- No threads data is available, as it does not make sense to merge thread data from separate runs.

5.1.7Symbol Analysis

The following are tasks that you can perform to analyze symbols within Profile Analyzer:

5.1.7.1Code Miner support

The Code Miner support in Visual Performance Analyzer enables you to populate an SQL database with information from Profile Analyzer profiles, and then perform SQL queries on the database to detect performance patterns that are not easily detected by traditional profilers. The database tables allow you to associate profile counter information, symbols, and disassembly instructions so that you can find inefficient or highly active patterns of instructions, instruction sequences, symbols, register usage, and so on. For example, a Code Miner query can be used to find the hottest pairs of sequential instructions, or all symbols that contain a particular instruction sequence, or all symbols that are hotter than a certain threshold that have a certain pattern in their name. Code Miner is ideal for analyzing flat profiles, the objective is to find patterns of disassembly code, or usage patterns of certain types of symbols, that are inefficient. Without Code Miner it is extremely difficult to determine which patterns are worth investigating. Because Code Miner lets you determine the overall cost of a particular pattern within an entire module or an entire profile, you can use it to detect the patterns that will yield the maximum benefit when optimized, replaced, or eliminated.

Code Miner user interface support within Profile Analyzer includes a **Code Miner wizard** for populating data from a profile, and two views: a **Code Miner Query** view, that lets you query the Code Miner tables for a particular profile to find patterns of interest, and that displays the results in a sortable column-based table; and a **Query Tree** view that saves queries and database configurations so that you can easily locate, edit, re-run past queries or queries imported from another user, such as Compiler listings.

5.1.7.1.1Populate Code Miner Database

In Profile Analyzer, you can choose to keep profile or trace file data into DB2 database. This can be realized via **Populate Code Miner Database Wizard**. Every time when you open **Populate Code Miner Database Wizard**, you can choose to create new table, append to existing table or clean tables. If you decide to keep data into database, the prefix of table which is designed to store data should be defined at first. In the next page of wizard, given name, host, port and admin password, Profile Analyzer can get access to the database and populate data automatically. Please follow the steps below to populate profile file into new table in existing database:

- 1. Open a profile file and right click.
- 2. Choose Populate CodeMiner Database.
- 3. Defines table prefix, create new tables on database and include proper fields as you need.

🚳 Populate Codeliner Database Wizard	
Populate Codeminer Database - Profile settings Select the data to be populated into the database.	
Path to the listing file:	
OR Path to the listing directory:	
Table prefix (XXXX, where XXXX_DISASSM is a resulting table) Populate listing file Populate profile data Create OPCODE/OPERAND fields in disassembly Create instruction type fields in disassembly Create new tables in database Append to existing database Log detailed sql messages	
< Back Next > Finish Ca	incel

4. Input db2 connection information into next wizard page. Be sure to pass firewall so as to connect database.

Populate CodeLiner Data	atabase Wiza	rd		X
Populate Codeminer Date Enter the database connect	tabase - DB i tion informati	nformation on.		
Database Name:				
VPA				
Username:				
DB2inst2				
Password:				

DB2 Host:				
9.186.10.109				
DB2 Port:				
50001				
	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

5. Click Finish

5.1.7.1.2Code Miner Database Queries

To check data in database, run query in Code Miner Queries. Please follow the steps below:

- 1. Create a new connection to the db2 database above through Code Miner Queries or Query Tree view
- Press Edit button in Code Miner Queries view
- Right-click and choose Define Database Connection



2. Input sort in Prefix box and press to list fields and tables with input prefix

Temporar Tror.	iling Query Tree	Inlined Calls 🔁	CodeMiner Que	aries 🗙 🎇	, -8
db2inst2@9.1	86.10.109:50001/	/VPA Edit	Prefix:	SORT	•
					<u>^</u>
1					<u> </u>
0 🗗 🛛	3 🗟 💊 😭	🖉 🙀 Title:		Group: no gr	oup selected
VPA:SORT					
VPA:SORT 🔀 Prefix	Table	Field name		Туре	Leng
VPA:SORT 🔀 Prefix SORT	Table DISASSM	Field name SID		Type BIGINT	Leng A
VPA:SORT 🔀 Prefix SORT SORT	Table DISASSM DISASSM	Field name SID ADDRESS		Type BIGINT BIGINT	Leng A 8 8
VPA:SORT X Prefix SORT SORT SORT	Table DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET		Type BIGINT BIGINT BIGINT	Leng 🔨 8 8 8
VPA:SORT X Prefix SORT SORT SORT SORT	Table DISASSM DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET BYTES		Type BIGINT BIGINT BIGINT VARCHAR	Leng 🔨 8 8 8 255
VPA:SORT X Prefix SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET BYTES DISASSEMBL	ř	Type BIGINT BIGINT BIGINT VARCHAR VARCHAR	Leng A 8 8 8 255 1024
VPA:SORT X Prefix SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASSM DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET BYTES DISASSEMBL' SEQ	ř	Type BIGINT BIGINT BIGINT VARCHAR VARCHAR INTEGER	Leng A 8 8 255 1024 4
VPA:SORT X Prefix SORT SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASSM DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET BYTES DISASSEMBL SEQ OPCODE	Ÿ	Type BIGINT BIGINT BIGINT VARCHAR VARCHAR INTEGER VARCHAR	Leng A 8 8 255 1024 4 255
VPA:SORT 🔀 Prefix SORT SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASSM DISASSM DISASSM DISASSM DISASSM	Field name SID ADDRESS OFFSET BYTES DISASSEMBL SEQ OPCODE OPERANDS	Y	Type BIGINT BIGINT BIGINT VARCHAR VARCHAR VARCHAR VARCHAR	Leng 8 8 255 1024 4 255 1024

3. Choose lines and right-click to add select statement to query

Temporar Troffing	Query Tree Inli	ined Calls 🔁 CodeMiner Que	eries 🗙 🎇	
db2inst2@9.186.10). 109:50001//VPA	Edit Prefix:	SORT	•
				~
				~
		S24 T: 11 - 1	Group : no mous	ralacted
			oroup. no group	Serected
VPA:SORT				
VPA:SORT 🛛 Prefix	Table	Field name	Type	Leng
VFA:SORT 🔀 Prefix SORT	Table DISASSM	Field name SID	Type BIGINT	Leng 🔨
VPA:SORT 🔀 Prefix SORT SORT	Table DISASSM DISASSM	Field name SID ADDRESS	Type BIGINT BIGINT	Leng 🔨 8 8
VPA:SORT X Prefix SORT SORT SORT	Table DISASSM DISASSM DISASSI AJJ	Field name SID ADDRESS	Type BIGINT BIGINT IGINT	Leng ^ 8 8 8
VPA:SORT & Prefix SORT SORT SORT SORT	Table DISASSM DISASSM DISASSI Add DISASSI	Field name SID ADDRESS s lect statement to query	Type BIGINT BIGINT IGINT ARCHAR	Leng 🔨 8 8 8 255
VPA:SORT & Prefix SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASS DISASS DISASS Copy	Field name SID ADDRESS sylect statement to query	Type BIGINT BIGINT IGINT ARCHAR ARCHAR	Leng 🔨 8 📕 8 255 1024
VPA:SORT 🔀 Prefix SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASSI DISASSI DISASSI DISASSI Sele	Field name SID ADDRESS sylect statement to query ct all	Type BIGINT BIGINT IGINT ARCHAR ARCHAR NTEGER	Leng 🔨 8 📕 8 255 1024 4
VPA:SORT 🔀 Prefix SORT SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASS DISASS DISASS DISASS DISASSM	Field name SID ADDRESS slect statement to query ct all OPCODE	Type BIGINT BIGINT IGINT ARCHAR ARCHAR NTEGER VARCHAR	Leng 8 8 255 1024 4 255
VPA:SORT 🔀 Prefix SORT SORT SORT SORT SORT SORT SORT SORT	Table DISASSM DISASSM DISASS DISASS DISASS DISASSM DISASSM	Field name SID ADDRESS slect statement to query ct all OPCODE OPERANDS	Type BIGINT BIGINT IGINT ARCHAR ARCHAR NTEGER VARCHAR VARCHAR	Leng 8 8 255 1024 4 255 1024

4. Press to run this statement in database

Ten	mporal	Profiling	Query Tree	Inlir	ned Calls 🔁 C	odeMiner Que	ries 🗙 🎽	°, – –
d	lb2inst	209.186.10	. 109:50001//	VPA	Edit	Prefix:	SORT	•
SI	ELECT FROM	dis.offset ~disassm (lis					
Run query								
	VPA: SU	RT 23				,		
	Pref:	ix	Table		Field name		Type	Leng 🔨
	SORT		DISASSM		SID		BIGINT	8
	SORT		DISASSM		ADDRESS		BIGINT	8
	SORT		DISASSM		OFFSET		BIGINT	8
	SORT		DISASSM		BYTES		VARCHAR	255
	SORT SORT		DISASSM DISASSM		BYTES DISASSEMBLY		VARCHAR VARCHAR	255 1024
	SORT SORT SORT		DISASSM DISASSM DISASSM		BYTES DISASSEMBLY SEQ		VARCHAR VARCHAR INTEGER	255 1024 4
	SORT SORT SORT SORT		DISASSM DISASSM DISASSM DISASSM		BYTES DISASSEMBLY SEQ OPCODE		VARCHAR VARCHAR INTEGER VARCHAR	255 1024 4 255
	SORT SORT SORT SORT SORT		DISASSM DISASSM DISASSM DISASSM DISASSM		BYTES DISASSEMBLY SEQ OPCODE OPERANDS		VARCHAR VARCHAR INTEGER VARCHAR VARCHAR	255 1024 4 255 1024

5. Double-click instruction item in new pop-up tab to view the profile this instruction belongs to



5.1.7.2Couple with Code Analyzer

Profile Analyzer can integrate with Code Analyzer for better navigation and comparison of module information between profiling file and executable file. This function can be initiated in generic hierarchy view. Then you can scroll both kind of information in Disassembly/Offsets view of Profile Analyzer and Instruction Table view of Code Analyzer at the same time. To couple with Code Analyzer, be sure to have both profile and binary file containing at least one same module. Please follow the steps below:

• Open a profile file.
Profile Analyzer - E:/TOOLS,	/ppc/sort.etm - Eclipse SDK			
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject	<u>Run W</u> indow <u>H</u> elp			
🖬 • 🔛 🖻 🖹 🐠 🕸 🥸	🔅 💀 🖧 🔐 🗛 • 🔗 🎨 🗇 •	-> -	🖹 🚹 Profile J	An »
Profiling Reso 🛛 🔭 🗖	E:/TOOLS/ppc/sort.etm 🗙			
LOCALHOST (Local)	 ■ F:/TOOLS/ppc/sort.etm (5329.0 ticks ■ F^O Process > Thread > Module ■ F^O Process > Module ■ F^O Modules 	ticks % 2207 41.41 1712 32.13 1381 25.91 22 0.41 2 0.04 1 0.02 1 0.02 1 0.02 1 0.02 1 0.02 1 0.02 1 0.02	Module Symbol/Funct /unix h_cede_end_po ./sort .xxxxsort /unix h_confer_end /unix .waitproc /unix .util_upd_cuu /unix .low /unix .low /unix .low /unix .unlock_enabl	ions A oint I point rr_tb_F le_mem
Samples Distri ☆ ♥ □	< · · · >	<pre>Modules (10 rows)</pre>		>
	🔚 Disassembly/Offsets 🛛 Source Code Com	npiler Listing Profile	Details »3	
58.04% 41.96% root: Modules - Total ticks: 5329, tir				

• Navigate generic hierarchy view, click a module and view its disassembly and offset information by pressing ENTER button.

Profile Analyzer - E:/TOOLS,	/ppc/sort.etm -	Eclipse SDK			
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject ;	<u>Run W</u> indow <u>H</u> elp				
📬 • 🔛 🗁 🔄 🗎 🖉 🚳 🖏 🥸	s 💀 🖧 📔	💁 -] 🛷 🎨 🤇	÷ + +> +	🖹 🚹 Profi	le An »
Profiling Reso 🛛 😕 🗖	E:/TOOLS/ppc/so	rt.etm 🗙			
LOCALHOST (Local)	Reference in the image of	c/sort.etm (5329.0 + > Thread > Module > Module t (3093 ticks/58.04% (2236 ticks/41.96%)	ticks 11712 55 3 1712 55 3 1381 44	Symbol/Functions Symbol/Functions .xxxsort .xxxshow	
Samplas Distri 😚 🎽 🗖	<		X (2002)		
			10tal. ticks (3093),	time snare (30.04%)	(2 rows)
	🛗 Disassembly/Off	isets 🛛 Source Co	de Compiler Listing Profile	e Details 73	
	Offsets for: .xxxx	sort (./sort)		🚔 📥	눈 🏓 🏹
.xxxxsort .xxxshow	Offset	Bytes	Disassembly		*
	0x00000000	9421ffb0	stwu r1,-80(r1)		
	0x0000004	90610068	stw r3,+104(r1)		
	0x0000008	38600000	li r3,+0		
	0x000000c	90610044	stw r3,+68(r1)		
	0x00000010	80610044	lwz r3,+68(r1)		
	0x00000014	2c03000a	cmpi cr0,0,r3,+10		
	🚴 0x0000018	4080012c	be BO_IF_NOT,CRO_LT,Ox:	L44	
55.35% 44.65%	0x000001c	38600000	li r3,+0		
and all and a Tabal Airbay 2002 at	0x00000020	90610040	stw r3,+64(r1)		
module: All Symbol under ./sort	<	00610040	1		>
	p				

• Right-click this module symbol in generic hierarchy view. In popup menu, choose " Open in CodeAnalyzer".

Profile Analyzer - E:/TOOLS,	/ppc/sort.etm -	Eclipse SDK				
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject	<u>R</u> un <u>W</u> indow <u>H</u> elp					
🖬 • 🔛 🖻 🔯 🗢 🕸 🥸 🥸	s 💀 🖧 📔	Q •] 🔗] 🍤 🤅	• + -> +		🖹 射 Prof	ile An »
Profiling Reso 🛛 😕 🗖	E:/TOOLS/ppc/so	rt.etm 🗙				
LOCALHOST (Local)	Process Freess Frees	z/sort.etm (5329.0 t: > Thread > Module > Module Select default count Bucket Hierarchy Management Compare this profile Populate CodeMiner D Open in CodeAnalyzer	icks tick 1712 1381 .er with another latabase	ss % 55.35 44.65	Symbol/Functions .xxxxsort .xxxshow	
Samples Distri 🕱 🍡 🗖 🗖	<		🔊 Total: ti	cks (3093), t	ime share (58.04%)	(2 rows)
III ● ▽	Disassembly/Off	sets 🛛 Source Cod	le Compiler Li	sting Profile	Details ^{>>} 3	
	Offsets for: .xxxxs	sort (./sort)			ē 🔿	- 🖕 💙
.xxxxsort .xxxshow	Offset	Bytes	Disassembly			¥ 🔼
	0x0000000	9421ffb0	stwu r1,-80	(rl)		
	0x0000004	30610068	stw r3,+104)	(r1)		
	0x0000000c	90610044	stw r3 +68/r	-1.1		
	0×00000010	80610044	1wz r3.+68()	-1)		
	0x00000014	2c03000a	cmpi cr0.0.1	:3,+10		
	3 0x0000018	4080012c	be BO IF NOT	CRO LT,Ox1	4	
55.35% 44.65%	0x000001c	38600000	li r3,+0			
	0x00000020	90610040	stw r3,+64(1	:1)		
module: ./sort = lotal ticks: 3093, t:	0-00000024	00610040	1 1644-			× *
inourie. All Symbol under ./sort						

• Choose the corresponding binary file of this module.

😂 Profile An	alyzer - E:/	TOOLS	/ppc/sort.et	n - Eclipse Sl	DK			
<u>F</u> ile <u>E</u> dit <u>N</u> avi	gate Se <u>a</u> rch <u>P</u> r	oject ;	<u>Run W</u> indow <u>H</u> el	P				
Select execu	table for m	odule	./sort (3093	ticks/58.04%) to ? 🔀		🖹 🚹 Prof:	ile An »
查找范围(<u>I</u>):	🚞 ppc			💌 🔶 🖻				
				_		κs %	Symbol/Functions	
2	Ca. dll					55.35	.xxxxsort	
我最近的文档	sort					44.03	. XXXSNOW	
	wort, etm							
	_							
桌面								
3%891%48								
我的电脑								
网上邻居						(2000)		
						icks (3093), t	ime share (58.04%)	(2 rows)
	where the for some					sting Profile	Details "S	
	文件名(20):	sort		<u>•</u>	1 11#0		🚊 🌩	👈 🔿 🏹
	文件类型(I):	All Fi	les (*.*)	•	取消			*
			0x0000000	94211180	SCWU 11,-80	(r1)		
			0x0000004	90610068	stw r3,+104	(r1)		
			0x0000008	38600000	li r3,+0			
			0x0000000	90610044	stw r3,+68((r1)		
			0x00000010	80610044	1wz r3,+68((T1)		
		$\overline{}$		4080012c	bc BO TR NO	13,710 IT CRO LT 0v1	44	
55.35%	44.65%	~	0x0000001c	38600000	li r3,+0			
and the former of the second s	- T.I.I.I.I 7	002	0x0000020	90610040	stw r3,+64((r1)		
module: /sort	- lotal ticks: 3 bol under /sort	1093, t:	. 0-0000024	00610040	1 1.642			
module. All Sym	bor under ./Sor(·						
1								

• Later, Code Analyzer Perspective opens automatically with this binary file. To scroll Profile Analyzer view with Code Analyzer view at the same time, be sure to open Disassembly/Offsets view. Now, when you select an table row in Disassembly/Offsets view, the instructions of this address will be highlighted in Instructions Table accordingly.

<pre>GodeAnalyzer - E:\TOOLS\ppc\sort - Eclipse Platform</pre>			
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp			
11 • 22 ≙ 10 • 1 • 10 • 10 • 10 • 10 • 10 • 10] 🗄 × 🕅 × 🖄	∫ Q _ • ∫ <i>A</i> ∕	😰 🔚 CodeAnalyzer 🚮 Profile An 🐉 Java
Instructions Table 🕱 🗖 🗖	Disassembly/Off	fsets 🗙	(■ □ □ () () ()
🔨 🗸 🕺 🍓 👘 i. i. i. ppr 🕅 🖂 🛄 🗸	Offsets for: .xxxx	sort (./sort)	
Rile: cont a Reputien: comparent Talan: 57 PB: addr:	Offset	Bytes	Disassembly 🔼
File. Sort. c Function XXXXSort Index. 51 bb. addr.	0x00000000	9421ffb0	stwu r180(r1)
Address Opcode Mnemonic Comment Freq. Graph 🔨	0x00000004	90610068	stw r3.+104(r1)
0x100 0x9061 stw r3	0x0000008	38600000	li r3.+0
0x100 0x3860 li r3,0 .bf 0	0x000000c	90610044	stw r3.+68(r1)
0x100 0x9061 stw r3	0x00000010	80610044	lwz r3,+68(r1)
0x100 0x8061 1wz r3	0x00000014	2c03000a	cmpi cr0,0,r3,+10
0x100 0x2c03 cmpi c 0	1 0x0000018	4080018c	be BO IF NOT, CRO L
0x100 0x4080 bge cr 0	0x0000001c	38600000	li r3,+0
· · · · · · · · · · · · · · · · · · ·	0x00000020	90610040	stw r3,+64(r1)
0x100 0x3860 li r3,0	0x00000024	80610040	lwz r3,+64(r1)
0x100 0x9061 stw r3	0x00000028	2c03000a	cmpi cr0,0,r3,+10
0x100 0x8061 lwz r3 0	3 0x0000002c	40800100	be BO IF NOT, CRO L
0x100 0x2c03 cmpi c 0	0x00000030	80610068	1wz r3,+104(r1)
0x100 0x4080 bge cr	0x00000034	80810040	lwz r4,+64(r1)
	0x0000038	5484103a	rlwinm r4,r4,2,0,2
0x100 0x8061 lwz r3 0	0x0000003c	7c63202e	1wzx r3,r3,r4
0x100 0x8081 1wz r4	0x00000040	80a10068	1wz r5,+104(r1)
0x100 0x5484 rlwinm 0	0x00000044	80c10040	lwz r6,+64(r1)
0x100 0x7c63 1wzx r	0x00000048	38c60001	addi r6,r6,+1
0x100 0x80a1 1wz r5 0	0x0000004c	54c6103a	rlwinm r6,r6,2,0,2
0x100 0x80c1 1wz r6	0x00000050	7ca5302e	lwzx r5,r5,r6
0x100 0x38c6 addi r	0x00000054	7c032800	cmp cr0,0,r3,r5
Ux1UU Ux54c6 rlwinm U	🚴 0x0000058	408100bc	be BO_IF_NOT,CRO_G
Ux1UU Ux7ca5 lwzx r U	0x000005c	80610068	1wz r3,+104(r1)
Ux1UU Ux7c03 cmp cr U	0x0000060	7c63202e	1wzx r3,r3,r4
Ux1UU Ux4U81 ble cr U	0×0000064	90610048	star r3 +72(r1) 🔡 🔤
	<		>
cmpi - Compare Immediate D-form			

You may also trigger highlighting from Instructions Table of Code Analyzer. Be sure to press **b**button in tool bar.

5.1.7.3View disassembly comparison

In this release, you can compare the offset, ticks and disassembly of two profiles in disassembly comparison view. You can open Disassembly Comparison view by choosing **Windows-> Show View -> Other... -> Profile Analyzer -> Disassembly Comparison** or just find it in the right bottom panes.

To view the disassembly comparison, you should follow these steps:

- 1. Compare two profiles through **Profile Comparison Wizard** or just open an .etc file.
- 2. Open Profile Comparison view by choosing Windows-> Show View -> Other... -> Profile Analyzer -> Profile Comparison

Disassembl	y/Offsets	Source (Code Java	/Hierarchy 📑	Profile	Comparis	ion 🗙	»²			
								👫 🗄	\$ €		
First Node: E:/TOOLS/ppc/20.24.1_misc_32.etm											
Second Node: E:/TOOLS/ppc/20.24.2_misc_64.etm											
%CPU_1	%CPU_2	us/Tx_1	us/Tx_2	us/Tx_delta	%change	%total	%accum	Module	Symb 🔥		
[56.81]	[93.61]	[0.57]	[0.94]	[0.37]	[64, 78]				[Tot:		
0.86		0.01		-0.01		-2.33	95.78	misc_32	.free		
8.08	8.88	0.08	0.09	0.01	10.01	2.20	97.97	//libc	NoSyn		
0.80		0.01		-0.01		-2.18	95.80	//libe	.free		
	0.78		0.01	0.01		2.11	97.91	//libc	pta		
0.74		0.01		-0.01		-2.02	95.89	misc_32	. doi 1		
0.69		0.01		-0.01		-1.87	94.02	misc_32	. doi 1		
	0.67		0.01	0.01		1.81	95.83	misc_64	.mall		
	0.61		0.01	0.01		1.66	97.49	//libe	.mall		
	0.61		0.01	0.01		1.66	99.15	misc_64	.free		
	0.56		0.01	0.01		1.51	100.66	//libc	. srai		
	0.56		0.01	0.01		1.51	102.16	misc_64	. doi t		
0.46	0.94	0.00	0.01	0.00	106.05	1.32	103.48	//libpt	NoSyn		
0.40		0.00		0.00		-1.09	102.39	misc_32	. mall 🤍		
	0.20		0.00	0.00		1.06	103 //5	mica 64	PW AT		
									>		

3. Right-click the line in Profile Comparison View and choose Show disassembly comparison

Di	isassembly	y/Offsets	Source C	ode J	Java,	/Hierarchy 📑	Profile	Comparis	on 🗙	»3		
										👫 🗄	\$. E	
F	First Node: E:/TOOLS/ppc/20.24.1_misc_32.etm											
S	Second Node: E:/TOOLS/ppc/20.24.2_misc_64.etm											
	%CPU_1	%CPU_2	us/Tx_1	us/T:	x_2	us/Tx_delta	%change	%total	%accum	Module	Symbo	
	[56.81]	[93.61]	[0.57]	[0.9	94]	[0.37]	[64.78]				[Tot:	
	0.86		0.01			-0.01		-2.33	95.78	misc_32	.free	
	8.08	8.88	0.08	0	00	0.01	10.01	0.00	97.97	//libe	NoSyn	
	0.80		0.01		93	New Compariso	n		95.80	//libc	.free	
		0.78		0					97.91	//libc	pta	
	0.74		0.01		L.	Save Comparis	on		95.89	misc_32	. doit	
	0.69		0.01						94.02	misc_32	. doi 1	
		0.67		0		Show disassem	bly compa	arison	95.83	misc_64	.mall	
		0.61		0			-		97.49	//libc	.mall	
		0.61		0	\$ 25	Expand all	,		99.15	misc_64	.fre∈	
		0.56		0		r			100.66	//libc	. srai	
		0.56		0	5	Collapse all			102.16	misc_64	. doi 1	
	0.46	0.94	0.00	0	—				103.48	//libpt	NoSyn	
	0.40		0.00			Сору			102.39	misc_32	. mall 🗸	
	<			-		Select All					>	

4. The corresponding disassembly comparison view opens.

Disasse	mbly/Offsets	Java/H:	ierarchy Profile Comparison 🏨 Disassem	blyCompar 🗙 🍡 🖳 🗖
Symbol:	.pthread_se	lf		
Line	Offset	Ticks	E:/TOOLS/ppc/20.24.1_misc_32.etm	E:/T00LS/ppc/20.24.
[00000000	11	rlwinm r3,r0,2,0,29	rlwinm r3,r0,2,0,2:
	00000004	4	lwzx r29,r30,r3	1wzx r29,r30,r3
	00000008		bl 0xd0134294	bl 0xd0134294
	0000000c		lwz r2,+20(r1)	lwz r2,+20(r1)
	00000010		lwz r0,+28(r31)	lwz r0,+28(r31)
	00000014	2	andi. r0,r0,0x140	andi. r0,r0,0x140
	00000018		bc B0_IF,CR0_EQ,0xd011b6f4	bc BO_IF,CRO_EQ,Ox
	0000001c		ori r3,r31,0x0	ori r3,r31,0x0
	00000020	1	bl 0xd012028c	bl 0xd012028c
	00000024		ori r0,r0,0x0	ori r0,r0,0x0
	00000028		lwz r0,+32(r31)	lwz r0,+32(r31)
	0000002c		andi. r0,r0,0x80	andi. r0,r0,0x80 🛛 💻 💻
	00000030	1	bc B0_IF_NOT,CR0_E0,0xd011b7d4	bc BO_IF_NOT,CRO_E
	00000034		addi r4,r1,+64	addi r4,r1,+64
	00000038		li r3,+0	li r3,+0
	0000003c		bl 0xd010fa30	bl 0xd010fa30 🛛 🕙
<				>

There are two rows of hottest bars referring to two profiles which are compared. You can navigate disassembly comparison view by clicking them as follows:

Disasse	mbly/Offsets	Java/H	ierarchy Profile Comparison 🌇 Dis	sassemi	blyCompar 🗙 🎽		
Symbol:	.pthread_se	lf					
Line	Offset	Ticks	E:/TOOLS/ppc/20.24.1_misc_32.	.etm	E:/T00LS/ppc/20.24.		
	00000000	11	rlwinm r3,r0,2,0,29		rlwinm r3,r0,2,0,2		
	00000004	4	1wzx r29,r30,r3		1wzx r29,r30,r3		
	00000008		bl 0xd0134294)1 0xd0134294			
	000000c		lwz r2,+20(r1)		lwz r2,+20(r1)	- (M)	
	00000010		lwz r0,+28(r31)	z r0,+28(r31)			
	00000014	2	andi. r0,r0,0x140	di. r0,r0,0x140			
	00000018		bc B0_IF,CR0_EQ,0xd011b6f4		be BO_IF,CRO_EQ,Ox	-	
	0000001c		ori r3,r31,0x0		ori r3,r31,0x0		
	00000020	1	bl 0xd012028c	ľ	bl 0xd012028c		
	00000024		ori r0,r0,0x0		ori r0,r0,0x0		
	00000028		lwz r0,+32(r31)		lwz r0,+32(r31)		
	0000002c		andi. r0,r0,0x80		andi. r0,r0,0x80		
	00000030	1	be B0_IF_NOT,CR0_EQ,Oxd011b7	d4	be BO_IF_NOT,CRO_E		
	00000034		addi r4,r1,+64		addi r4,r1,+64		
	00000038		li r3,+0		li r3,+0	_	
	0000003c		bl 0xd010fa30		bl 0xd010fa30	×	
<					>		

To right-click and select menu item, you can sort columns by source line number or offset.

Disasser	mbly/Offsets	Java/H	ierarchy Profile Comparison ኩ Disassen	nblyCompar 🗙 🍡 🗖 🗖
Symbol:	.pthread_se	lf		
Line	Offset	Ticks	E:/TOOLS/ppc/20.24.1_misc_32.etm	E:/T00LS/ppc/20.24.
	00000000	11	rlwinm r3,r0,2,0,29	rlwinm r3,r0,2,0,2:
	00000004	4	lwzx r29,r30,r3	1wzx r29,r30,r3
	00000008		bl 0xd0134294	bl 0xd0134294
	0000000c		lwz r2,+20(r1)	1wz r2,+20(r1)
	00000010		lwz r0,+28(r31)	1wz r0,+28(r31)
	00000014	2	The Order disessembly by source line p	umber , r0,0x140
	00000018		the disassembly by source line in	,CRO_EQ,Ox
	0000001c		💭 Örder disassembly by offset	31,0x0
	00000020	1		2028c
	00000024		ё Сору	0,0x0
	00000028		Select All	32(r31)
	0000002c		Show Countors	,r0,0x80
	00000030	1		NOT,CRO_E
	00000034		addi r4,r1,+64	addi r4,r1,+64
	00000038		li r3,+0	li r3,+0
	0000003c		bl 0xd010fa30	bl 0xd010fa30 🛛 🞽
<				>

5.1.7.4View offsets and disassembly

Profile Analyzer can disassemble the instruction stream for any symbol for which such a stream is available. Disassemble support is available for the following platforms:

- Intel IA32
- AMD-64 or EM64T (same instruction set)
- PowerPC
- zSeries
- CELL/B.E. (both PPE and SPE)

Whether the profile contains an instruction stream is dependent on the profiling tools used to create it.

For JITCODE (JIT-compiled Java methods), instruction streams are available if the JPROF library was loaded with the JVM (using the -Xrunjprof option), the jints sub-option was specified as part of this option, and the log-jita2n* files produced were available at the time that mergetprof was run. Only the IBM Virtual Machine for Java supports the JPROF library.

When disassembly can be generated for a symbol, Profile Analyzer displays a table containing instruction addresses, the bytes for each instruction, the instruction sequence, and tick information. The following view shows the disassembly for a Java method on an Intel IA32 system:

Code Distribution	Comparison View (lass Hierarchy f	Listing Information	🔛 OffsetAsm Infor	ma 🗙 🎽	ı □ □
Offsets for: Hello	.foo(I)V(Process	java. exe d1c)			8 5	▶ 🛍 🔤 🖓
Offset	Bytes	Disassembly		Remarks	\$	Ticks
0x10b94e46	33db	XOR EBX, EBX				
0x10b94dfa	85ff	TEST EDI,EDI			9.36	31
0x10b94ele	85db	TEST EBX,EBX				
0x10b94e0b	85c0	TEST BAX, BAX			0.30	1 L
0x10b94e7d	83ec08	SUB ESP,8H				
0x10b94dd0	83ec08	SUB ESP,8H				
0x10b94df8	2bfa	SUB EDI,EDX			24.7	82
0x10b94ded	cleblf	SHR EBX,31			8.45	28
0x10b94de8	clfa06	SAR EDX,6			14.8	45
0x10b94e02	c20400	RET 4H			1.51	5
0x10b94ddc	53	PUSH EBX				
0x10b94e4e	50	PUSH BAX				
0x10b94e39	50	PUSH EAX				
0x10b94e4f	6a2e	PUSH 2eH				
0x10b94e3a	6a2e	PUSH 2eH				
0x10b94dfe	5Ъ	POP EBX			2.71	2
0x10b94e57	8b12	MOV EDX,DWORD	PTR [EDX]			
0x10b94e51	8b15d07a5d34	MOV EDX,DWORD	PTR [345d7ad0H]			
0x10b94ddd	8b7c2410	MOV EDI,DWORD	PTR [ESP+10H]	N	0.60	2
0x10b94e70	bf08000000	MOV EDI,8H		13		
0x10b94e85	bf6014d800	MOV EDI,0d814	60H			
0x10b94deb	8bda	MOV EBX, EDX			9.06	30
0x10b94e3c	8b18	MOV EBX,DWORD	PTR [EAX]			
0x10b94e22	8b1dd07a5d34	MOV EBX,DWORD	PTR [345d7ad0H]			
0x10b94e0f	bb01000000	MOV EBX,1H				
0x10b94e4a	8b442408	MOV BAX, DWORD	PTR [ESP+8H]			
0x10b94e36	8b4328	MOV BAX, DWORD	PTR [EBX+28H]			
0x10b94e28	8b03	MOV BAX, DWORD	PTR [EBX]		0.30	L
0x10b94e05	8b0590eleb00	MOV BAX, DWORD	PTR [Oebel90H]			~
<						

If no disassembly is available, Profile Analyzer displays an Offsets view containing ticks for each offset. The following view shows the offsets for the NTOSKRNL.EXE module of the same profile; this module has a single symbol referred as NoSymbols for that symbol data (and by extension, instruction stream of a symbol) could not be obtained currently:

Code Distribution	Comparison	View Curr	ent selection	Listing Informatio	n 🔝 OffsetAsm	Informa 🖂	»d		
Offsets for: NoSym	nbols(all pr	ocesses)				۵ 😓	0 m	$\left \cdot\right $	
Offset	*	Ticks						^	
8040b35d	0.34	1		N					
8040b6e1	0.34	1						_	
8040cf80	0.34	1							
8040cf98	0.34	1							
80411848	0.34	1							
80412d29	0.34	1							_
8041434c	1.02	3							
80414392	0.68	2							
804143d3	0.68	2							
804144c2	0.34	1							
8041475f	0.34	1							-
804147cf	1.02	3							_
80415fel	0.34	1							
8041a610	0.34	1							
8041c63c	0.34	1							-
8041ca7a	0.34	1							
8041d1a2	0.34	1							
8041d2fc	0.34	1							
8041d321	0.34	1						×	
Distance in the second se									

If you are expecting to see disassembly data for a symbol and instead see only offset data, check the following:

There should be a tprof.micro section (for static-compiled methods) or a log-jita2n section (for JIT-compiled methods) in the profile you have loaded.

- The appropriate section should contain instruction data. In the tprof.micro section, the symbol must have a sequence of lines beginning with C:; if no such lines exist in the tprof.micro section, the -off option may not have been specified in the POST options (if you were manually profiling). In the JITA2N section, after each symbol there should be a sequence of binary bytes or hex data.
- You may be able to view the disassembly by switching to the Disassembly view. Click on the pulldown menu at the top right of the view and ensure that the **Show disassembly** item is checked.

5.1.7.4.1Navigating the Disassembly/Offsets View

You can quickly navigate to areas of high activity in this view using either the **Hotness bar** or a combination of sort and selection actions:

• Navigate to hot areas by sorting and selecting

You can click on any column in the offsetAsm Information view to sort by that column. Repeated clicks on the same column reverse the previous sorting order. To navigate to hot areas in a symbol you can follow these steps:

- 1. Click on a column heading that relates to CPU activity, to sort the view by that column (e.g. Ticks, %CPU activity, or a CPU counter if the profile contains CPU counter counts). The lines with the most events are sorted to the top.
- 2. Select a line of interest; the top line should be the one with the most events in the column you selected.
- 3. Click on the **Offset** column heading to sort by offsets again. The busy line you had selected in the previous step remains selected and remains in the viewable area.

If your platform supports symbol call resolution (currently only the x86 and x86-64 platforms, as these are the only platforms in which direct relative or absolute branch instructions are used to make calls to other symbols), you can also quickly find calls to resolved targets by sorting by the **Remarks** column. The following shows disassembly for a JIT-compiled Java method, sorted by the Remarks column so that lines containing call targets are displayed at the top:

Offset	Bytes	Disassembly	Remarks
수 0x0256DED0	E8E9DOF5FF	CALL 24CAFBEH	java/lang/String.equals(Ljava/lang/Object;)Z_24cafb4
수 0x0256E920	E899C6F5FF	CALL 24CAFBEH	java/lang/String.equals(Ljava/lang/Object;)Z_24cafb4
수 0x0256F158	E861BEF5FF	CALL 24CAFBEH	java/lang/String.equals(Ljava/lang/Object;)Z_24cafb4
	E8FF5CF8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
40x0256DD41	E8B847F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 OxO256DDEA	E80F47F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
40x0256DFD8	E82145F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256E0D9	E82044F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256E7BB	E83E3DF8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256E85B	E89E3CF8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256EA2B	ESCE3AFSFF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256EB28	E8D139F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
Ox0256EFC8	E83135F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
40x0256F06A	E88F34F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256F280	E87932F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
수 0x0256F378	E88131F8FF	CALL 24F24FEH	java/lang/StringBuffer. <init>(Ljava/lang/String;)V_24</init>
	ESDOA9FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
	E8A6A9FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
수 0x0256DD60	E87894FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
40x0256DE02	E8D693FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
수 0x0256DFE8	E8F091FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
🔷 0x0256E0F0	E8E890FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
수 0x0256E7D3	E8058AFBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
수 0x0256E873	E86589FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
수 0x0256EA39	E89F87FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff
ሩ 0х0256ЕВЗА	E89E86FBFF	CALL 25271DDH	java/lang/StringBuffer.append(I)Ljava/lang/StringBuff

You can use the same three-step sorting technique as for offsets, to find calls to a particular symbol:

- 4. Sort by the **Remarks** column. You may need to select the column header twice, if no call targets are visible the first time you select the column.
- 5. Select the line containing a call target of interest.
- 6. Sort by the offsets column. The line you had previously selected is now in the viewable area and instructions are displayed in offset order.

You can double-click on a line containing a call target to switch the current symbol in the offsetAsm Information view to the target symbol.

Loop nest detection and branch target detection

When Profile Analyzer loads the disassembly for a symbol it analyzes internal direct branches in the disassembly to determine loop patterns. Any backward branch may be considered the end of a loop, provided certain other parameters are met. Any block of code detected to be within a loop is indented by one space; if multiple nested loops are detected, sections of code may appear more deeply indented. In some cases the level of indentation may be extreme, as in the following example:

Offset	Bytes	Disassembly	Remarks	
Ox0256E819	48895128	MOV QWORD P		
Ox0256E81D	4833D2	XOR RDX, RDX		
Ox0256E820	48895010	MOV QWORD P		
Ox0256E824	48895018	MOV QWORD P		
Ox0256E828	48895020	MOV QWORD P		
Ox0256E82C	4C8930	MOV QWORD I		
0x0256E82F	488BDO	MOV RDX, RAX		
Ox0256E832	48C1E20D	SHL RDX,13		
Ox0256E836	4881E20000	AND RDX,7FF		
Ox0256E83D	4881CAOE80	OR RDX,800EH		
0~02568844	48805008	MUAL UNUM		

Here the indentation shows at least 15 levels of nesting. While it is unlikely that a programmer would have written a loop nest 15 layers deep, this level of nesting may occur where a compiler has inlined calls that occur within loops, and the inlined calls themselves contain other nested loops or further inlined calls.

You can remove loop nest indenting by clicking on the Plicon at the top of the view. If the ricon is displayed, clicking on it will display loop nest indenting for a symbol whose disassembly was not indented.

Reordering columns

You can reorder the columns of the Disassembly/Offsets view to hide or show particular columns or change the order in which columns are displayed. This is one of the features of eclipse 3.1. You can reorder the column just by clicking on the column name and dragging it to the place where you want it to be. It shows as follows:

Class Hierarchy fo	r: JITCODE module. 👫	Disassembly/Offs	ets 🗙 Source Code Compil	er Listing		
Offsets for: Hello.	foo(I)V(Process 3356)			🖹 🖥	🔶 n (- ×
Offset	Disassembly	Disassembly	Bytes	\$	Ticks	
0x10B94DF0	ADD EDX,EBX		03045	3.32	11	
🚴 0x10B94DD6	JBE 10B94E70H		0F8694000000			
0x10B94DF8	SUB EDI,EDX		2BFA	24.7	82	
0x10B94E46	XOR EBX, EBX		33DB			
0x10B94DD3	CMP ESP, DWORD PTR	[EBP+18H]	3B6518	6.64	22	
0x10B94E39	PUSH BAX		50			
0x10B94E4E	PUSH BAX		50			
0x10B94DDC	PUSH EBX		53			
Ox10B94DFE	POP EBX		5B	2.71	9	
0x10B94DF2	IMUL EDX, EDX, 3E8H		69D2E8030000	9.97	33	
0x10B94E3A	PUSH 2EH		6A2E			
0x10B94E4F	PUSH 2EH		6A2E			
🚴 Ox10B94DFC	JE 10B94E05H		7407			~
<					>	J

Branch and call navigation

Any disassembly line that contains a branch to a known target within the current symbol, or a call to another symbol, is indicated by an arrow in the left margin.

denotes a forward branch, one whose target is a subsequent instruction.

 ${}^{\textcircled}$ denotes a backward branch, one whose target is a previous instruction.

denotes a call or branch to another profiled symbol. This is only available for x86 and x86-64 platforms.

When you double-click on a line containing one of these icons, the view changes to show the target of the branch (a different location in the current symbol, or the target symbol of a call).

To navigate back to the last in-symbol branch you selected, after you have followed that branch, press the to button on the offsetAsm Information view toolbar.

The hotness bar

By clicking on an area in the hotness bar, you will be taken to the corresponding disassembly instructions, or offsets. For lengthy disassembled methods, you may need to page up or down to find the hot area in question, as a line in the hotness bar that is one pixel high may relate to several pages of disassembly.

Alphaworks

When you select a line in the Disassembly/Offset Information view, a yellow square appears in the hotness bar to show the currently selected area of the symbol.

5.1.7.5View source code

When an executable or library has been compiled with line number information (for example the -g option on some compilers), the platform profiler, like Tprof on AIX, may be able to obtain line number information for profiled symbols in such an executable or library. You can then view source code for these symbols within Profile Analyzer.

Line number support is available when the TPROF post-processing command includes the -off option. This option is enabled by default when you use the run.tprof_e script or run the profiling session from the **Profiling Configurations** view.

When you first select a symbol for which line numbers are available from the Symbols view, a dialog is displayed to ask whether you want to view source code for the symbol:



If you choose **Yes**, a File dialog is displayed that lets you navigate to the path containing the file. The name of the file you choose from this dialog does not have to match the name in the TPROF output, but if the line numbers do not match those of the file from which the code was compiled (for example, if the file has been edited since it was compiled), the tick information may not map to the correct source line numbers.

If you choose **No**, you are not prompted to enter source for any other symbols in the current profile, but when you load a different profile containing line number information, you may again be prompted to locate source files. If you choose **Don't ask me again**, you will not be asked to open a source file until you exit and restart Profile Analyzer.

The following view shows source code for a symbol:

Sou	rce Code 🔀	2 %	10 🏓 🎽	
Source	for: Hello.foo(I)V(Process 3356)			
Line	Source	\$	Ticks	~
1				
2	import java.io.File;			
3	<pre>import java.io.RandomAccessFile;</pre>			=
4				
5	public class Hello			
6	(
7				1.1.1
8	private static boolean flag = false;			
9				
10	private final static int bar(int i)			
11	(
12	return i % 1000;			
13	}			
14				
15	public static void foo(int i)			
16	(
17	<pre>int foo = bar(i);</pre>	84.5	280	
18				~
1.10		10.0	10	

If you choose No or Don't ask me again, no source code will be shown. The source code view shows as follows:

🖻 Source Code 🛿 🔋 🖗 🎽 🖓 🏓 🍸 🖓 🗖
ource for: java/io/BufferedWriter.flushBuffer()V(Process 3356, none available)
The current symbol has source line number information
but no source file has been associated
Associate File

You can again associate source file by pressing the button in the center of the view, or click **Associate Source File** icon icon the toolbar of the view.

When you click on different areas of the hotness bar in the right side of the source code view, the corresponding line in the source file you select will be highlighted. You can see it in the following view:

sed Sour	ce Code 🗙	ê 🍫	10 🖻 🎙		
Source	for: Hello.foo(I)V(Process 3356)				
Line	Source	*	Ticks	~	
11	(
12	return i % 1000;				
13	}				
14					
15	public static void foo(int i)			_ ,	
16	(_ =	–र म
17	<pre>int foo = bar(i);</pre>	84.5	280		
18					
19	if (foo == 0)	12.0	40		
20	(_	
21	flag = !flag;	0.30	1	_	
22	if (flag)			_	
23	System.err.print('.');	0.30	1	_	
24	}			_	
25	}	2.71	9	_	
26				_	
27	public static void main(String[] argv)			_	
28	(~	

You can export these source codes to file by selecting Export to Files... in the menu of the toolbar as follows:

5Rd Sour	rce Code 🕄	🚔 🎭 👘	19 -	
Source	for: Hello.foo(I)V(Process 3356)	🚔 Сору		
Line	Source	Link.	5	~
11	(
12	return i % 1000;	🚔 Change <u>f</u> ont size/type		
13	}	🗊 <u>E</u> xport to Files		
14			<u> </u>	
15	public static void foo(int i)	Reset Default		_ =
16	(Annual A. Sumar Bill	- L	- = 💻
17	int foo = bar(i);	Associate Source File	P	
18				
19	if (foo == 0)	12.0	40	
20	{			
21	flag = !flag;	0.30	1	
22	if (flag)			
23	System.err.print('.');	0.30	1	_
24	}			_
25)	2.71	9	_
26				
27	public static void main(String[] arg	v)		_
28	(~

5.1.7.6View temporal profiling

When a Profile Analyzer profile contains a trace buffer section, Profile Analyzer attributes buffer events to appropriate symbol offsets, symbols, modules, threads, and processes. When you select a profile object (click on a process, thread, or module in the process tree, double-click on a symbol in the symbol list, or double-click on a disassembly or offset line with tick information in the offsetAsm Information view), Profile Analyzer can display a

temporal graph showing *when* during the profile run the ticks for that object occurred. This version of the Temporal profiling view is called **Tick intensity over time**. The following screen capture shows the Temporal Profiling view for a java.exe process in a profile that ran for about 7.5 seconds:



You can change the number of intervals of a temporal profile by sliding the intervals slider (on the left) to the left or right. This changes the number of intervals used to display the temporal graph. Changing the granularity of a 25-second trace run to 50 intervals will result in each bar showing the events for a particular half-second. For the same trace run, a granularity of 10 would have 2.5 seconds attributed to each bar. The following screen capture shows the same information as above, but with a granularity of 8 (that is, 8 equal-time intervals):



When the selected object is a Profile, Process, Thread, or Module, up to six of the "natural" children of that object are shown in a line graph superimposed on the bar graph, as can be seen in the above images. A child is only shown if it is a significant contributor to the parent tick count. You can hide the tick information for a child by

deselecting the check box beside its name below the bar graph. The following screen capture shows the same profile, with only the top two children selected:



The "natural child" of a profile object is as follows:

Parent	Natural children						
Profile	rocesses						
Process	 Modules, if "Ignore thread data" in the process tree checkbox is selected Threads, if "Ignore thread data" checkbox is <i>not</i> selected 						
Thread	Modules						
Module	Symbols						
Symbol	No children						
Offset tick	No children						

5.1.7.6.1 Changing the tick scale for children

In some profiles, it may be hard to distinguish the lines of the "natural children" of a particular profile object. You can slide the **Zoom children vertically** slide bar to exaggerate or diminish the scale for the line graphs that are superimposed on the bar graph. The following shows the view immediately above, with all children selected and their vertical scale exaggerated by a factor to 400% of the parent object scale. The lines for the highest-contributing children extend off the top edge of the chart, but the difference in relative contribution of the three lesser modules is easier to distinguish because their lines are further apart.



5.1.7.6.2Zooming in on a time range

To zoom on a particular time range, press and hold mouse button 1 at one end of the time range, and drag left or right. A rectangle shows the time range selected (the vertical dimensions of the rectangle are not relevant to the result of the selection). The following shows the initial selection of a time range within a profile:



When you release mouse button 1 the selected area is zoomed:



You can drag a zoomed view forward or backward in time by holding down mouse button 3 (the right mouse button for left-handed users) and dragging. The following animated graphic shows this effect:



Note that the chart bars and lines move up and down as you drag: this is because, each time Profile Analyzer handles an increment of the drag event, it redistributes the trace events acording to the current co-ordinates.

You can further zoom an already zoomed image by selecting a new zoom area:



This zooms to the time range 5.76 to 9.10 seconds:



To restore the time view to the full duration of the profile, right-click over the bar chart.

5.1.7.6.3Time cursor

From VPA 5.0 on, temporal profiling view is decorated with two time cursors, one for start time, and the other for end time. Temporal profiling view displays a column chart of time intervals of all the profile data. Time is labeled along below the column chart, once every some time intervals. The time cursor helps user accurately position the time line of the column chart. When user selects the cursor and drags it along the column line, cursor time is displayed as it is moved. At first, the start cursor is at the bottom left and the end cursor is at the top right.



it is the temporal profiling view for a .etm profile data file with time data. The two cursors are displayed at the initial positions in the column chart.



When user drag the cursors, time line is displayed and time data is changed as the cursor is moved. Here the start cursor is labeled 11.26, and the end cursor is dragged to 23.78, in second.



Time cursor helps user focus on a range within a time interval, and helps user position the mouse to select a concerned range of time interval. In the above graph, the cursors having displayed a range of time interval, if user is interested in the interval of profile data, he can drag a rectangle aligned with the cursors. It helps user ranged area of time intervals more accurately.



When user selects a time interval range with mouse, temporal profiling view refreshes the graph, and displays new column chart of profile data inside selected time interval. The above chart is resulted from selecting time intervals range from 8.91 second and 32.75 second. The start cursor is at the left most and the end cursor is at the right most. In this graph time cursor is able to move its position too.

5.1.7.6.4Time/memory profiling

The temporal profiling view also lets you view how tick events for a selected object are distributed within a matrix of memory and time. This is mainly of interest to compiler writers, or other specialists concerned with how well busy sections of a symbol or module are distributed within a processor's instruction cache. To switch to the

Time/memory profiling view select the **Show memory usage over time** icon . Select a profile object for which this view makes sense - typically a symbol or module. (It is not normally productive to view the entire profile or a particular process in this view, because individual libraries within the profile or even a single process may occupy widely different memory ranges.) The following shows a time/memory view of the JITCODE module of a java process (the module containing JIT-compiled Java methods):



Individual colored rectangles in this view represent profile intensity, for a given time and a given address range. To produce this view, Profile Analyzer divides the profile ticks for the selected object (profile, process, module or symbol) into equal time intervals (determined by the left-hand or time interval slider, as for the **Tick intensity over time** view) and divides the memory range of the selected object into equal memory intervals (determined by the right-hand or memory interval slider). Individual rectangles that contain ticks represent a region of memory that was busy at a particular time. If there is sufficient space, Profile Analyzer displays the tick count for each active intersection within the rectangle. If you increase the number of intervals the ticks may disappear but the color scheme still gives an indication of which memory ranges are busy at what times, with darker shades denoting higher tick counts:



The table at the bottom of the view displays information about the active objects in a particular rectangle in the view. To use this capability, make sure the Temporal Profiling view is the active view (click on the tab), then **hold down the Shift key** while moving the mouse. As the pointer moves over different areas of the graph the objects that occupy the memory range for the current area are shown in the table. Two tick counts are shown for each object: **In range** identifies the number of ticks the indicated object contributes to the current time/space interval, while **Total** represents the number of ticks the indicated object contributes to the profile as a whole. In the following view, the Shift key is being held down and the pointer is over the rectangle with a tick count of 426. The busiest symbols for that time/memory range are displayed at the top of the table:

Disassemb	sassembly/Offset Information 🛛 Java/Classes Hierarchy 🛛 Console 👫 Temporal Profiling 🗙 💦 🖓 🗖									
	2	16	24	15	94	101	102	95	105	76
		36	83	19	115	130	179	180	193	164
		10	57		4	7	6	6	9	7
		2	35	3	6	11	12	13	10	8
			13		1					
			4							
			13	65	257	261	339	378	385	279
				46	252	308	416	426	410	331
				8	88	76	107	96	100	78
					122	164	177	179	178	162
Ó					3		5			Seconds
Module	JITCODE	, 7654/765	54 events four	ıd					1 2	Entire profile
•	▲ 10 intervals, 763.3 ms/interval 10 intervals of 27K								Þ	
Name	Name							range	Total	<u> </u>
com/ibm/torolab/XMLdis/SizedLong.toHexString()Ljava/lang/String;								54	312	
com/ibi	com/ibm/torolab/XMLdis/FormatOperation.format(Ljava/lang/Object;)Ljava/la							41 41	229	
com/ibi	m/torolab/	XML dis/Di	isassembler a	etGlobalEori	Name(Liava	/land/Strind:		31	175	•

Once you release the Shift key, the table contents do not change. This allows you to use the mouse to scroll through the table after you have chosen a particular rectangle, without mouse movements changing what is displayed in the table.

5.1.7.6.5How to use the time/memory view

The time/memory view below is for a static compiled function within the garbage collection module of the J9 virtual machine for Java:

Disassembly/Offset Information J	isassembly/Offset Information Java/Classes Hierarchy Console 🔝 Temporal Profiling 🗙 🛛 🖓 🗖							
	2	36	57	47	64	57	68	36
	5	87	271	199	254	260	244	157
		6	7	7	3	8	6	2
		21	75	70	89	80	82	43
	1	12	21	20	12	23	21	13
0.0	2.0		4.0		6.0		8.0	Seconds
 Symbol MM_MarkingScheme	etscanPointe	rArrayObjec	tSplit, 2466/	2466 events	s found		10 30	Ent e profile
 I I 	0 intervals, 9	64.6 ms/inte	rval		6 inter	vals of 69	•	Þ
Name In range Total								

In this view, each row represents 69 bytes (as shown by the legend beside the right-hand slider bar). You can see that there are two fairly busy ranges: the first range consisting of the top two rows, and the second range consisting of the fifth row (the row whose first displayed value is 21 ticks).

One use of the time/memory view is to show whether code is properly ordered within busy symbols. For instance, the above function might provide better performance if the areas of code that are busy were closer together in memory, as they would likely use fewer I-cache lines if grouped together. Note that you should only attempt code reordering based on the time/memory view after analyzing the same symbol in several profiles, and there are no guarantees that your reordering will yield improvements. For example, compilers may completely reorder sections of your code when they generate the machine code for a symbol. However, this view may help you identify symbols or modules with time/memory usage patterns that warrant further investigation.

5.1.8Configure database connections and manage cached database files

To improve the performance, Profile Analyzer will load its files into a database, either hsqldb or DB2, instead of keeping them in the memory, sometimes in page files. Then, each action just executes a query to get the data needed without any useless data.

To open this view, click Window \rightarrow Show View \rightarrow Other, then, select Database Connections under Visual Performance Analyzer category, as below.

🖨 Show View 🛛 🔀
type filter text
PDE Analyzer
Heam Heam
OK Cancel

Open View

When you start VPA for the first time, a default connection of hsqldb will be created, as well as product supports under this connection node, as below.

🖥 Database Connections 🗙 Navig	gator	Remote	Data	Collectio	
				¢,	<u>A</u> ↓ - ▽
	Size	• I	Date		
🖃 🗗 Local Connection (Hsqldb)					
🖅 🤮 Counter Analyzer	N/A				
🛓 📊 Profile Analyzer(active)	N/A				

Default Connection and Product Supports

The only Profile Analyzer support is set as active, so that user can open files without any setting actions. However, you can create other connection or edit this default connection as you like, such as modifying its path.

e	×
Choose Database Choose database to create connection.	-
⊙Hsqldb ○IBM DB2	
⑦ < Back Next > Finish Cancel	
e V	×
New Hsql Connection Set the properties for Hsqldb connection.	
Connection Name: My Hsqldb	
Connection Name: My Hsqldb Database Path: c:\hsqldb .	
Connection Name: My Hsqldb Database Path: c:\hsqldb .	

Create Hsqldb Connection

e		\mathbf{X}
Choose Database Choose database to cree	ate connection.	
○ Hsqldb ⊙ IBM DB2		
? < <u>Back</u>	Next > Finish Cancel	
•	• • • • • • • • • • • • • • • • • • •	×
New DB2 Connection Set properties for D	on 182 connection.	
Connection Name:	9. 186. 10. 107	
Host Address:	9. 186. 10. 107	
Port:	50000	
DB Name:	dbmodel	
User Name:	db2admin	
	Test Connection .	
⑦ <u>< B</u> ack	<u>Mext</u> > <u>F</u> inish Cane	el

Create DB2 Connection

After these two connections have been created, the view looks as below.



User can create Profile Analyzer support under each connection, for common use, hsqldb is enough, for performance consideration, db2 is the better choice.

⊕ Local Co ⊕ 9. 186. 10 ⊕ Mr. Hall ⊕ Mr. Hall ⊕	onnection (Hsqldb)). 107 1						
	🗣 New Connection						
📝 Edit Connection							
	🗙 Delete Connection						
	👖 Create Profile Analyzer Support						
	😫 Create Counter Analyzer Support						

Create Profile Analyzer Support under Hsqldb Connection



Create Profile Analyzer Support under DB2 Connection

Hsqldb, as we known, is an embedded database system, so it must have some limitations. To prevent too much disk space occupied as more and more files being loaded in, you should set a size limitation. This num of Size Limitation stands for the upper limit of the hsqldb's data file. If the database file was larger than the num, system will delete oldest files, until the size of the database file is smaller than the size limitation. During the auto delete process, some file whose size is larger than the size limitation will be deleted first.



Set Hslqdb's size limitation for Profile Analyzer Support

You can also delete files manually to release disk space. Multi-selecting is allowed. See the picture below.

	Size	Date	
🖃 🗗 Local Connection (Hsqld			
🕀 🖸 Counter Analyzer	N/A		
🕀 📊 Profile Analyzer(act	N/A		
🗐 🚰 9. 186. 10. 107			
🖨 🖵 My Hsqldb			
🛓 <u> </u> Counter Analyzer	1.16		
🚊 📊 Profile Analyzer	4.24		
	1.02	2007-04-24 11:30	
	1.20	2007-04-29 18:21	
C:/Documents_and	1.01	2007-05-08 11:09	
C:/Documents 🗙 D	elete Cach	ed Files 29	

To set one Profile Analyzer support as active, you should switch the perspective to Profile Analyzer Perspective, so that the related menu named "Set Active" will occur, click it to set the connection you like as active.

Select Active Connection	×
Select Active Connection Select Active Connection to process file.	
Connection: Local Connection (Hsqldb) (active) Local Connection (Hsqldb) (active) 9.186.10.107 My Hsqldb	
OK Cancel)

Select Active Connection

Active Profile Analyzer Support cannot be deleted. To delete it, you should set other support as active first. Connection cannot be deleted, until you delete all the product supports under this connection.

This view support sorting operation. You can sort the files by name, size or date.

To sort, select the sort mode on the action bar, or click the title directly.



5.2Code Analyzer

You can also find the Code Analyzer User Guide from within VPA. Select **Help - Help Contents** within VPA. To get context sensitive help, press **F1** for Windows and AIX or press **Ctrl+F1** for Linux.

5.2.1Load an executable for analysis

When you first start Visual Performance Analyzer after installation, the default perspective is Profile Analyzer. To open CodeAnalyzer, you can choose **Windows -> Show Perspective -> Other-> CodeAnalyzer**.

😼 Visual Performance Analyzer <u>F</u>ile <u>E</u>dit Se<u>a</u>rch <u>T</u>ools <u>W</u>indow <u>H</u>elp 🔚 🛄 🔯 」 🦘 🗋 😡 🦉 🧃 🕘 🛆 🖴 🥸 🦉 🖬 🦉 🦓 🦾 🖓 📗 💆 🕶 🔺 📶 🎆 Frogram Tree 🛛 Navigator 🖵 🗖 🎆 Instructions Table 🛛 😕 - 🐉 🔟 🛛 is is is PPC 🛐 🚞 🚹 - 8 🛛 Static Color Bar 🗙 i i Address Opcode Mnemonic Comment Freq. Graph < > 📔 🖉 🍟 🗖 Instruction Properties 🔚 Disassembly/Offsets 🔀 📃 CAConsole 🖂 👜 | 🌩 🐇 | 🍑 🏱

The following screen capture shows the default workbench window of CodeAnalyzer.

Choose File -> CodeAnalyzer -> Analyze Executable.

SVisual Performance A	nalyzer							
<u>File E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> in	ndow <u>H</u> elp							
Open File <u>.</u>	1 P P = A O F S	3 200	ni) 🗛 🗍 🖢 📼	1 · A				
<u>C</u> lose Ctrl+W	🗖 🗖 🚺 Instructions Table 🖂	13 - 同	5 🔟 i _s i _s	i, PPE 🕅			🚮 Static Color Bar 🗙	
C <u>l</u> ose All Ctrl+Shift+W							i	i 🔻
CodeAnalyzer	🕨 😡 Analyze Rxecutable	ode	Mnemonic	Comment	Freq.	Graph	Bree	
Exit	🦉 Add <u>P</u> rofile Info							
	Add Sampling Info							
	Collect Dispatch Group Info							
	A Collect Hazard Info							
	Actions	-						
		-						
	Statistics							
						>		
🚍 CAConsole 🔀		🗋 🖉 🎽 '	- 🗆 Instructi	on Propertie	s 🔝 I	isassembly/C)ffsets 🛛	
							🚔 🔶 🗠	⇒

In the pop-up wizard, select the executable you want to analyze.

🖗 Choose Executable File to Analyze	
Open File:	
E:\TOOLS\CodeAnalyzer\example\li32\li32	
Recent List	
	Open Cancel

If you press **Open**, the executable will be loaded in CodeAnalyzer.

ScodeAnalyzer - E:\TOOLS\Co	deAnalyzer\example\li	32\li32 - Ecli	pse Platform				ScodeAnalyzer - E:\TOOLS\CodeAnalyzer\example\li32\li32 - Eclipse Platform						
<u>F</u> ile <u>E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> eJ	lp												
🌆 📊 🔽 🌤 🛷 😡 🦉 🐗	- North All 🚘 🍕 👯 🕅 🎙	🕽 🖹 🍕 🥙 📩	ぬ ぬ - ᅒ -	жÌ									
Program Tree 🔀 Navigator 🗖 🗖	Instructions Table 🗙	B	- 💦 🔞 🛛 i	i _b i _d PPC	8 🗆			🙀 Stati	. 🛛 🗖 🗖				
	File: crtOmain.s Func	tion: Index: 0	BB: addr: Ox100	0001c8 siz	e: 11 (4	4) exec: (0		i i ▽				
File: crt0main.s	Address	Opcode	Mnemonic	Comment	Freq.	Graph	^	Fred					
	≥FILE crt0main. s	FUNCTION						TT AT					
	0x100001c8	0x82420000	lwz r18,0(r2)	sta	0								
🖃 🔁 xlisp. c	0x100001cc	0x80f20008	lwz r7,8(r18)		0								
± 📄 . main	0x100001d0	0x39000000	li r8,0		0								
🖻 🗁 os. c	0x100001d4	0x91070000	stw 18,0(r7)		0								
🕀 📄 . osfinish	0x100001d8	0x81320000	lwz r9,0(r18)		0								
🕀 📄 . osfinit	0x100001dc	0x90690000	stw r3,0(r9)		0								
🛨 📄 . oscheck	0x100001e0	0x81320004	lwz r9,4(r18)		0								
😟 📄 . osputc	0x100001e4	0x90890000	stw r4,0(r9)		0								
🕀 📄 . osgetc	0x100001e8	0x80e20004	lwz r7,4(r2)	-> err	0								
🛨 📄 . osrand	0x100001ec	0x91070000	stw r8,0(r7)		0								
🕀 📄 . osinit	0x100001f0	0x48000051	bl 0x10000240	.main	0	J.							
E-B glink.s													
tisbut	0x100001f4	0x6000000	ori r0, r0, 0x0		0								
Elink. s	0x100001f8	0x80f20008	lwz r7,8(r18)		0								
🛨 📄IIIDUI	0x100001fc	0x80e70000	lwz r7,0(r7)		0								
F. rand	0x10000200	0x2c070000	empi er0,0x		0								
itrunc s	0x10000204	0x4182000 c	beg cr0, Ox		0	i.							
+ itrunc. s													
glink. s	0x10000208	0x8112000 c	lwz r8, 12 (r		0								
🗄 🗎 . fp_raise_xcp	0×1000020 c	0x80680000	1wz r3.0(r8)		0								
□ 🔁 💶						4							
🕂 🗎 . printf 🛛 🕙	0-10000210	04000E0.3E	31 0-100052.4	: .	0	15	⊻						
📃 CAConsole 🖂			Instruction Pr	operties 📗	🔓 Disass	embly/Offse	ts 23						
Webug into: .\Program\Veritication\P	rogramVerifier.cpp: done.							· 🛋 📄					
@Debug into: .\Frogram\Veritication\L	teratorsTester.cpp: running	iterator tester =											
@Debug info: .\Program\Verification\I	teratorsTester.cpp: done.												
DEBUG - Free before stub loading = 316	64208												
DEBUG - Still free after stub loading	= 3171808												
DEBUG - before initInListsAndCreateFw	nctionBranches 7286664												
DEBUG - after initinListsAndureaterun DEBUG - before sortFunctionBranchList	ctionBranches (U21856 < 7401032												
DEBUG - after sortFunctionBranchLists	7225032		-										
DEBUG - Finished load	xows=32 cols=15												
DEDOG Created Res viewEines object.	10#5-02 0015-15	>	 Image: A set of the set of the										

When an executable is loaded, a wizard for further action appears as follows that allows you to load profile information.

🚳 Post-Load Options	
 ✓ Losd Profile ✓ Collect Performance Comments ✓ Perform Grouping Analysis 	Instrumentation Profile: Sampling Profile: Event name
	OK Cancel

If you want to open profile information of the loaded executable, you can choose the address of profile file in the above wizard. After pressing **OK**, the profile information will be added into CodeAnalyzer workbench window.

ScodeAnalyzer - E:\TOOLS\Co	deAnalyzer\example\li	32 \1i32 - Ecli p	se Platform						
<u>F</u> ile <u>E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> e	lp								
] 📰 🚹 💟] 🌤] 🔗] 😡 🦉 🐗	🚱 🛆 🖴 🎕 👯 😫) 🖹 😹 🏄 👬 🛛	A 🖢 - 🕅 -	<u>M</u>					
Mavigator 🖵 🗖	Instructions Table 🗙	B	- 💦 🔞 🛛 i 🖓	i _b i _d PPE	3 🗆	1		Stati 🔀	
	File: crtOmain.s Func	tion: Index: O H	B: addr: 0x100	001c8 size	e: 11(4	4) exec: 0		i	i 🗢
File: crt0main.s	Address	Opcode	Mnemonic	Comment	Freq.	Graph	<u> </u>		
	≥FILE crt0main.s	FUNCTION				Ē		e.d.	<u> </u>
	0x100001c8	0x82420000	1wz r18,0(r2)	sta	0				
🖻 🧁 xlisp. c	0x100001cc	0x80f20008	lwz r7,8(r18)		0				
🗄 📄 . main	0x100001 d0	0x39000000	li r8,0		0				
🚊 🗁 os. c	0x100001d4	0x91070000	stw r8,0(r7)		0				
😟 📄 . osfinish	0x100001d8	0x81320000	1wz r9,0(r18)		0				
🕀 📄 . osfinit	0x100001 dc	0x90690000	stw r3,0(r9)		0				
🛨 🖷 📄 . oscheck	0x100001e0	0x81320004	1wz r9,4(r18)		0				
± sputc	0x100001e4	0x90890000	stw r4,0(r9)		0				
+	0x100001e8	0x80e20004	1wz r7,4(r2)	\rightarrow err	0				
+ Srand	0x100001ec	0x91070000	stw r8,0(r7)		0				
elink e	0x100001f0	0x48000051	bl 0x10000240	.main	0	¥			
fl shuf									
glink s	0x100001f4	0x6000000	ori r0,r0,0x0		0				
😟 🗎 . filbuf	0x100001f8	0x80f20008	lwz r7,8(r18)		0				
E 🔁 glink. s	0x100001fc	0x80e70000	1wz r7,0(r7)		0				
🛨 📄 . r and	0x10000200	0x2c070000	cmpi cr0,0x		0				
🖻 🥟 _i trunc. s	0x10000204	0x4182000c	beq cr0, Ox		0	lγ			
🗄 📄 _i trunc. s									
🖻 🧁 glink. s	0x10000208	0x8112000c	1wz r8,12(r		0				
🕀 📄 . fp_raise_xcp	0x1000020c	0x80680000	1wz r3,0(r8)		0				
e jink.s						۲.	~		
+ ~ = . printf	0~10000210	04000E0.3E	11 0-100052-4		0	רו ו			
📮 CAConsole 🛛			Instruction Pr	operties 📗	Disass	embly/Offsets	5 23		
DEBUG - Number of BBs: 4590 DEBUG - Still free often stub loading	- 2171909	^					1	🖴 🥧 🐇	⇒ ⊽
DEBUG - Still free after stub loading DEBUG - before initInListsAndCreateFu	5 - 3111000 mctionBranches 7286664		ĨI					<u>o i r i i</u>	
DEBUG - after initInListsAndCreateFun	ctionBranches 7021856								
DEBUG - before sortFunctionBranchList DEBUG - after sortFunctionBranchLists	: 7401032 : 7225032								
DEBUG - Finished load									
DEBUG - created new viewLines object: nullDEBUG - Free after color assignme	rows=32 cols=15 of = 5690144								
DEBUG - Free before color assignment	= 5647856								
DEBUG - Free before color assignment	= 5647856 = 5637368								
DEBUG - Coloring finished									
L		×							

5.2.2Adding profiling information

If you have been working with an executable, without profiling information, you can add the profiling information by Choosing File -> CodeAnalyzer -> Add Profile Information.

If you find the profile information file, of the loaded executable, you can choose profile file in the wizard. After pressing **OK**, the profile information will be added into CodeAnalyzer workbench window.

The following screen capture shows the workbench window after profile information is loaded.

ScodeAnalyzer - E:\TOOLS\Co	deAnalyzer\example\li	132 \1132 - Ecli p	se Platform					X
<u>F</u> ile <u>E</u> dit Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> el	Lp							
🌆 🚹 🔽 🖙 🖋 😡 🦉 🐗	💽 🛆 🕋 🙊 👯 🖬 🕻	B 🖹 🏄 🍻 👘	A 🖢 - 🖓 -	<u>A</u>				
🚮 Program Tree 🔀 Navigator 🗖 🗖	Instructions Table 🗙	B	- 💦 🐚 🛛 i 🖓	i _b i _d PPE	3 🔲 🔟 🤉		🎆 Stati 🕱 🖵	
	File: crtOmain.s Fun	ction: Index: 0]	BB: addr: Ox100	001c8 size	e: 11(44) exe	:c: 0	i i	$\overline{\nabla}$
File: crtOmain s	Address	Opcode	Mnemonic	Comment	Freq. Graph		Free	
E- C crt0main s	FILE crt0main.s	FUNCTION					rred.	<u> </u>
	0x100001c8	0x82420000	lwz r18,0(r2)	sta	0			
- > xlisp. c	0x100001cc	0x80f20008	lwz r7,8(r18)		0			
🗄 📄 . main	0x100001d0	0x39000000	li r8,0		0			
	0x100001d4	0x91070000	stw r8,0(r7)		0			
🕀 📄 . osfinish	0x100001d8	0x81320000	1wz r9,0(r18)		0			
🕀 📄 . osfinit	0x100001dc	0x90690000	stw r3,0(r9)		0			
🖅 📄 . oscheck	0x100001e0	0x81320004	lwz r9,4(r18)		0			
. osputc	0x100001e4	0x90890000	stw r4,0(r9)		0			
±…≣ .osgetc	0x100001e8	0x80e20004	lwz r7,4(r2)	-> err	0			
+ . osrand	0x100001ec	0x91070000	stw r8,0(r7)		0			
eliste e	0x100001f0	0x48000051	bl 0x10000240	.main	0 🗸			
fl shuf								
E glink s	0x100001f4	0x6000000	ori r0,r0,0x0		0			
😟 🗎 . filbuf	0x100001f8	0x80f20008	lwz r7,8(r18)		0			
E-> glink.s	0x100001fc	0x80e70000	1wz r7,0(r7)		0			
主 🖷 📄 . r and	0x10000200	0x2c070000	empi er0,0x		0			
🖻 🦻 _i trunc. s	0x10000204	0x4182000 c	beq crO, Ox		0 7			
🗄 📄 _i trunc. s								
🖹 🗁 glink. s	0x10000208	0x8112000c	1wz r8, 12 (r		0			
主 📄 . fp_raise_xcp	0x1000020c	0x80680000	1wz r3,0(r8)		0			
E Bink.s					* ا	~		
i i+i= .printf	0-10000210	0400050.35	11 0.100052.4					
📮 CAConsole 🔀			Instruction Pr	operties 📗	Disassembly/O	ffsets 🖾		
DEBUG - Number of BBs: 4590 DEBUG - Still free ofter stub loading	= 3171808	^					🚔 🔿 🕤 📦	\bigtriangledown
DEBUG - before initInListSAndCreateFu	nctionBranches 7286664						<u> </u>	
DEBUG - after initInListsAndCreateFund	ctionBranches 7021856							
DEBUG - after sortFunctionBranchLists	7225032							
DEBUG - Finished load	-00 3 -15							
nullDEBUG - Free after color assignment	rows-32 cois-15 nt = 5690144							
DEBUG - Free before color assignment =	= 5647856							
DEBUG - Free before color assignment = DEBUG - Free before color assignment =	= 5647856 = 5637368							
DEBUG - Coloring finished								
L		×	9					

5.2.3Navigate the Executable

5.2.3.1 Navigate the Program Tree

Program tree displays a hierarchical view of the loaded executable. It is automatically opened in the left side of the workbench window when you first use CodeAnalyzer. You can open program tree view by choosing **Windows -> Show View -> Program Tree**. First, you need to load an executable. Then navigate program tree by expanding all, collapsing all, sorting, going into opening source code and getting control flow.

5.2.3.1.1Expand All

To get all the functions under each file in the loaded executable, press button \square in title bar. The result view of program tree expands as follows:
🚰 Program Tree 🗙 Navigator 🛛 🗖 🗖
🖽 🖻 📴 📭 👫 🍐 🔶 🔫
File: xldbug.c
⊡… 🤁 crtOmain.s 🔥 ⊞… 📄start
🚊 🗁 os. c
🗄 🖷 📄 . osfinish
🛨 ··· 📄 . osfinit
🕀 📄 . oscheck
🛨 🖷 📄 . osputc
🗄 📄 . osgetc
🗄 📄 . osrand
主 📄 . osinit
🖻 🗁 glink. s
😟 📄flsbuf
🖻 🗁 glink. s
🗄 📄filbuf
🖻 🗁 glink. s
🛨 📄 . rand
🚊 🗁 _i trunc. s
🗄 📄 _i trunc. s
🗄 🗁 glink. s
🗄 📄 . fp_raise_xcp
🖻 🕞 glink. s 🔛

5.2.3.1.2Collapse All

To close all sub-items of each file in the executable, press button in title bar. The result view of program tree shows as follows:

Program Tree 🗙 Navigator 🗖	
🖽 🖻 🎝 🞝 👫 🏠 🔶 🔿	•
File: xldbug.c	
🗄 🗁 glink. s	~
🗄 🗁 glink. s	
🗄 🗁 streat. s	
🗄 🗁 glink. s	
🗄 🗁 longjmp. s	
🗄 🗁 glink. s	
🗄 🗁 longjmp. s	
🗄 🗁 xlinit. c	
🗄 🗁 xlbfun. c	
🗄 🗁 xllist. c	
🗄 🗁 xlmath. c	
🗄 🗁 sin. c	
🗄 🗁 cos. c	
⊞… 🥭 tan. c	_
🗄 🗁 exp. c	
🗄 🗁 sqrt. s	
⊞… 🥭 pow. c	
🗄 🗁 noname	=
E- Z xlcont. c	
🗄 🗁 xlsys. c	
🗄 🗁 xlstr. c	
	-
	The second secon

5.2.3.1.3Sort

There are three kinds of sorting in program tree view: lexicographical order, ascending order and descending order. You can press their corresponding buttons $L^2_{\mathbb{Z}}$, $L_{\mathbb{H}}$, in the view's title bar or right-click any object in the view and choose these items.

5.2.3.1.4Go Into

If you want to get detailed information of the items which are under certain object, you can select this object and press. The program tree will display all its sub-items. If you want to go to upper level, just press. Pressing an lead you to root view. For example, if you want to see all the functions in file os.c, choose os.c file in program tree as follows:

👖 Program Tree 🗙 Navigator 🗁	, 🗆
🕀 🕞 📴 📭 👫 🍐 🗇 👄	-
File: os.c	
🕀 🗁 crtOmain. s	~
🗄 🗁 xlisp. c	
🗄 🗝 os. c	
🗄 🗁 glink. s	
🗄 🗁 glink. s	=
🗄 🗁 glink. s	
🗄 🗁 _i trunc. s	
🗄 🗁 glink. s	
🗄 🗁 glink. s	
🗄 🗁 xljump. c	
🗄 🗁 xldbug. c	
🗄 🗁 xlprin. c	
🗄 🗁 glink. s	
🖭 🥭 xlio. c	
E- 🔁 xleval. c	
🗈 🗁 xlsym. c	
🗄 🗁 xldmem. c	
🗄 🗁 glink. s	$\mathbf{\sim}$

The result view displays all the functions within this file.

🌇 Program Tree 🗙 Navigator 🛛 🖓 🗖
🕀 🕞 📴 👫 👫 🏠 🗇 🗢 🔫
File: os.c Function: .osfinish
🕀 🖷 📄 . osfinish
🕀 📄 . osfinit
🗄 📄 . oscheck
🗄 🖷 📄 . osputc
🗄 🖷 📄 . osgetc
🗄 📄 . osrand
主 📄 . osinit

You can also navigate forward or backward as you like.

5.2.3.1.5Open source code

To open source code, right-click any object in program tree and select **Open Source Code**.

🌇 Program Tree 🗙	Navigator 🛛 🗆 🗖
E E] Jª₂ ↓ ↓ 🍐 ⇔ 🔿 🔻
File: xljum	p.c Function: .xlreturn
🚊 🗁 xljump. c	~
🕀 📄 . findta	rget
🕀 📄 . xlsign	al 🗾
😟 📄 🔝 xlthro	*
🕀 📄 🔝 xlret	
🕀 📄 . xlgo	↓ ZLexicographical Sort
🕀 📄 . xlcor	🚰 Ascending Sort
🕀 📄 .xlcle	Ascending Sort
🗄 📄 .xltor	•
🕀 📄 . xlju	🏠 Go Home
🛨 📄 .xlend	
🛨 🕀 📄 .xlbe;	A GO DECK
E Z kldbug.c	🛶 Go <u>I</u> nto
🛨 📑 .xlabd-	
tertar	Control Flow
🕂 📄 . stdpi	Store Same Cale
🕂 📄 . stdpr	a open Source Code
	t
🛨 🖃 . Xibakti	race
🛨 📄 . Stackti	op
E doornoo	oop ~
H larro	rint 🔽

5.2.3.1.6Get Control Flow

To get calling functions and called functions, select a function in the program tree, right-click and choose **Control Flow**.

🎢 Program Tree 🗙 🦹 🗖 🗖	Inst	ruction T	able 🖾	
▽	File	: xljump.	c Function:	
	Addres	5	Opcode	
	🔁 FILE	xl j	FUNCTION .	fi
File. Xijump. c Function.	0x100	0008cc	0x7c0802a6	
H . Ip_raise_xcp	0x100	00840	0x9421ffb0	
Hunger gilling swintf	0x100	000844	0x90010058	
E princi	0x100	868000	0x90610068	
H find	0+100	2008dc	0x9081006c	
Lexicographical	Sort	008e0	0x8062002c	
🗄 📄 .xltl 📙 Ascending Sort		008e4	0x80630000	
E . xlre I According Sout		008e8	0x28030000	
. xlgo		008ec	0x90610040	
🕀 📄 . xlco 👔 Go Home		008£0	0x41820034	
🗄 📄 . xlcl				
E xlto		008£4	0x80610040	
🕀 📄 . xlju 📥 Go <u>I</u> nto		008f8	0x80a30000	
the start and st		6.11.4	P	
the control Flow	· · ·	Lailed	Functions	
🕂 🚽 👔 . xlat 📓 Open Source Code	. [Calling	g Functions	_
🕀 📄 .xlfail	010	00000	02900000	
🕀 📄 . stdputstr	0x100	00000	0x38a00000	
🕀 📄 . stdprint	0x100	00906	0x40000430	
🕂 📄 . xldinit 😽 😽			0.00040040	

The called functions of the selected function in the above screen capture show as follows:

€	🛢 Outgoing Branches 🛛 🗙								
	Function Name .xljump .xlabort	Address 0x10000d68 0x10000edc	File Name xljump.c xldbug.c	Executes 0					
			OK	Goto Function					

The calling functions of the selected function in the above screen capture show as follows:

e	Incoming Br	anches		X
	Function Name	Address	File Name	Executes
	.xlcontinue	0x10000c78	xljump.c	0
	.xlcleanup	0x10000cc8	xljump.c	0
	.xltoplevel	0x10000d18	xljump.c	0
			02	Cata Rupatian
			40	Goto Function

5.2.3.2View Static Color Bar

Static color bar gives an overview of frequency distribution of basic blocks in the loaded executable. You can open this view by choosing **Windows -> Show View -> Static Color Bar**.

To obtain this information, do following steps:

- 1. Load an executable
- 2. Add profile information

After profile information is added, there is a yellow pointer in the static color bar. It indicates the position of basic block you have selected. The yellow pointer scrolls in accordance with the basic block you select in the **Instructions Table** or **Program Tree**.

The following screen capture shows the selection of a specific color bar.

🗟 CodeAnalyzer - E:\TOOLS\CodeAnalyzer\example\li32\li32 - Visual Performance Analyzer							
File Edit Segrch Tools Window Help							
] 🛄 🔟 📭] 🍫] 🖉] 😡 🦉 🚜 🖙 🙊 🐼 🗱 🐚 🕲 🦽 🐗 🧄 🛄] 🖓 - 🐼 - 🔕							
🎆 Program Tree 🛛 Navigator 🛛 🗖 🎆 Instruction Table 🛇 🧏 📲 🦻 is is ppr 📓 🗔 🔢 🔛 🖬 🗸 🖓							👫 Static Colo 🗙 🖳 🗆
	File: xleva	1.c Function: .:	xleval Inde	x: 752 BB: #	addr: 0x10003	318 size: 6(24)	i i 🗸
File: xleval.c Function: .xleval	Address	Opcode	Mnemonic	Comment	Freq.	Graph 🔨	Free
+							rred.
🗈 📄 .putfloat	0x10003318	0x80820118	lwz r4,28	-> xltrac	365822		
😟 📄 . putdec	0x1000331c	0x80640000	lwz r3,0(365822		
🕀 📄 . putatm	0x10003320	0x3063ffff	addic r3,		365822		
🕀 📄 . putstring 📃	0x10003324	0x90640000	stw r3,0(365822		
🛨 📄 . xlputstr 🦳	0x10003328	0x80610068	1wz r3,10		365822		
庄 📄 .xlterpri	0x1000332c	0x90610044	stw r3,68		365822		
😟 📄 . xlprint						4	
🖻 🥭 glink. s	0x10003330	0x80010058	lwz r0.88		365822	-	
😟 📄 . sprintf	0x10003334	0x7c0803a6	mtlr rO		365822		
🛱 🗁 xlio. c	0x10003338	0x30210050	addic r1		365822		
庄 📄 . xlflush	0x1000333e	0x4e800020	helr Ox14		365822		
🕀 📄 . xlputc		TRACE BACK	bear out the				
🖭 📄 . xlpeek	0v10003340				0		
🛨 🖷 📄 .xlgetc	0v10003344	0v00002041	۰ ۵		0		
E- xleval. c	0v10003348	0v80000101			0		
L. xisave	0x1000334a	0v0000000			0		
+ 1 skeyword	0+10003350	0+000000068			0		
+	0x10003350	0x0000010			0		
+ eviun	0x10003354	0x00001000	x1		0		
	0x10003356		evat		U	•	
T P and back	0.4000005	N UBKEALRED					
	0x1000335c				U		
Two vierning							
+ vlvevel	0x10003360	Ux7cU8U2a6	mfir rU	, .xlsinit	1		
	0x10003364	Oxbfalfff4	stmw r29,		1		
	0×10003368	0×90010008	stwirf) 8 (
CAConsole 🗟 Comments 😒					Instruction	Properties 🛛	\$ ~ □ □
🗢 ≑ Description	File	Function	🗢 Addres	<u>s </u>	Branch Profile	Value Profile Di	spatch Info
\Lambda Load instruction accesses the sa	xlmath. c	. checkfneg	0x1000edc8		Resource Name	Value	Count
🔥 Load instruction accesses the sa	xlmath. c	. checkfzero	0x1000ee34				
🔥 Load instruction accesses the sa	tan.c	.tan	0x1000fbc4				
🔥 Load instruction accesses the sa	tan. c	. tan	0x1000fce4				
🔥 Load instruction accesses the sa	exp. c	. exp	0x100100ec				
🔥 Load instruction accesses the sa	exp. c	. exp	0x10010120				
🔥 Load instruction accesses the sa	exp. c	. exp	0x10010184				
🔥 Load instruction accesses the sa	pow.c	.expinner2	0x100103c0				
▲ Load instruction accesses the sa	pow.c	.expinner2	0x10010494				
	0 1EO	5.6E4 13E4 23	E4 36 <mark>E4</mark> 4	<mark>6</mark> E4 51E4	87E4 91E4		

5.2.3.3Navigate Instructions View

Instructions view is the default view of Instructions Table. It shows the contents of an executable or shared object as a table of assembly instructions, with its control flow graph drawn vertically at its side. To open this view, select **Switch to instructions** in the Instructions Table's title bar. The explanation of functions of some buttons in title bar can be found in Instructions Table.

In instructions view, when a new file begins, there is a line of table beginning with icon and followed by its name. When a new function starts, there is also declaration in a separate line. Instructions which belong to the same basic block are organized together with blank lines between them. The last basic block in a function begins with a line as follows.

Instruction	Table 🗙	В	- 杉 🐚 🕴 i _g	i _b i _d PPE 🛐	🔲 🚹 🗸 🖻	- D
File: ???	Function: ???	Index: 3632 B	B: addr: Ox10	00ff99 size:	1(3) exec: O	
Address	Opcode	Mnemonic	Comment	Freq.	Graph	
0x1000ff74	0xfc823000	fempu er1,		0		
0x1000ff78	0x4bfffc30	b 0x1000fba8		0	Ł	
0x1000ff7c	0x4e800020	N		0		
	🔀 TRACE BACK					
0x1000ff80	0x0000000			0		
0x1000ff84	0x00002240	"@		0		
0x1000ff88	0x0000002			0		
0x1000ff8c	0xc0000000			0		
0x1000ff90	0x0000041 c			0		
0x1000ff94	0x00037461	ta		0		
0x1000ff98	0x6e	n		0		
	🖉 UNBEACHED					
0x1000ff99	0x000000	. 63		0		
🍃 FILE exp. c	FUNCTION . exp					
0x1000ff9c	0x8082022c	lwz r4,556	, .exp, =>	0		-
0x1000ffa0	0xfc000a10	fabs f0, f1		0		
0x1000ffa4	0x80620230	lwz r3,560	-> 0x2000	0		
0x1000ffa8	0xfd40048e	mffs f10		0		
Ox1000ffac	0x80a20234	lwz r5,564	->itab	0		
0x1000ffb0	0xc0630000	lfs f3,0(r3)		0		
0.4000.00.4	0.0000004	20 00 000				

After adding FDPR-Pro profile information, you can get the frequency of each instruction in color. The denotation of each kind of color can be found in the lower part of workbench window. Instruction groups are indicated in the front of each instruction. In **comment** column, there are red triangles which contain performance comments to specific instructions.

Instruction	Table 🗙	B	- 杉 🐚 🕴 i.	i, id PPE 🖹	□ 1			
File: xldmen	File: xldmem.c Function: .mark Index: 1017 BB: addr: 0x10004714 size: 8(32) exec:							
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^		
					<u>ا لا</u>	_		
0x10004714	0x80810040	lwz r4,64(973686				
0x10004718	0x9081004c	stw r4,76(973686				
0x1000471 c	0x88640001	lbz r3,1(r4)		973686				
0x10004720	0x60630001	ori r3,r3,		973686		-		
0x10004724	0x5463063e	rlwinm r3,		973686				
0x10004728	0x98640001	stb r3,1(r4)		973686				
0x1000472c	0x80610040	lwz r3,64(973686				
0x10004730	Ox4bfffa85	bl 0x10004	livecar	973686	Ł			
			43					
0x10004734	0x2c030000	cmpi cr0,0		973686				
0x10004738	0x41820044	beq cr0, 0		973686	$\overline{\mathbf{Y}}$			
0x1000473c	0x80810040	lwz r4,64(868467				
0x10004740	0x90810050	stw r4,80(868467				
0x10004744	0x88640001	1bz r3,1(r4)		868467				
0x10004748	0x60630002	ori r3,r3,		868467				
0x1000474c	0x5463063e	rlwinm r3,		868467				
0x10004750	0x98640001	stb r3,1(r4)		868467				
0x10004754	0x80610044	lwz r3,68(868467				
0x10004758	0x90610048	stw r3,72(868467		~		
0.4000485	0.00040040	a a.e./		000107				

If you right-click any line item in Instruction View, a menu list appears. Typically, there are four sections of this menu. In the first section, you can get PPC(Power PC) assembly reference help, branch profile information, dispatch information, and set points to collect the value of important resources during profiling. In the second section, you can choose to find specific instruction or open source code of the selected instruction. In the third section, the menu will show the target of the basic block (**Fall thru** means the next basic block) to which the selected instruction belongs. The last section shows the callers of the basic block to which the selected instruction belongs. The first and the last two sections may be varied according to the selected instruction.

5.2.3.3.1Get PPC help

To get the assembly reference of certain instruction, right-click on the instruction and choose Show PPC Help. You can also select the instruction and press ^{PPL} button in title bar.

5.2.3.3.2Show Dispatch Information

To get the dispatch information of certain instruction, right-click this instruction and choose **Show Dispatch Info**. You can also select this instruction and press i_{c} button in title bar. To obtain more detailed description, please refer to View dispatch information.

5.2.3.3.3Show Branch Information

To get the branch information of certain basic block, right-click the last instruction of this block and choose **Show Branch Info**. You can also select the last instruction of this basic block and press i_{B} in title bar. To obtain more detailed description, please refer to View branch profile.

5.2.3.3.4Collect Value Profiling

Before profile information of the loaded executable is added, you can collect value of certain resources of specific instructions. To get this information, you need to first select these instructions by right-clicking an instruction and choosing **Collect Valuing Profiling**. A wizard will appear for you to choose resources value you try to get. You can also do this by pressing **1**_p button in title bar. To obtain more detailed information, please refer to View value profile.

5.2.3.3.5Find Instructions

To find specific instructions in this view, press button sin title bar. You can also open it by right-clicking any instruction and choosing **Find**.

Instruction	Table 🗙	В	- 💦 🔞 🕴 i	i _b i _d PPE 🖹	🔲 <u> </u> 🔻 🖓 🗖
File: exp.c	Function: .exp	Index: 3633	BB: addr: Ox10	DOOff9c size:	17(68) exec: 0
Address	Opcode	Mnemonic	Comment	Freq.	Graph 🔼
0x1000ff9c	0x8082022c	lwz r4,556	, .exp, ->	0	
0x1000ffa0	Oxfc000a10	fabs f0, f1		0	
0x1000ffa4	0x80620230	lwz r3,560	-> 0x2000	0	
0x1000ffa8	0xfd40048e	mffs f10		0	
Ox1000ffac	0x80a20234	lwz r5,564	->itab	0	
0x1000ffb0	0xc0630000	lfs f3,0(r3)		0	
0x1000ffb4	0xc <mark>0830004</mark>	16- 64 4 (*3)		0	
0x1000ffb8	Oxf PPE Show PPC Hel	ւթ		0	
Ox1000ffbc	0x3			0	
0x1000ffc0	0xf 🍼 I nd			0	
0x1000ffc4	Oxd 🗿 Open Source	Code		0	
0x1000ffc8	0xc8c40088	114 16,136		0	
Ox1000ffcc	0xc8040080	lfd f0,128		0	
0x1000ffd0	0xc8440078	lfd f2,120		0	
0x1000ffd4	0xc8e40070	1fd f7,112		0	
0x1000ffd8	0xc9640060	lfd f11,96		0	
0x1000ffdc	0x40810090	ble cr0, 0		0	$\overline{\mathbf{v}}$
0x1000ffe0	0xc8640090	lfd f3,144		0	
0x1000ffe4	0xc8a40050	1fd f5,80(0	
0x1000ffe8	Oxfd21183a	fmadd f9, f		0	
0 4000 cc	0.0040000	1 C1 C 404			

The following screen capture shows the initial activating of Find wizard.

A Find Instruction wizard appears.

🐓 Find Instructio	n 🔀					
Contains: 0x10000468	T					
In column ✓ Address ✓ Opcode ✓ Mnemonic ✓ Comment	Options Goto Address Case Sensitive Whole Word Regular Expression					
Direction Forward C Backward Scope						
0x1001540c						
Enter an address range to search in.						
Find	Close					

The default instruction to be sought is that of the selected instruction. The initial scope of this wizard is set between the start address and the end address of the loaded executable or shared object. You may change the scope as you like, but it should not exceed the boundary. To choose the direction of searching, select **Forward** or **Backward**. Error and information messages are displayed in the bottom of the dialogue.

5.2.3.3.6Open Source Code

To get the source code of the selected instruction, right-click this instruction and choose **Open Source Code**. You can also use the button in title bar. For more information, please refer to View source code.

5.2.3.3.7Go to callers and callees

Right click the first instruction in this basic block. The menu list shows its callers, their addresses and functions name which they belong to. The caller is defined to be the first instruction of basic block which calls the selected instruction.

Instruction	Table 🗙	33	- 💦 🐚 🛛 i,	G is id PPE 😭	🔲 🚹 🔻 '	- 8
File: exp.c	Function: .exp	Index: 3634 B	B: addr: Ox	1000ffe0 size:	35(140) exec	:: 0
Address	Opcode	Mnemonic	Comment	Freq.	Graph	
0x1000ffc0	0xfc040000	fempu erO,		0		
0x1000ffc4	Oxd861ffe8	stfd f3,-2		0		
0x1000ffc8	0xc8c40088	lfd f6,136		0		
0x1000ffcc	0xc8040080	lfd f0,128		0		
0x1000ffd0	0xc8440078	lfd f2,120		0		
0x1000ffd4	0xc8e40070	1fd f7,112		0		
0x1000ffd8	0xc9640060	lfd f11,96		0		
0x1000ffdc	0x40810090	ble cr0, 0		0	$\overline{\Psi}$	
0x1000ffe0	0xc8640090	1 FA F3 144		0		
0x1000ffe4	Oxc8a40 PPE Show PPO	Help (0		
0x1000ffe8	0xfd211		-	0		
0x1000ffec	Oxc9040 💕 Find			0		
0x1000fff0	Oxc8040 🛐 Open Sou	urce Code		0		
0x1000fff4	0xfc691		_	0		
0x1000fff8	Oxd921i Goto cal	ler 0x1000ff9c.		0		-
0x1000fffc	Oxfc2208fc	finsub 式	_	0		
0x10010000	0xfc44307a	fmadd f2, f		0		
0x10010004	0x8061fff4	lwz r3,=12		0		
0x10010008	Oxd841fff8	stfd f2,-8		0		
0x1001000c	0x5463a016	rlwinm r3,		0		~
	0.0004.00.0					· ·

To get target basic blocks of the selected instruction, right-click the last instruction of a basic block. The menu list will show its target basic blocks, with addresses of their first instructions and functions name which they belong to. If its target is next basic block, it shows **Fall thru** only.

Instruction	Table 🗙	В	- 🍢 🔟 🕴 i_G	is id PPE 🕃	🔲 🚹 🔻 🗖	' 🗆
File: exp.c	Function: .exp	Index: 3637	BB: addr: Ox	100100a8 size	: 3(12) exec:	0
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^
					¥	
0x100100a8	0xc8a40018	lfd f5,24(0		
0x100100ac	0xfc812800	fempu er1,		0		
0х100100Ъ0	0+409-000-	has as 7 9		0	$\overline{\Psi}$	
	PPE Show PPC Help					
0х100100Ъ4	in Show Branch Pro	ofile		0		
0x100100Ъ8				0		
	💕 Find				¥	
0х100100Ъс	🔊 Open Source Cod			0		
0x100100c0	in open bource con	I		0	$\overline{\mathbf{v}}$	
	Goto 0x100100ba					
0x100100c4	Fallthru	3)		0		
0x100100c8	Uxtc2100Za 😽	fadd fl,fl		0		
0x100100cc	0x4e800020	bclr Ox14,		0		
					¥	
0x100100d0	0x4181ffc4	bgt cr0, 0		0	<u>۱</u>	-
0x100100d4	0x4c5df382	cror crBit		0		
0x100100d8	0x4082009c	bne cr0, 0		0	$\overline{\mathbf{Y}}$	
0x100100dc	0xc103000c	lfs f8,12(0		
	0.010.000					

The screen capture above shows that the selected basic block has two target basic blocks. One is the one below, and the other is in the function .main. You can jump to these basic blocks simply by clicking it. You can verify this relationship by referring to graph beside it.

5.2.3.3.8Navigate along with program tree

In the program tree on the left side of the workbench window, a hierarchical organization of loaded executable or shared object is displayed. You can navigate along Instructions Table by selecting objects in program tree, or vice verse.

For example, if you select the first file in the program tree, the first instruction of this file will be highlighted in instructions view, the first basic block of this file will be highlighted in blocks view and the first function of this file will be highlighted in functions view.

The following screen capture shows the selection of the first file in Program Tree.

Program Tree 🗙 Navigator 🗖	- 🗆
🖻 🖃 🛃 🚛 🖡 🏠 🗇 🔿	-
File: crtOmain.s	
🖃 🗁 crtOmain. s	~
🖃 📄start	
<mark></mark> 0 [Ox100001f4]	
<mark></mark> 0 [0x10000210]	
<mark></mark> 0 [0x10000214]	
0 [0x1000021c]	
🖻 🥭 xlisp. c	
🖻 📄 .main	
0 [0x10000240]	
0 [0x10000264]	
0 [0x10000278]	
0 [0x10000284]	
0 [0x100002ac]	
0 [0x100002b8]	
0 [0x100002bc]	
0 [0x100002c0]	
	\sim

The selected instructions in instructions view changes correspondingly.

5.2.3.4Navigate the Blocks View

Blocks View shows the detailed information of loaded executable or shared object in form of basic blocks. To open this view, select **Switch to blocks** in the Instructions Table's title bar. The explanation of functions of some buttons in title bar can be found in Instructions Table .In blocks view, there is an icon Fin the front of the first basic block of a file in loaded executable or shared object. When a new function begins, an icon appears in the front of its first basic block.

The following screen capture shows blocks view of a loaded executable.

Instruction Table	- Basic Blocks 🗙	F +	💱 🔟 ig is io P	PC 🖹 🔲 <u> </u> 🗸 🖓 🗖
File: crtOm	ain.s Function:	Index: O BB: addr:	0x100001c8 size:	11(44) exec: 0
Address	BB's Last Inst	Size	Freq.	Graph 🔨
0x100001c8	bl 0x10000240	11 (44)	0	₽
۲۲ 0x100001£4	beq cr0, 0x10000	5 (20)	0	$\overline{\mathbf{v}}$
0x10000208	lwz r3,0(r8)	2 (8)	0	
0x10000210	bl 0x100052e4	1 (4)	0	⊻ ↓
0x10000214	tw 0x4,r1,r1	2 (8)	0	
0x1000021 c	UNREACHED	3 (12)	0	
0x10000228	UNREACHED	6 (24)	0	
👌.main 0x10000240	bl 0x100006f0	9 (36)	0	+ ↓
0x10000264	bl 0x10000e40	5 (20)	0	↓
0x10000278	bl 0x1000749c	3 (12)	0	
0x10000284	beq cr0, 0x10000	6 (24)	0	₇
0x1000029c	bl 0x100008a8	2 (8)	0	

After adding FDPR-Pro profile information, you can get the frequency of each basic block in color. The denotation of each kind of color can be found in the lower part of the window.

Instruction Table	- Basic Blocks 🗙	3 -	💱 🔟 ig is ip P	PE 📓 🔲 <u> </u> 🗸 🖓 🗖
File: xldmen.c	Function: .mark I	Index: 1016 BB: add	r: 0x10004710 size	: 1(4) exec: 504331
Address	BB's Last Inst	Size	Freq.	Graph 🔨
0x100046ec	b 0x100048a0	1 (4)	43311	\mathbf{a}
0x100046£0	stw r3,64(r1)	4 (16)	99538	
0x10004700	beq cr0, 0x10004	4 (16)	1478017	
0x10004710	ъ 0x100047d4 😽	1 (4)	504331	
0x10004714	bl 0x100041b4	8 (32)	973686	
0x10004734	beq cr0, 0x10004	2 (8)	973686	↓
0x1000473c	b 0x100047d0	16 (64)	868467	↓
0x1000477c	bl 0x100040e4	2 (8)	105219	¥ +
0x10004784	beq cr0, 0x10004	2 (8)	105219	
0x1000478c	b 0x100047d0	16 (64)	786	
0x100047cc	b 0x100047d4	1 (4)	104433	'
0x100047d0	Ъ 0x10004700	1 (4)	869253	↓ <mark>↓</mark> ↓

If you right-click a basic block, a menu list will appear. Typically, there are three sections in this menu. In the first section, you can search for specific basic block or open source code of the selected basic block. In the second section, the menu shows the target of this basic block (**Fall thru** means the next basic block). The last section gives information of all the callers of this basic block. The last two sections may be varied according to the selected basic block.

5.2.3.4.1 Find basic blocks

To find specific basic blocks in this view, press button ^N in title bar. You can also open it by right-clicking any basic block and choosing **Find**.

The following screen capture shows the initial activating of Find wizard.

Instruction	Table	- Basic Blocks 🗙	3 -	ኛ 🐚 i _g i _g i _g i _p P	PE 🖹 🔲 <u> </u> 🔻 🗖 🗋
File:	xlisp.	c Function: .main	n Index: 8 BB: add	lr: 0x10000264 size	: 5(20) exec: 1
Address		BB's Last Inst	Size	Freq.	Graph 🔨
0x10000210		bl 0x100052e4	1 (4)	0	¥ ↓
0x10000214		tw 0x4,r1,r1	2 (8)	0	
0x1000021 c		UNREACHED	3 (12)	0	
0x10000228		UNREACHED	6 (24)	0	
👌. main 0x1000	0240	bl 0x100006f0	9 (36)	1	_+ ↓
0x10000264	💕 Fi	l nd		1	⊋
0x10000278	DP:	en Source Code	}	1	
0x10000284	Go Fai	to Ox10000e40 (.xlbegi Llthru	in))	1	₇
0x1000029c	Go	to caller 0x10000240		0	$ _{\mathcal{V}}$
0x100002a4		bl 0x100004a8	2 (8)	0	$ _{\gamma}$
0x100002ac		bl 0x100052e4	3 (12)	0	▶
0х100002Ъ8		lwz r2,20(r1)	1 (4)	0	

A Find BB wizard appears.

Contains: 0x100046ec In column Address Goto
In column Address Size BB's Last Instruction Direction Case Sensitive Regular Expression Direction Case Sensitive Regular Expression Ox100001c8 - Ox1001540c
 Address Goto Address Size Case Sensitive BB's Last Whole Word Instruction Regular Expression Direction © Forward C Backward Scope 0x100001c8 - 0x1001540c
Size Case Sensitive BB's Last Whole Word Instruction Regular Expression Direction Backward Scope 0x100001c8
BB's Last Whole Word Instruction Regular Expression Direction Backward Scope 0x100001c8
Instruction Regular Expression Direction Forward C Backward Scope Ox100001c8 - Ox1001540c
Direction Forward Scope 0x100001c8 - 0x1001540c
Scope 0x100001c8 - 0x1001540c
0x100001c8 - 0x1001540c
Enter an address range to search in.
Find Close

The default basic block to be sought is that of the selected one. The initial scope of this wizard is set between the start address and the end address of the loaded executable or shared object. You may change the scope as you like, but it should not exceed the boundary. To choose the direction of searching, select **Forward** or **Backward**. Error and information messages are displayed in the bottom of the dialogue.

5.2.3.4.20pen Source Code

To get the source code of the selected basic block, right-click this basic block and choose **Open Source Code**. You can also use the button in title bar. For more information, please refer to View source code.

5.2.3.4.3Go to callers and callees

Right click any basic block in this view. The menu list shows all its callers and their addresses. The address of caller is defined to be that of the first instruction in the basic block which calls the selected basic block. The menu list also displays the information of its target basic blocks, along with addresses of their first instructions and functions name which they belong to. If its target is next basic block, it shows **Fall thru** only. By choosing its callees or callers, you can jump directly to these basic blocks.

In the screen capture below, the selected basic block has two callees and one caller.

Instruction T	able - Basic Blocks 🗙	F -	💱 🔟 ig ig ig P	PE 📓 📰 <u> </u> 🔻 🗖 🗋
File: x	lisp.c Function: .main	a Index: 8 BB: add	r: 0x10000264 size	: 5(20) exec: 1
Address	BB's Last Inst	Size	Freq.	Graph 🔨
0x10000210	bl 0x100052e4	1 (4)	0	¥ ↓
0x10000214	tw 0x4,r1,r1	2 (8)	0	
0x1000021 c	UNREACHED	3 (12)	0	
0x10000228	UNREACHED	6 (24)	0	
👌.main 0x100002	240 bl 0x100006f0	9 (36)	1	-¥- -↓
0x10000264	bl 0x10000e40	5 (20)	1	1
0x10000278	Find Open Source Code	.2)	1	_¥
0x10000284	Goto Ox10000e40 (.xlbegin	(4)	1	₇
0x1000029c	Fallthru	3)	0	
0x100002a4	br ux10000240	2(8)	0	
0x100002ac	bl 0x100052e4	3 (12)	0	*
0x100002b8	lwz r2,20(r1)	1 (4)	0	

You can verify these relationships by referring to graph column on the right side.

5.2.3.5Navigate Functions View

Functions View shows the detailed information of loaded executable or shared object in form of functions. To open this view, select **Switch to functions** in the Instructions Table's title bar. The explanation of functions of some buttons in title bar can be found in Instructions Table .In functions view, there is an icon in the front of the first function of a file in loaded executable or shared object.

The following screen capture shows functions view of a loaded executable.

🚺 Instruction Tab	ole - Functions 🗙		ଅ - 矝 🔞	i _g i _b i _d PPE) 🗆 <u> </u> 🗸 ¬ ¬ E		
File: crtOmain.s Function:							
Address	Function Name	Func's File	#BB's	Freq.	Graph 🔨		
0x100001c8		crtOmain.s	6	0	₽₽		
<mark>}</mark> 0×10000240	.main	xlisp.c	43	0	॒୬ ᠯᢣᠯ᠊ᡇᠯᡇᠯᡇᠯᡇᠯ		
<mark>}</mark> 0×100004₃8	.osfinish	os. c	2	0	<		
0x100004c8	.osfinit	05. C	2	0			
0x100004e8	. oscheck	05. C	2	0			
0x10000508	. osputc	05. C	6	0			
0x100005b0	.osgetc	05. C	6	0			
0x10000654	. osrand	05. C	4	0			
0x100006f0	.osinit	05. C	3	0	↓		
∂ 0×10000748	flsbuf	glink.s	2	0			
<u>-</u> 0x1000076 e	filbuf	glink.s	2	0			

After adding FDPR-Pro profile information, you can get the frequency of each function in color. The denotation of each kind of color can be found in the lower part of workbench window.

Instruction Tal	ole - Functions 🗙		ଅ - 杉 🔞	i _g i _b i _d PPE) 🗆 <u> </u> 🔻 🗖 🗋
		File: crtOmai	in.s Function:		
Address	Function Name	Func's File	#BB's	Freq.	Graph 🔨
0x10003d48	.xlenter	xlsym.c	15	1370	╡ ┝┦┾┦┾┨┰
<u>-</u> 0x10003ec0	.xlminit	xldmem.c	8	0	╷╷╷╷╷╷╷ ┝┦ ┑ ┙
0x10003fc8	. stats	xldmem.c	14	0	
0x100040e4	.livecdr	xldmem.c	12	327492	│ │ │ │ │ ↓ │ ≯ │ ≯ │
0x100041b4	.livecar	xldmem.c	18	312311	 ↓
0x100042bc	. addseg	xldmem.c	10	905	
0x1000443c	.sweep	xldmem.c	26	175268	
0x10004648	. vmark	xldmem.c	5	29917	
0x100046d0	.mark	xldmem.c	25	554462	
0x100048cc	. ge	xldmem.c	8	17191	
0x1000497c	.findmem	xldmem.c	5	97	

If you right-click a function, a menu list will appear. Typically, there are three sections of this menu. In the first section, you can choose to find functions or open source code of the selected function. In the second section, the menu shows the target functions of this one. The third section displays the callers of this function. The last two sections may be varied according to the selected function.

5.2.3.5.1 Find functions

To find specific functions in this view, press button sin title bar. You can also open it by right-clicking any function and choosing **Find**.

The following screen capture shows the initial activating of Find wizard.

Instruction Tal	ole - Functions 🗙		J - 杉 🔘	i _g i _g i _d PPE	ð 🔲 <u></u> 🔻	- 8
		File: xldmen.c	Function: . add	seg		
Address	Function Name	Func's File	# BB's	Freq.	Graph	<u>^</u>
0x10003d48	.xlenter	xlsym.c	15	1370	╡ ┥┥┥┥	
🔁 0x10003ec0	.xlminit	xldmem.c	8	0	→ →	, , ,
0x10003fc8	. stats	xldmem.c	14	0		
0x100040e4	.livecdr	xldmem.c	12	327492	 	*
0x100041b4	.livecar	xldmem.c	18	312311)
0x100042bc	. addseg 🛛 📕	Find	1	05		
0x1000443c	. sweep) Open Source Code		75268		
0x10004648	. vmark —	Goto 0x10005308	(.calloc)	9917		
0x100046d0	.mark	Goto caller 0x10 Goto caller 0x10	012d3c (.xexpand)	54462		
0x100048cc	. ge	xldmem.c	8	17191		
0x1000497c	.findmem	xldmem.c	5	97		~

A Find Function wizard appears.

У Find Function					
Contains: Ox100042bc	•				
In column Address Function Name Function File # BB's	Options Goto Address Case Sensitive Whole Word Regular Expression				
Direction Forward	C Backward				
Scope Ox100001c8 - Ox1001540c Enter an address range to search in.					
Find Close					

The default function to be sought is that of the selected one. The initial scope of this wizard is set between the start address and the end address of the loaded executable or shared object. You may change the scope as you like, but it should not exceed the boundary. To choose the direction of searching, select **Forward** or **Backward**. Error and information messages are displayed in the bottom of the dialogue.

5.2.3.5.20pen Source Code

To get the source code of the selected function, right-click this function and choose **Open Source Code**. You can also use the button in title bar. For more information, please refer to View source code.

5.2.3.5.3Go to callers and callees

Right click any function in this view. The menu list shows all its callers and their addresses. The address of caller is defined to be that of the first instruction in the basic block which calls the selected function. The menu list also displays the information of its target basic blocks, along with addresses of their first instructions and functions name which they belong to. If its target is next basic block, it shows **Fall thru** only. By choosing its callees or callers, you can jump directly to these functions.

Instruction Tal	ble - Functions 🗙		🕲 🔸 🌄 🔘	i _G i _B i _D PPE) 🔲 <u> </u> 🗸 ¬ ¬ 🗆)			
File: os.c Function: .osrand								
Address	Function Name	Func's File	# BB's	Freq.	Graph 🔼			
0x100004c8	. osfinit	05. C	2	0				
0x100004e8	. oscheck	os. c	2	0	۲ ۲			
0x10000508	. ospute	os. c	6	0				
0x100005b0	.osgetc	05. C	6	0	$ _{\mathcal{V}}$			
0x10000654	• 😽 Find			0	 ↓ ↓ ↓ ↓ ↓			
0x100006f0	. c 📓 Open Source	: Code		0	ار			
0 x10000748	Goto 0x1000 	0790 (.rand) 07b4 (_itrunc.s)		0	<u>ا</u>			
<u>-</u> 0x1000076 c	·- Goto caller	0x1000e720 (. una	(y)	0				
avi 10000790	. rand	glink. s	2	0	*			
∂0x100007b4	_i trunc. s	_i trunc. s	11	0				
<mark>}</mark> 0×10000884	.fp_raise_xcp	glink. s	2	0	∉ ∨			

The following screen capture below, the selected function has one callee and six callers.

You can verify these relationships by referring to graph column on the right side of view.

5.2.4Instruction Properties Analysis

The following tasks allow you to analyze instruction properties within Code Analyzer:

5.2.4.1View Branch Profile

Branch Profile table shows the detailed information of targets of the instruction in the end of an instruction group. This information is available only after loading a profile file.

To get the information, please follow these steps: load an executable, add profiling information, be sure to open the Instructions Table, set instructions view in the Instructions Table, select the last instruction of an instruction group and right-click it and then choose **Show Branch Profile**.

Instruction 1	able 🗙		ይ - 杉 🔘	$i_{\scriptscriptstyle G}$ $i_{\scriptscriptstyle B}$ $i_{\scriptscriptstyle D}$ PPC	3 🗖 🔟 🍸 🗖	
File: d	ertOmain.s Funct	ion: Index: O	BB: addr: Ox100	001c8 size: 11	(44) exec: 1	
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^
0x100001e8	0x80e20004	lwz r7,4(r2)	-> errno 0x2	1		
0x100001ec	0x91070000	stw r8,0(r7)		1		
$0 \times 100001 f0$	0+48000051	L1_0∞10000240	.main	1	$\overline{\mathbf{Y}}$	
	PPE Show PPC Hel	Lp 🛛				
0x100001f4	i _a Shew Branch	Profile		0		
0x100001f8	(<u> </u>			0		
0x100001fc	(💕 Find			0		
0x10000200	🛛 🗿 Open Source	Code ···		0		
0x10000204	(·····		0	Įγ	
	Goto 0x10000)240 (.main)				
0x10000208	(Fallthru	þ		0		
0x1000020c	ΰχουρουου	1wz r3,U(r8)		0		
					l 🕹	
0x10000210	0x480050d5	bl 0x100052e4	.exit	0	ļγ	
0x10000214	0x80410014	1wz r2,20(r1)		0		
0x10000218	0x7c810808	tw Ox4,r1,r1		0		
	强 TRACE BAC					
0x1000021c	0x00000000			0		
0x10000220	0x000c0000			0		
0x10000224	0x00000000			0		
	强 UNREACHED					
0x10000228	0x0000000			0		*

You can also press the button i_{II} in the Instructions Table's title bar.

Instruction	Table 🗙		B 🔹 🌄 🔞	i _g i b i _d PPE	🗟 🗔 <u>II</u> 🎽		
File:	File: crtOmain.s Function: Index: O BB: addr: Ox100001c8 Show branch profile for selected i						
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^	
0x100001e8	0x80e20004	lwz r7,4(r2)	-> errno 0x2	1		-	
0x100001ec	0x91070000	stw r8,0(r7)		1			
0x100001f0	0x48000051	bl 0x10000240	.main	1	₽		
0x100001f4	0x6000000	ori r0,r0,0x0		0			
0x100001f8	0x80f20008	lwz r7,8(r18)		0			
0x100001fc	0x80e70000	lwz r7,0(r7)		0			
0x10000200	0x2c070000	empi er0,0x0		0			
0x10000204	0x4182000c	beq crO, Ox1		0	l⊋		
0x10000208	0x8112000c	lwz r8,12(r18)		0			
0x1000020c	0x80680000	lwz r3,0(r8)		0			
					_ *		
0x10000210	0x480050d5	bl 0x100052e4	.exit	0	I]⊋		
0x10000214	0x80410014	lwz r2,20(r1)		0			
0x10000218	0x7c810808	tw 0x4, r1, r1		0			
	🔝 TRACE BAC						
0x1000021 c	0x00000000			0			
0x10000220	0x000c0000			0			
0x10000224	0x00000000			0			
	🔝 UNREACHED						
0x10000228	0x00000000			0		~	

Then branch profile tab of instruction properties will appear in the right bottom of the workbench window. It displays the addresses (including function's name) and counts of the target basic blocks of the selected instruction group.

📅 Instruction Properties 🗙 🔪	🔄 🔻 🗆 🗖
Branch Profile Value Profile	Dispatch Info
To Address	Count
Ox1000002c8 (threads_init)	2

To simultaneously display branch profile information while scrolling along Instructions Table, press in Instruction Properties view's title bar. If you select an instruction within an instruction group, branch profile will show following information:

🚮 Instruction Properties 🛛	🔄 🔄 👻	
Branch Profile Value Profile Dispatch Info		
Not a branch instruction. Address: 0x100000280 Executes:	2	<

5.2.4.2View Dispatch Information

In Power5 or Power6 architecture, instructions are tracked in groups of one to five instructions rather than as individual instructions. Groups are formed that contain up to five internal instructions, each occupying an internal instruction slot (numbered 0 through 4). Each internal instruction slot in a group feeds separate issue queues for the floating-point units, the branch execution unit, the CR execution unit, the logical CR execution unit, the fixed-point execution units and the load/store execution units. With profile information, CodeAnalyzer can display this information in Dispatch information tab in Instruction Properties view.

To get dispatch information of an executable, please follow these steps: Load an executable, Add the profiling information, open the Instruction Table, Set the instructions view and press () in the Code Analyzer toolbar.

Select the kind of Power architecture your executable is run on in the following wizard.

🚝 Question		×
Group by rules of:		
	Power5 Power6	

In the Instruction Table view, select an instruction, right-click and choose Show Dispatch Info. The architecture you have chosen in the previous step will display alongside the menu item.

	Instruction Tab	le X		ቌ • 漀	🔞 i _g i _g	i, PPE 😭	□ 1	
	File: crt0	ain.s Function:	Index:	O BB: addr:	0x100001c	8 size: 11(4	14) exec: 1	
	Address	Opcode	Mnemoni	c Comme	at Fr	req.	Graph	^
	0x100001 d0	0x39000000	li r8,0		1		i	
Ĺ	0x100001 d4	0x91070000	stw r8,0)(r7)	1			
Г	0x100001d8	0x81320000	lwz r9,0) (r	1			
Í	0x100001dc	0x90690000	stw r3,0)(r9)	1			
	0x100001e0	0x81320004	1wz r9,4	4 (r	1			
Ĺ	0x100001e4	0x90890000	stw r4,0)(r9)	1			
Г	0x100001e8	0x80e20004	lwz r7,4	$t(r2) \rightarrow er$	rno 0 1			
	0x100001ec	0x91070000	stw r8,0)(r7)	1			
L	0x100001 f0	0v48000051	ы 0х100)00main	1		\mathbf{F}	
	PPE Show PPC	C Help						
Г	Ox: in Show Bra	anch Profile	r0, 1	r0,	0			
	Ox:	motal Tata (Roman5)	r7,8	3 (r	0			
	Ox: 10 Show Dis	spacen into (Lowers)	r7,0)(r7)	0			
	0x: 🌄 Find		i cr(), 0	0			
L	0x:		cr0,	0	0		ΙΨ	
	E Upen Sou	urce Lode						
Г	Ox: Goto Ox1	.0000240 (.main)	r8, 1	12 (0			
	Ox: Fallthru	1	r3,0)(r8)	0			
		-					4	
L	0x10000210	0x480050d5	bl 0x100	005exit	0		Ψ	
								~
	0.40000044	0.00440044					1 1 1	

5.2.4.3View Value Profile

Value Profile is used to show the resources, their values and counts of specific instruction in the loaded executable. To collect these values, please follow these steps: load an executable, open the Instructions Table view and set instructions view and then set to collect resources of some instruction by right-clicking each of them and choose Collect Value Profiling.

Instructio	n Table 🗙		ቌ 🔸 💦 🔘	$i_{\scriptscriptstyle G}$ $i_{\scriptscriptstyle B}$ $i_{\scriptscriptstyle D}$ P	PC 🛐 📰 <u> </u> 🍸	
File	crtOmain.s Function	1: Index: O BI	8: addr: 0x1000)01c8 size:	11(44) exec: O	
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^
≥FILE crt.	📔 FUNCTION					_
0x100001c8	0x82420000	lwz r18,0(r2)	start,	0		
0x100001cc	0x80f20008	lwz r7,8(r18)		0		
0x100001d0	0x39000000	li r8,0		0		
0x100001d4	0x91070000	stw r8,0(r7)		0		
0x100001d8	0x81320000	1wz r9,0(r18)		0		
0x100001de	0x90690000	stw r3,0(r9)		0		
0x100001e0	PPE Show PPC Help	r9,4(r18)		0		
0x100001e4	Collect Value Profili	ng r4,0(r9)		0		
0x100001e8		r7,4(r2)	-> errno Ox	0		
0x100001ec	💖 Find	r8,0(r7)		0		
0x100001f0	🗃 Open Source Code	0x10000240	.main	0		_
0x100001f4	0x6000000	ori r0,r0,0x0		0		
0x100001f8	0x80f20008	lwz r7,8(r18)		0		
0x100001fc	0x80e70000	lwz r7,0(r7)		0		
0x10000200	0x2c070000	cmpi cr0,0x		0		
0x10000204	0x4182000c	beq crO, Ox		0	$ _{\mathcal{V}}$	
0x10000208	0x8112000c	lwz r8, 12 (r		0		
0x1000020c	0x80680000	lwz r3,0(r8)		0		~

Choose resources and their types in the following wizard.

🚝 Collect Value Profiling For 0x10000020c 🛛 🛛 🔀								
Resources	Туре		Resource Name	Value type	_			
r0 ^ r1 r2 r3	⊙ Absolute	Add >>						
r4 r5 r6 r7	O Symbol	Remove <<						
			OK	Cancel				

Press \mathbf{Ok} . The following screen capture shows an icon may appear on the selected instructions after the above steps.

Instruction T	able 🗙		ይ - ಶ 🐌	i _g i _b i _d PPE) 🔲 <u> </u> 🗸 ר	• 🗆
File: d	ertOmain.s Function	: Index: OBH	: addr: 0x1000	01c8 size: 11(44) exec: O	
Address	Opcode	Mnemonic	Comment	Freq.	Graph	^
≥FILE crt	FUNCTION					-
0x100001c8	0x82420000	lwz r18,0(r2)		0		
0x100001cc	0x80f20008	lwz r7,8(r18)		0		
0x100001d0	0x39000000	li r8,0		0		
0x100001d4	0x91070000	stw r8,0(r7)		0		
0x100001d8	0x81320000	lwz r9,0(r18)		0		
0x100001dc	🔍 0x90690000 📐	stw r3,0(r9)		0		
0x100001e0	0x81320004 V	lwz r9,4(r18)		0		
0x100001e4	0x90890000	stw r4,0(r9)		0		
0x100001e8	0x80e20004	lwz r7,4(r2)	-> errno Ox	0		
0x100001ec	0x91070000	stw r8,0(r7)		0		
0x100001f0	0x48000051	bl 0x10000240	.main	0	$\overline{\mathbf{v}}$	
0x100001f4	0x6000000	ori r0,r0,0x0		0		
0x100001f8	0x80f20008	lwz r7,8(r18)		0		
0x100001fc	0x80e70000	lwz r7,0(r7)		0		
0x10000200	0x2c070000	cmpi cr0,0x		0		
0x10000204	0x4182000c	beq crO, Ox		0	↓	
0x10000208	0x8112000c	lwz r8,12(r		0		
0x1000020c	0x80680000	lwz r3,0(r8)		0		~
					l ale	

Press 📾 in the CodeAnalyzer toolbar to run instrumentation.

Press to write the instrumented file to disk. If you are running on a windows, please make sure to set the output of profile-file value to ./<filename>

Upload the instrumented executable and the profile file you have created to an AIX machine. Make sure to put them under the same directory.

Run the instrumented executable with some training data. It will write information of collected value to your original profile file. You can get relevant training data from SPEC2000.

Copy the profile file created back to the windows.

Load the original executable in CodeAnalyzer again.

Add profile file you've created in --profile--file.

The following screen capture shows that the original grey icon may change to green icon in the front of the selected instruction after doing the above steps.

Instruction 1	able 🗙		ቌ 🔹 💦 🐚 🛛	i _g i _b i _d ppe 🗑) 🔳 🚹 🔻 🗖 🗖
File: d	ertOmain.s Function:	Index: O BB	: addr: 0x1000	01c8 size: 11(4	44) exec: 1
Address	Opcode	Mnemonic	Comment	Freq.	Graph 🗛
≥FILE crt	FUNCTION				
0x100001c8	0x82420000	lwz r18,0(r2)	start,	1	
0x100001cc	0x80f20008	lwz r7,8(r18)		1	
0x100001d0	0x39000000	li r8,0		1	
0x100001d4	0x91070000	stw r8,0(r7)		1	
$0 \times 100001 d8$	🔍 0x81320000	1wz r9,0(r18)		1	
0x100001dc	0x90690000	stw r3,0(r9)		1	
0x100001e0	0x81320004	1wz r9,4(r18)		1	
0x100001e4	0x90890000	stw r4,0(r9)		1	
0x100001e8	🔍 0x80e20004	lwz r7,4(r2)	-> errno 0x	1	
0x100001ec	0x91070000	stw r8,0(r7)		1	
0x100001f0	0x48000051	Ъl 0x10000240	.main	1	₽
0x100001f4	0x60000000	ori r0,r0,0x0		0	
0x100001f8	0x80f20008	lwz r7,8(r18)		0	
0x100001fc	0x80e70000	lwz r7,0(r7)		0	
0x10000200	0x2c070000	cmpi cr0,0x		0	
0x10000204	0x4182000c	beq crO, Ox		0	l₽
0x10000208	0x8112000c	lwz r8,12(r		0	
0x1000020c	0x80680000	lwz r3,0(r8)		0	
					l ale

To view the value of collected resources, right-click the marked instruction and choose **Show Value Profile**. The resources value will be displayed in Value Profile table. You can also press in Instructions Table view's title bar.

📊 Instruction Properties 🛛 🧏 🔻 🗖 🗖						
Branch Profile	Value Profile Dispatch Info					
Resource Name		Value		Count		
r18		53687195	2	1		
r9		-5590387	37	1		
J						

To simultaneously display value profile information while scrolling along Instructions Table, press 🛸

in Instruction Properties view's title bar. If you select an instruction without green icon, value profile will show as follows:

Branch Profile			
	Value Profile	Dispatch Info	
Sony	r, No informa for this in	tion was collected struction	<

5.2.4.4Collect Comments

CodeAnalyzer can display comments generated by FDPR-Pro engine. Different type of files have different comments for collection. So far there are three sources of comments: Power 5, Power 6 and general comments. All the general comments are dependent on profile information. To view there comments, you need to collect comments first. Please follow the steps below:

Load an executable

Add profile information of this executable

Be sure to open Instructions Table

Select File - > Code Analyzer - > Collect Hazard Info or press button A to choose type of comments to collect

Press button 🔤 to open Comments View

CAConsole 🧟 Comments 🗙			
💠 🌲 Description	🗢 File	Function	🗢 Address
\Lambda Load instruction accesses the same addr	exp. c	. exp	0x100100ec
\Lambda Load instruction accesses the same addr	exp. c	. exp	0x10010120
\Lambda Load instruction accesses the same addr	exp. c	. exp	0x10010184
\Lambda Load instruction accesses the same addr	pow. c	.expinner2	0x10010380
🚵 Load instruction accesses the same addr	pow.c	. expinner2	0x10010494
\Lambda Load instruction accesses the same addr	pow. c	.expinner2	0x100104c8
\Lambda Load instruction accesses the same addr	pow. c	.expinner2	0x1001052c
\Lambda Load instruction accesses the same addr	tan. c	. tan	0x1000fbc4
\Lambda Load instruction accesses the same addr	tan. c	. tan	Ox1000fce4
\Lambda Load instruction accesses the same addr	xlmath.c	. checkfneg	0x1000edc8
\Lambda Load instruction accesses the same addr	xlmath. c	. checkfzero	0x1000ee34

Press button 🚰 or 🎦 in tool bar to navigate each comment in Instruction table

Press button to display the currently collected comments statistics

Please notice that if you have restricted grouping to some architecture, the comments view may remove inappropriate comments from it.

5.2.5Statistic Analysis

Code Analyzer provides users with a graphical display of statistics gathered on the loaded executable based on different degrees of granularity. There are three perspectives of analysis: file, function and instruction mix. They are shown in the form of tab in Statistics view. All the graphs are drawn to scale in cylinder. Each column is colored according to its frequency heat. The Statistics view normally displays top (hottest) files, functions or instructions. In the upper level of each tab, there is filter button. Only items that pass the average threshold you set in filter value will be displayed. Therefore, if you enter 0% and press **refresh**, the view will show all the files. And if you enter 100% and press **refresh**, a single column will be displayed, representing the hottest execution unit.

To open these views, you need to load an executable and add profile information first. Then press the corresponding buttons in Code Analyzer's toolbar. You can also open it from **File -> Code Analyzer -> Statistics**.



When your cursor stops in a column in the graph for a while, there is reference information like this:

5.2.5.1View File Heat Graph

To open file heat graph, press button Kin CodeAnalyzer toolbar or choose **File -> CodeAnalyzer -> Statistics -> Files Heat**. You can also select the tab within Statistics view .

After that, load an executable and add necessary profile information.

The following screen capture shows a typical file heat graph of loaded executable.



The x-ordinate shows the names of files in the loaded executable. The y-ordinate denotes execute count distribution. Each column of the graph refers to a file in the loaded executable. Their color shows how frequently they are called. You can set graph options to customize the view. There are three methods in the average method box: simple average, weighted average and highest function value. By inputting the minimum percentage in the latter box and press **Refresh**, you can get the count distribution of those files whose percentages calculated by average method are above this value.

For example, if you use the highest function value as average method and try to filter 25% files, the graph options should be set as follows:

-Graph Options	
Average Method: Highest Function Value 🗸 Filter files under (%): 25	Refresh

Press **Refresh**. Then the files whose highest function value percentages are at least 25% of the maximum value will be listed in the graph as follows:



5.2.5.2View Function Heat Graph

To get function heat graph, you can press button *in CodeAnalyzer toolbar or choose* **File -> CodeAnalyzer -> Statistics -> Functions Heat** . You can also select the tab in Statistics view .

After that, load an executable and add necessary profile information.

The following screen capture shows a typical function heat graph of loaded executable.



The x-ordinate shows the names of functions in the loaded executable. The y-ordinate denotes execute count distribution. Each column of the graph refers to a function in the loaded executable. Their color shows how frequently they are called. You can set graph options to customize the view. There are four average methods: simple average, weighted average, highest BB value and prolog value. By inputting the minimum percentage in the latter box and press **Refresh**, you can get the count distribution of those functions whose percentages calculated by average method are above this value.

For example, if you use the highest BB value as average method and try to filter 25% files, the graph options should be set as follows:

-Graph Options		
Average Method: Highest H	B Value - Filter functions under (%): 10.0	Refresh

Press **Refresh**. Then the functions whose highest BB value percentages are at least 25% of the maximum value will be listed in the graph as follows:


5.2.5.3View Instruction Graph

To open instruction graph, press button Mixin CodeAnalyzer toolbar or choose File -> CodeAnalyzer -> Statistics -> Instruction Mix. You can also select the tab in Statistics view . There are two modes of instruction mix graph: count and percentage. You can press the button Show Executions or Show Count to switch between them.

The following screen capture shows a typical instruction count graph of loaded executable.



The x-ordinate shows the names of instructions in the loaded executable. The y-ordinate denotes count distribution of instructions. Each column of the graph refers to an instruction. Their color shows how frequently these instructions appear in the loaded executable.

The following screen capture shows a typical instruction executions graph of loaded executable.



The x-ordinate shows the names of instructions in the loaded executable. The y-ordinate denotes executions distribution. Each column of the graph refers to an instruction. Their color shows how frequently these instructions are executed in the loaded executable.

5.2.5.4View Comments Graph

To get comments graph, be sure to collect comments first.

Then press button in toolbar or choose File -> CodeAnalyzer -> Statistics -> Comments . You can also select its tab in Statistics view directly. There are two options of graph: show graph for comment and show graph for function. In graph for comments, you can select type of comments to display.



In the above graph, the x-ordinate shows the names of functions that have comments of **Load After Store** in power 5. The yordinate denotes number of this comment for each function. Each column of the graph refers to a function.

Instruction Table	😹 Statistics 🗙				- 8
Graph Options - C Show graph © Show graph	for comment for function -	checkfneg 💌	Filter values	: under (%):	Refresh
		Comments For F	unction		
1.0- 0.8- 0.6- 0.4- 0.4- 0.2- 0.2-					
0 1		Load After Store			
		Comments Nar	nes		
File Heat Graph	Function Heat Graph	Instruction Mix Graph	Comments Graph		

In graph for function, you can select any of functions which have comments and display the number of different comments it contains. In the above graph, the x-ordinate shows the name of comments we try to collect in the first step. The y-ordinate denotes number of this comments in **.checkfneg** function. You can filter the value of columns by pressing **Refresh**.

5.3Pipeline Analyzer

Pipeline Analyzer is a port of the <u>IBM Performance Simulator for Linux on POWER™</u>, another alphaWorks technology. Pipeline joins the VPA toolkit to provide VPA users with the means of examining how code is executed on various IBM POWER processors. Pipeline Analyzer displays the pipeline execution of instruction traces generated by a POWER series processor. It does so by providing a scroll view and a resource view of the instruction execution.

You can also find the Pipeline Analyzer User Guide from within VPA. Select Help - Help Contents within VPA. To get context sensitive help, press F1 for Windows and AIX or press Ctrl+F1 for Linux.

5.3.1Load an existing pipeline file

The <u>IBM Performance Simulator for Linux on POWER™</u> project has directions for capturing an instruction trace and generating Pipeline data files.

Once you have made a run and generated a .pipe and .config file you can use the Pipeline Analyzer to look at them. When you start Visual Performance Analyzer, the default perspective is Profile Analyzer Perspective. To open Pipeline Analyzer, choose **Window -> Open Perspective -> Other -> Pipeline Analyzer**.

😼 Pipeline Analyzer - Visual Performance Analyzer R <u>F</u>ile <u>E</u>dit Se<u>a</u>rch <u>T</u>ools <u>W</u>indow <u>H</u>elp 📓 📊 🔯 🛛 🍫 🗌 🖋 🔍 ९ ९ २ म् 💽 🕎 🗁 🗆 🗖 - -🔉 Pipeline View 🗙 General Offset Cycles: Base: Lines: Cursor: Offset: Divider: Event Message ^ × < >

The following screen capture shows the initial layout of Pipeline Analyzer.

Open File			<u>?×</u>
Look in:	🗀 sample_profile	• 🔁 📥 💽	# #
My Recent Documents Desktop My Documents My Computer	 branch_flush.config branch_flush.pipe nimbus.resource_mode.config nimbus.resource_mode_copy.config nimbus.resource_mode_copy.pipe resource-mode.config resource-mode_copy.config resource-mode_copy.pipe resource-mode_copy.pipe scroll_complex.pipe scroll-mode.config scroll-mode.config scroll-mode.config scroll-mode.config 	scroll-mode_copy.pipe	
My Network	File name: scroll-mode.pipe		Open
Places	Files of type:		Cancel

Choose File -> Open File_, and in Open File dialog select .pipe file with scroll mode information for inspection.

Please note if your .config file has different name as .pipe file, a second dialog for corresponding .config file will turn up for you to choose.

Next, the Pipeline Analyzer Perspective loads the data of .pipe file. A scroll editor is opened automatically. The general information of this .pipe file is displayed in the panel of Pipe View. The following screen capture shows the data loading of this file.



🚳 Pipeline Analyzer - nimbus.resource_mode -	Visual Performance Analyzer	🗖 🖬 🚬
<u>F</u> ile <u>E</u> dit P <u>i</u> peline Se <u>a</u> rch <u>T</u> ools <u>M</u> indow <u>H</u> elp		
] 🚮 📊 😰] 🎭] 🖉		
Y Pipeline View 😒		🔍 ् ् + 🖸 🕎 🕝 🖓 🗖
Cycles: 508 Base: 225:20		
Lines: 11648 Cursor: 36:2944		
Divider: 1 Offset: -189:2924	And the second se	
Event Message	2445	
Dprefetch(type:4, value:129)		
		× 1
		>
🔯 nimbus. resource_mode 🗙 🛛 Scroll-mode		- 8
QQQ J 🖩 🖳 💻		
	an source Name	
	AT_SLA_id[0][30] ;	
	esource: AT_SLA_id[1][0] AT_SLA_id[1][1] AT_SLA_id[1][1]	
2950		
	AT_SLA_[d[1][9] AT_SLA_id[1][9]	
F	AT_SLA_id[1][1]	
	AT_SLA_id[1][13]	
	AT_SLA_id[1][15]	
2960	AT_SLA_id[1][16]	
	AT_SLA_id[1][18]	
	AT_SLA_id[1][19] AT_SLA_id[1][20]	
	AT_SLA_id[1][21]	
	AT_SLA_id[1][22] AT_SLA_id[1][23]	
	AT_SLA_id[1][24]	
7970	AT_SLA_Id[1][25] AT_SLA_id[1][26]	
······································	AT_SLA_id[1][27]	
	AT_SLA_[d[1][28] AT_SLA_id[1][29]	
· · · · · · · · · · · · · · · · · ·		
2980		

To open .pipe file with resource mode information, choose **File -> Open File_**. A resource editor is opened as follows:

5.3.2Navigating the scroll pipe view

Each time Pipeline Analyzer Perspective is opened, a pipe view turns up in the left part of perspective. It shows the detailed information of the currently active editor.

To open Pipe View manually, choose Window -> Show View -> Pipeline Category -> Pipe View. Note every perspective contains only one pipe view.

To view pipeline file containing scroll mode information, select File -> Open File and choose corresponding file. A scroll editor opens in the Pipeline Analyzer Editor and Pipe View display its information at the same time.

5.3.2.1Using the overview graph

In the overview graph, the green box indicates the boundary of data displayed in the currently active scroll editor. To display data elsewhere, click its location in the graph. You will see that the green box moves to where you just click and scroll editor displays the data in detail.



5.3.2.2Zoom in or out

To zoom in or out this graph, press buttons in tool bar. To fit both width and height of the graph to overview panel, choose in tool bar. Note that no scroll bars appear in overview panel after this operation.

5.3.2.3The event message and offset panel

The Event Message and Offset panel displays the denotation of instruction event which your mouse points at in the Scroll Editor. The following screen shot shows the change of this information when the mouse moves from symbol D to symbol M in the same line of table.



If two more events in an instruction execution occur at the same time cycle, their corresponding symbol will be highlighted and all the events be listed in Event Message panel. The following screen shot shows the event message of a highlighted symbol "F" in the above picture.

Event Message F:fetch initiated F:fetch initiated

If there are too many event messages to display, you can resize this panel.

To scroll simultaneously with the currently active editor, press Pin tool bar. The following screen shot shows this effect.



Please note that this function ensures that the green box is always within the overview graph of Pipe View.

5.3.3Navigating the resource view

To view pipeline file containing resource mode information, select **File -> Open File** and choose corresponding file. A resource editor opens in the Pipeline Analyzer Editor and Pipe View display its information at the same time.



The following screen shot shows the appearance of resource editor after a pipeline file is loaded.

In resource editor, a side bar named **Resource Name** in the right lists all the resources recorded in pipeline file. Each line of table in the left shows the usage distribution of this resource during time period. Each symbol in the table means an instruction event. Its denotation is shown in Event Message panel of Pipe View. If more than two events struggle for one resource at the same time, its symbol in the table turns red automatically. In this case, all the event messages are listed in Event Message panel of Pipe View, you can see that this file has a total of 507 cycles and 11648 lines. The current time divider is 1, which means each symbol in the table indicates one time cycle.

The two red sliders in the table are named slider bar. It focuses on the cell which your mouse points at. Its ordinate is shown as Cursor value of Offset panel in Pipe View. The grey sliders in the table are named base axises. It focuses on the latest click of your left mouse. Its ordinate is shown as Base value of Offset panel in Pipe View. The distance between slider bar and base axises is calculated and shown as Offset value of Offset panel in Pipe View.

To navigate the editor, you can press left, ri0067ht, up, down keys or 'H', 'J', 'K', 'L' keys.

5.3.3.1Zoom in or out

To zoom in or out the table in scroll editor, press buttons in tool bar or select it in Pipeline Menu.

5.3.3.2Show Dots

To show dots in the table, press in tool bar or select it in Pipeline Menu. The following picture shows a table with dots.



5.3.3.3Show Hover

To show the denotation of each symbol in the table, press in tool bar or select it in Pipeline Menu. While your mouse points at a line in the table for a while, a label which explains this line of resource usage turns up. The following screen shot shows the symbol message for a confile point in resource editor.

N ni	imbus.resource_r	node 🗙					- 0
E	२ २ - 🔳	88 🖻 💻					
	40	50	60		70	Resource Name	
						CLQ_id[30] CLQ_id[31] DQ_id[0][0] DQ_id[0][1] DQ_id[0][2]	
- - _ 390		esource: Assage:	DQ_id[O][3] Dprefetch(type:4, Dprefetch(type:4, Dprefetch(type:4,	value:370) value:339) value:355)		<u>PQ_id[0][3]</u> DQ_id[0][4] DQ_id[0][5] DQ_id[0][6] DQ_id[0][7]	
-	· · · · · · · · · · · · · · · · · · ·			· · · ·	× × × × ×	DQ_id[0][8]	

5.3.3.4Show Slider

To show slider bar while scrolling around the table, press 🛱 in tool bar or select it in Pipeline Menu.

5.3.3.5Hide Unvisited Resources

When the usage distribution of pipeline file is distributed loosely, you can hide those unvisited resources. To do this function,

press Prin tool bar or select it in Pipeline Menu.

The following screen capture shows the condensed resource table for the above opened file.

🚱 Pipeline Analyzer - nimbus.resource_mode - Visual Performance Analyzer	
<u>F</u> ile <u>E</u> dit Pipeline Search <u>T</u> ools <u>M</u> indow <u>H</u> elp	
Pipeline View 🛛 🔍 🔍 🗸 🖓	
General Offset	
Lines: 877 Cursor: 65:57	
Divider: 1 Offset: 9:-330	
Event Message	
Dprefetch(type:4, value:404)	<u> </u>
NSM_d[0][0][1] source: NSM_d[0][0][1] NSM id[0][0][2]	
NSM idf0[[0][3]	

5.3.3.6Change Color for symbol

To change color for each symbol in the table, choose **Pipeline -> Settings... -> Trace.**

Preferences					
Trace	Trace Identifier Unknown TWprefetch TWdemand Sync Store Store StcxFail Stcx SoftTWprefetch SoftDprefetch (Symbol (Color	Description Unknown TWprefetch TWdemand Sync Store StcxFail Stcx SoftTWprefetch SoftDprefetch Defaults Apply	
				OK Cancel	

For example, if we set Dprefetch event in the opened resource file in the first screen shot to be yellow, the resource editor changes as follows:



Please note, the red lines in this view means conflict for resource use. This is a system-defined color.

5.3.3.7Change Time Divider

To change the number of time cycles for each symbol in the table, choose **Pipeline -> Change Time Divider**.



The default value of this box is the current time divider. The following screen capture shows the resource editor after setting time divider to be three.



5.4Counter Analyzer

The Counter Analyzer tool is a common tool to analyze hardware performance counter data among many IBM eServer platforms, which includes systems running on AIX, i5OS, zOS, Linux on POWER, Linux on CELL/B.E..

The Counter Analyzer tool accepts hardware performance counter data in the form of a cross-platform XML file format. The tool uses either build-in hsqldb database engine or external DB2 instance to store the raw performance counter data. The tool provides multiple views to help user identify the data. The views can be divided into two categories: one category is the "table" views, which are basically two-dimension tables displaying data. The data could be raw performance counter values, derived metrics, counter comparison results and so on. Another category is the "plot" views. In these views data are represented by different kind of plots. The data could also be raw performance counter values, derived metrics, and comparison results and so on. Besides these "table" views and "plot" views, there are also some "utility" views to help user configure and customize the tool.

You can also find the Pipeline Analyzer User Guides from the VPA. Select **Help - Help Contents** within VPA. To get context sensitive help, press **F1** for Windows and AIX or press **Ctrl+F1** for Linux.

5.4.1Basic concepts for Counter Analyzer

> Performance Monitoring Counter

Performance monitor counter provides comprehensive reports of events that are critical to performance on IBM systems. It is able to gather critical hardware events, such as the number of misses on all cache levels, the number of floating point instructions executed, the number of instruction loads that cause TLB misses.

> Metrics

Metric is calculated with user-defined formula and event count from performance monitor counter. It's used to provide performance information like CPU utilization rate, million instructions per second. This helps the algorithm designer or programmer identify and eliminate performance bottlenecks.

> CPI Breakdown Model

Cycles per instruction(CPI) is the measurement for analyzing the performance of a workload. CPI is simply defined as the number of processor clocked cycles needed to complete an instruction. It is calculated as CPI = Total Cycles / Number of Instructions Completed. A high CPI value usually implies underutilization of machine resources.

On a POWER5 system, you can break down your workload CPI into individual components, as the POWER5 has several programmable counters available to count events that can calculate the components of CPI and allow you to determine how to improve performance on a given workload.

The following is an instance of CPI breakdown model:

Total cycle <# cycles>	Completion cycles <a:group complete="" cycles=""></a:group>	PowerPC Base co <a1: more<="" one="" or="" th=""><th colspan="6">PPC Base completion cycles One or more PowerPC instructions completed this cycle></th></a1:>	PPC Base completion cycles One or more PowerPC instructions completed this cycle>					
		overhead of cracking/microcoding and grouping restriction <a2:(a)-(a1)></a2:(a)-(a1)>						
	Completion Table empty (GCT empty) cycles	I-cache miss pena <b1></b1>	llty					
		Branch redirection <b2></b2>	ı (branch mispre	ediction) penalty				
		others (Flush pena <b4: (b)-(b1)-(b2:<="" td=""><td colspan="5">others (Flush penalty etc.) <b4: (b)-(b1)-(b2=""></b4:></td></b4:>	others (Flush penalty etc.) <b4: (b)-(b1)-(b2=""></b4:>					
	Completion Stall cycles <c: total-(a)-(b)=""></c:>	Stall by LSU inst <c1></c1>	Stall by reject <c1a></c1a>	Stall by translation (rejected by ERAT miss) <c1a1></c1a1>				
				Other reject <c1a2: (c1a)-(c1a1)=""></c1a2:>				
			Stall by D-cache miss <c1b></c1b>					
			Stall by LSU basic latency, LSU Flush penalty <c1c: (c1)-(c1a)-(c1b)=""></c1c:>					
		Stall by FXU inst <c2></c2>	Stall by any form of DIV/MTSPR/MFSPR inst <c2a></c2a>					
			Stall by FXU basic latency <c2c: (c2)-(c2a)=""></c2c:>					
		Stall by FPU inst <c3></c3>	Stall by any form of FDIV/FSQRT inst <c3a></c3a>					
			Stall by FPU basic latency <c3b: (c3)-(c3a)=""></c3b:>					
		others (Stall by BF <c4: (completion<="" td=""><td>RU/CRU inst , flu Stall cycles)-(C</td><td>ush penalty (except LSU flush), etc.) 1)-(C2)-(C3) ></td></c4:>	RU/CRU inst , flu Stall cycles)-(C	ush penalty (except LSU flush), etc.) 1)-(C2)-(C3) >				

The above table represents a CPI breakdown model where the total cycles of a workload is divided into three components: Completion cycles, Completion Table empty or GCT empty cycles, and Completion Stall cycles. The base completion cycles are the number of cycles that would be needed if grouping was perfect. Otherwise, stalls happen, and they can be attributed to either Completion Table empty or Completion Stall cycles. A Completion Table empty condition occurs when no groups are completing on a given cycle because of either Icache miss or branch misdirection. Meanwhile, the Completion Stall cycles are those stalls caused by any of the following instructions: LSU, FXU, FXU long (all forms of div, mtspr, mfspr), FPU, and FPU long (all forms of fsqrt, fdiv); or by other events such as Dcache miss, Reject, and Reject by translating (ERAT miss).

5.4.2Load an existing counter data file

5.4.2.1 Open Counter Analyzer Perspective

1. When you first start Visual Performance Analyzer after installation, the default perspective is Profile Analyzer. To start Counter Analyzer, choose **Window -> Open Perspective -> Other.**

🚳 Visual Perfo	ormance Analyzer		
File Edit Tools	Window Help		
i 🔝 🚹 🔽 🧕	Open in New Window		
	New Editor		
	Open Perspective 🔹 🕨	Other	
	Show View		
	Customize Perspective		
	Save Perspective As		
	Reset Perspective		
	Close Perspective		
	Close All Perspectives		
	🔁 Working Sets		
	Preferences		

2. In the Select Perspective dialog choose **Counter Analyzer** and click **OK**.



3. The following screen capture shows the initial layout of Counter Analyzer.

Scounter Analyz	er - Visual Performance Analyzer	
File Edit Counter 1	Tools Window Help	
i 📕 📊 📡 🖸 🧮		
DX »1 🗆		Res 🛛 🗖 🗖
₽, 2, - ▽		ame
🗉 🗗 Local Conner		
< D >		
Name		
	Graph 🛛	
	Event Chart	max
	0.001 d	
	t t t t t t t t t t t t t t t t t t t	min
< >	<u> </u>	0

4. You can also open **Counter Analyzer** Perspective simply by clicking **G** in the toolbar.

5.4.2.2Load in counter data file

Choose File -> Open File_, and in Open File dialog select one counter data file with suffix ".pmf".

Open File			? 🗙
Look in:	bpm 🔁	- E 📸 🎟 -	
My Recent Documents Desktop My Documents	<pre>hpmc_1_100tsort.pmf hpmc_1_100tsortlong.pmf hpmc_1_100tsortlongx10.p hpmc_1_sortlongx10.pmf hpmc_1_tsort.pmf hpmc_1_tsort.pmf hpmc_1_tsort_cpu12.pmf hpmc_1_tsortlong.pmf hpmc_1_tsortlong.xml hpmc_1_tsortlong.xml hpmc_1_tsortlongx10.pmf hpmc_1_tsortlongx10.pmf</pre>	<pre>hpmc_cpi_metrics_two40.pmf hpmc_cpi_metrics_two40.pmf hpmc_cpibreakdown_1.pmf hpmc_metrics_1.pmf hpmc_metrics_1.pmf hpmc_metrics_2t.pmf hpmc_onecpu_1_16.pmf hpmc_onecpu_1_16t.pmf hpmccompare_1_2.pmf hpmccompare_1_3.pmf hpmccompare_3_4.pmf one_timeslice.pmf</pre>	
My Computer	hpmc_cpi_metrics.pmf	📼 one_timeslice.pmf.bak 📼 ×1.pmf	
S			>
My Network Places	File name: hpmc_cpib	reakdown_1.pmf	Open
1 1005	Files of type:	▼	Cancel

You can also load in counter data from repositories, which will not be covered here.

5.4.2.3Brief introduction to Counter Analyzer Perspective

After loading in the counter data of .pmf file, the Counter Analyzer Perspective displays the data in its views and editors. Primary information of details, metrics and CPI breakdown is displyed in Counter Editor. Resource statistics information of the file (if available) will be showed in tabular view **Resource Statistics**. View **Graph** illustrates the details, metrics and CPI breakdown in a graphic way.

Scounter Analy	🗟 Counter Analyzer - hpmc_cpibreakdown_1.pmf - Visual Perfor 📰 🗖 🔀						
<u>F</u> ile <u>E</u> dit Counter	Se <u>a</u> rch <u>T</u> ools <u>W</u> indow <u>H</u> elp						
] 🛄 📊 💟 🛄 🕷	ኈ 🖋 🖅 📶 💷 1.0 10⁵ 1						
C 🛛 🔭 🗆 🗖	🕒 hpmc_cpibreakdown 🗙 🎽 🗖 🗖 Resource 🖄 🦳 🗖						
🗗 🗙 🛇 🚷 💆	Events Averag Name						
Local Repos	PM_IPLUS_PPC_CMPL 1,025,910,15 PM_CMPLU_STALL_DCACHE_MISS 99,62 RV_INBLOCK						
	PM_CMPLU_STALL_DIV 10,11 RU_ISRSS(Kbytes*sec)						
	PM_CMPLU_STALL_ERAT_MISS 55,78 RU_IXRSS(Kbytes*sec)						
	PM_CMPLU_STALL_FDIV VS 2,56 RU_MAJFLT -						
	PM_CMPLU_STALL_FPU 2,58 RU_MAXRSS(Kbytes)						
	PM_CMPLU_STALL_FXU 639, 160, 51 RU_MINFLI						
	PM_CMPL0_STALL_ISU 3,802,173,30 RU MSGSND						
	Details Metrics CPI Breakdown						
	Graph 🔀						
Name 🔼							
Basic Repositor Repositor Repositor							
	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 min						

5.4.3Navigate the Counter Analyzer Perspective

The following are tasks that you can perform to navigate around counter data within Counter Analyzer.

5.4.3.10pen Counter Data

There are two ways to open counter data using Counter Analyzer, either from counter data files or from database repository.

5.4.3.1.1 Open Counter Data from PMF Files

1. Choose File -> Open File_, and in Open File dialog select one counter data file with suffix ".pmf".

Open File						? 🗙
Look in:	🗀 hpm			• 🗧 🛍	r 🔝 📩	
My Recent Documents Desktop My Documents	<pre>hpmc_1_100tsd hpmc_1_100tsd hpmc_1_100tsd hpmc_1_sort.pl hpmc_1_sort.pl hpmc_1_sort.pl hpmc_1_tsort.pl hpmc_1_tsort.pl hpmc_1_tsortlod hpmc_1_tsortlod hpmc_1_tsortlod hpmc_1_tsortlod hpmc_1_tsortx; hpmc_3_66_ts hpmc_cpi_metri</pre>	ort.pmf ortlong.pmf ortlongx10.pmf mf ngx10.pmf opu1.pmf opu12.pmf ong.pmf ong.xml ongx10.pmf s.pmf sort.pmf	hpmc_cpi_ hpmc_cpi_ hpmc_cpib hpmc_met hpmc_met hpmc_met hpmc_one hpmc_one hpmccomp hpmccomp hpmccomp hpmccomp hpmccomp hpmccomp one_times	metrics_two40.p metrics_two40.p rics_1.pmf rics_1.pmf rics_21.pmf rics_22.pmf cpu_1_16.pmf cpu_1_16.pmf pare_1_2.pmf pare_1_3.pmf pare_3_4.pmf lice.pmf lice.pmf.bak	omf pmf F	
~~~	i hpmc_cpi_metri	ics_t.pmf	🔤 ×1.pmf			
						>
My Network Places	File name:	hpmc_cpibreak	down_1.pmf		<u> </u>	Open
	Files of type:	× ×			•	Cancel

2. If one counter file is selected, it will be opened in editor, and its resource statistics information (if available) will be showed in tabular view **Resource Statistics**.

hpmc_cpibreakdown_1.	pmf 🖾 🔪			🔲 Resource Statist	ics 🗙	
Events	All Pr	Average	•	Name	Value	Descri
PM_1PLUS_PPC_CMPL	4,103,6	1,025,910,755		RV IDRSS (Kbyte	2519	Averag
PM_CMPLU_STALL_DCACHE	398, 482	99, 620		RU_INBLOCK	0	Number
PM_CMPLU_STALL_DIV	40, 468	10, 117		RU_ISRSS(Kbyte	0	Averag
PM_CMPLU_STALL_ERAT_M	223, 146	55, 786		RU_IXRSS(Kbyte	279	Averag
PM_CMPLU_STALL_FDIV	10, 253	2, 563		RU_MAJFLT	0	Number
PM_CMPLU_STALL_FPU	10, 337	2, 584		RU_MAXRSS(Kbytes)	128	Maximu
PM_CMPLU_STALL_FXU	2,556,6	639, 160, 579		RU_MINFLT	42	Number
PM_CMPLU_STALL_LSU	15,208,	3, 802, 173, 303		RU_MSGRCV	0	Number
PM CMPLU STALL REJECT	78, 693, 427	19,673,357		RU_MSGSND	0	Number
PM CYC	28,232,	7,058,110,230		RU_NIVCSW	1282	Number
PM FPU FULL CYC	0	0		RU_NSIGNALS	U	Number
PM GCT EMPTY CYC	610,196	152, 549, 204		KU_NSWAP	U	Number
PM GCT NOSLOT BR MPRE	91, 576, 737	22, 894, 184		NU_NYCSN	0	Number
PM GCT NOSLOT CYC	4,770,8	1, 192, 714, 981		Sweton Time (co	0.002128	Totol
PM GCT NOSLOT IC MISS	619,205	154,801		Near Time (sec	21 052919	Total
PM GCT NOSLOT SRO FUL	0	0		OSEL_IIMe (Seco	21.052515	IUCAL
PM GRP CMPL	4.103.6	1,025,911,698				
PM GRP IC MISS BR RED	153, 235,	38, 308, 819				
PM GRP MRK	250, 885	62, 721, 463				
PM INST CMPL	5.294.5	1.323.648.100				
PM INST DISP	27.805	6, 951, 449, 461				
DU TODA GUDI	E 014 0	1 200 500 005				
		>				
tails Metrics CPI Br	eakdown			<		

By default, a local repository is provided by VPA tool. You can also connect to remote DB2 repositories. Remote repositories can be created, configured, refreshed, and discarded. Files in these repositories can be opened in

Counter Editor and deleted. Besides, you can import counter data files into and query counter data from these repositories.

### 5.4.3.1.2 Open Counter Data From Connection View

By default, a local repository is provided by VPA tool. You can also connect to remote DB2 repositories. The local connection is a local connection, which you can add Counter Analyzer Support under it, as well as other supports. Remote connection is a DB2 connection. It can be created, configured, refreshed, and deleted.

#### Create/Configure/Refresh/Discard a Connection

### Create Connection

In **Database Connection** view, select **New Connection** in the context menu, you can create a new repository. Local Repository is created by default and can only be configured and refreshed. **Description** view lists basic information of the repository.



#### **Configure Connection**

The following is the Configure Dialog of Connection.

e	
New DB2 Connection Set properties for I	DN B2 connection.
Connection Name:	9. 186. 10. 107
Host Address:	9. 186. 10. 107
Port:	50000
DB Name:	dbmodel
User Name:	db2admin/
	Test Connection
0	< Back Mext > Finish Cancel

### Add Product Support

Right-click on the connection node, choose the product support you want to add to the connection node, a confirm dialog will occur.

You can create a new repository, or select an existing repository from the Existing List.

🖨 Add Counter	Analyzer	×
Add Counter Ana Add Counter Analyz	alyzer er support to this connection.	
Input the new re	epository name, or select an existing one.	
• New:		
O Existing:	COUNTER1 TTT	
0	OK Cancel	

#### **Refresh Connection**

Double click a Connection or click on the + icon to expand the tree.

A **Password Dialog** will pop up when opening the repository for the first time. If the password is saved, you no longer have to re-enter it when Counter Analyzer is closed and restarted later. If the password is not saved, you are required for it every time they start Counter Analyzer.

🖨 Password Dialog	×
Enter Password	
at: 9.186.10.107:50000/hxq@db2admin	
Enter password for this repository:	_
Save Password	
Warning: Saved passwords are stored in local file system in an unsafe way. You can clear the saved passwords from the right-click popup menu.	
OK Cancel	

You can also choose to clear the password on the context menu.



#### **Discard Connection**

You can choose to discard the Connection on the context menu. Be sure that delete all its children first.



### **Discard Support**

You can choose to discard the Product Support on the context menu.



#### > Import File

Right-click on the Support node, select Import File action, then, choose the file you want to import from the file dialog.

⊞ 🔂 Counter Analyz ⊟ 🖧 9. 186. 10. 107	er	
😑 🖸 Counter Ang	N/A	1
🔤 🚺 hpmccom	🖶 Refresh	05-24 10:46
	🔁 Import File	
	🗙 Delete	

#### > Open File in Editor

Right-click on the File node, select Open in Editor, or double-click on the file.



### 5.4.3.2View Counter Data

You can see raw counter data, metrics data and CPI breakdown data in different pages of editor. All these data are retrieved from one counter file.

#### 5.4.3.2.1View Raw Counter Data

Row counter data are shown In Tab **Details**. On the context menu, you can switch display mode here.

😂 x1.pmf 🗙			_	' 🗆
Events	All Processors	Average	Processor O	~
PM_CYC	50, 928, 414, 785	12, 732, 103, 696	235, 352	
PM_DATA_FROM	ent	809	2, 324	
PM_DATA_FROM	/ W / T.: S1 /	284	500	
PM_DATA_FROM	oup/Event/lime 51	235 Lee	477	
PM_DATA_FROM_LINEN Gro	oup/Time Slice/Evo	ent 9	15	
PM_DATA_FROM	38	9	0	
PM_DATA_TABLI 1.0 Sho	ow As Doubles 371	64, 593	108, 138	
PM_DTLB_MISS 10 ⁵ Sh	ow As Science Doul	226	465	
PM_FPU_1FLOP	ow AS Delence Dou	0	0	
PM_FPU_FIN 🗸 1 She	ow As Integer 30	7	0	
PM_FPU_STF		20	0	
PM_FXUO_FIN Fil	lter 057,402	2, 723, 764, 350	11,611	-
PM_FXU_FIN	17, 584, 414, 673	668	20, 894	
PM_INST_CMPL Sel	lect All	2, 371, 028, 511	916, 227, 990	
PM_INST_DISP	ow Selection to Cl	liphoard 625	35, 201	
PM_INST_DISP	py Defection to C.	403 403	44, 204	
PM_ITLB_MISS Fin	nd 3,585	896	289	
PM_LD_MISS_L1	5, 466	1,367	3, 058	×
2	0.000.100.404	770 040 101	007 000 545	
Details Metrics CPI Brea	kdown			

In all tabs, Filter is used to filter processors and events.



Row counter data can be displayed in three modes: **Event**, **Group/Event/Time Slice**, and **Group/Time Slice/Event**. If the counter data has time slice information, all these three modes can be supported. If not, only **Event** mode is enabled, and the other two modes are disabled.

#### "Event" Mode

In this mode, all events and their counter data are listed in the editor. Data here are normalized event count instead of actual data in counter data file.

Events	All Processors	Average	Processor O
PM_CYC	50, 928, 414, 785	12, 732, 103, 696	235, 352
PM_DATA_FROM_L2	3, 234	809	2, 324
PM_DATA_FROM_L2MISS	1,136	284	500
PM_DATA_FROM_L3	939	235	477
PM_DATA_FROM_LMEM	38	9	15
PM_DATA_FROM_RMEM	38	9	0
PM_DATA_TABLEWALK_CYC	258, 371	64, 593	108, 138
PM_DTLB_MISS	905	226	465
PM_FPU_1FLOP	0	0	0
PM_FPU_FIN	30	7	0
PM_FPU_STF	80	20	0
PM_FXUO_FIN	10, 895, 057, 402	2, 723, 764, 350	11,611
PM_FXU_FIN	17, 584, 414, 673	4, 396, 103, 668	20, 894
PM_INST_CMPL	9, 484, 114, 043	2,371,028,511	916, 227, 990
PM_INST_DISP	49, 938, 750, 502	12, 484, 687, 625	35, 201
PM_INST_DISP_ATTEMPT	64, 685, 453, 613	16, 171, 363, 403	44, 204

### "Group/Event/Time Slice" Mode

If the counter data has time slice information, there are two modes to display more. In **Group/Event/Time Slice** mode, the data can be grouped first by group, and then event. Data here are actual event count in counter data file.

Group/Event/TimeSlice)	All Processors	Average	Processo
🖃 pm_L1_tlbmiss (No=43)			
I PM_DATA_TABLEWALK_CYC	34, 604	8,651	14,
Time Slice O (1.0406	9, 796	2, 449	9,
Time Slice 9 (1.0399	13, 419	3, 355	
Time Slice 18 (1.049	5, 462	1,366	2,
Time Slice 27 (1.070	5, 927	1, 482	2,
+ 2 PM_DTLB_MISS	108	27	
	818	204	
+ 4 PM_LD_REF_L1	407, 838, 799	101, 959, 700	89,636,
	1,273,086,580	318, 271, 645	296, 306,
⊕ PM_RUN_CYC	6, 794, 592, 143	1,698,648,036	1,588,506,
<pre> pm_L1_DERAT_miss (No=44) </pre>			
<pre> pm_lsref_tlbmiss (No=130) </pre>			
+ pm_hpmcount1 (No=140)			
+ pm_hpmcount2 (No=141)			

#### "Group/Time Slice/Event" Mode

If the counter data has time slice information, the other mode to display more information is **Group/Time Slice/Event mode**, in which counter data is grouped first by group, and then time slice. Data here are also actual event count in counter data file.

Group/TimeSlice/Event)	All Processors	Average	Processor O
pm_L1_tlbmiss (No=43)			
Time Slice 0 (1.040692)			
1 PM_DATA_TABLEWALK_CYC	9, 796	2, 449	9, 555
2 PM_DTLB_MISS	44	11	44
3 PM_LD_MISS_L1	342	86	337
4 PM_LD_REF_L1	95, 163, 255	23, 79	89, 635, 808
5 PM_INST_CMPL	314, 592, 934	78,64	296, 304, 297
6 PM_RUN_CYC	1,686,561,209	421,6	1,588,489
Time Slice 9 (1.039908)			
<pre> pm_L1_DERAT_miss (No=44) </pre>			
+ pm_dsource2 (No=49)			
<pre> pm_lsref_tlbmiss (No=130) </pre>			
+ pm_hpmcount1 (No=140)			
+ pm_hpmcount2 (No=141)			

#### 5.4.3.2.2View Metrics Data

Metrics data are shown in Tab Metrics.

#### Edit/Load/Save Metrics Variables

In the right part of the editor, all variables associated with the current metrics are listed in a table with their names and values. Left click the **Value** column of each variable, and you can modify its value. On the context menu, users can further choose **Load Variables** to load one variables file, and choose **Save Variables** to save current variables in the editor to one variables file. If one variables file is loaded, all variables' values are updated, which makes all metrics be calculated again. (Only Variable "total_time" is read only and cannot be overwritten.)

😂 x1.pmf 🗙			- 8
Metric	All Processors	^	Name Value
PMD_011_KATE (%) PMD_MIPS (MIPS)	302.389		HPM_AVG_L2_LATENCY
PMD_INST_PER_CYC	0.186		HPM_MEM_LATENCY
PMD_HW_FP_PER_UTIME (M	N/A		MEM_LINE_SIZE
PMD_HW_FP_RATE (M HWflo PMD_FV (M)	0.0		proc_freq total_time 31.364
PMD_FX_PER_CYC (M/s)	0.345		Load Variables
PMD_FP_LD_ST (M) PMD_TNST_PER_EP_LD_ST	0.0		Save Variables
PMD_PRC_INST_DISP_CMPL	18.991		
PMD_DATA_L2 (M) PMD_PRC_L2_ACCESS (%)	0.011		
PMD_L2_TRAF (MBytes)	N/A		
PMD_L2_BDW (MBytes/s) PMD_12_ID_FST_LAT_AVG_(	N/A N/A		
PMD_LD_ST (M)	13, 813. 551		
PMD_INST_PER_LD_ST	0.687	~	
	2		J
Details [Metrics] CPI Breakd	lown		

# **Change Metrics**

You can choose to change metrics file on the context menu, and applies it to the active counter data.

😂 x1.pmf 🗙			
Metric	All Processors	Å7 🔨	Name
PMD_UTI_RATE (%)	N/A		CACHE_LINE_SIZE
PMD_MIPS (MIPS)	302.389		HPM_AVG_L2_LATENCY
PMD_INST_PER_CYC	0.186		HPM_MEM_LATENCY
PMD_HW_FP_PER_CYC	0.0		HPM_TLB_LATENCY
PMD_HW_FP_PER_UTIME_(M	N/A		MEM_LINE_SIZE
PMD_HW_FP_RATE (M H)	<b>1.0</b> Show As Double		z_freq al_time
PMD FX PER CYC (M/s)	10 ⁵ Show As Science	e Double	
PMD_FP_LD_ST (M) PMD INST PER FP LD S	1 Show As Integer		
PMD_PRC_INST_DISP_CM PMD_DATA_L2_(M)	Filter		
PMD PRC L2 ACCESS (S	Change Metrics	<u></u>	
PMD_L2_TRAF (MBytes) PMD_L2_BDW (MBytes/	Select All	1	
PMD_L2_LD_EST_LAT_A ¹	🖺Copy Selection	to Clipboard	1
PMD_LD_ST (M) PMD_INST_PER_LD_ST	Find	_	
<		>	<
Details Metrics CPI Break	down		

The metrics file to be selected can be either external files or built-in files. The change derived metrics dialog is as follows:

🚳 Change Deri	ved Tetrics
Select Another Select another d	Derived Metrics Definition File erived metrics definition file.
• File : ] • Built-in :	 [Plugin]/metadata/POWER3-metrics.xml [Plugin]/metadata/POWER4-metrics.xml [Plugin]/metadata/POWER4-II-metrics.xml [Plugin]/metadata/POWER5-II-metrics.xml [Plugin]/metadata/POWER5-II-metrics.xml [Plugin]/metadata/POWER6-metrics.xml [Plugin]/metadata/PowerPC604-metrics.xml [Plugin]/metadata/PowerPC604e-metrics.xml [Plugin]/metadata/PowerPC970-metrics.xml
	OK Cancel

### 5.4.3.2.3View CPI Breakdown Data

CPI breakdown data are shown in Tab CPI Breakdown.

#### Change CPI Breakdown Model

You can change CPI breakdown model on the context menu, and apply it to the active counter data.

😂 hpmc_cpibreakdown_1.pmf 🗙 🔮 x1.pmf		- 8
CPI Breakdown Model	All Processors	A 🔨
🖃 🕌 TOTAL_CPI	5.332 (100.0%)	5.332 (
CPI_CMPL_CYC	0.775 (14.54%)	0.775 (
🖃 🔡 CPI_GCT_EMPTY	0.115 (2.16%)	0.115
CPI_GCT_EMPTY_TC_UTSS	ററ ഗ്രംഭി	0.0
🔛 CPI_GCT_EMPT: 🗸 1.0 Show As Double		017
CPI_GCT_EMPT: 10 ⁶ Show As Science	Double	098
E VI_STALL_CYC		442
🖃 💒 CPI_STALL_LSI 🔰 Show As Integer		872 (
E KILAN		015
CPI_ST CPI_ST		0.0
CPI_ST Change CPI Brea	kdown Model	015
CPI_STALL Export as HTML	· 12	0.0
CPI_STALL Select All		858 (
🖃 💒 CPI_STALL_FXI	Select MIT	483
💒 CPI_STALL 🗮 Copy Selection	to Clipboard	0.0
CPI_STALL Find		483 🗸
		>
Details Metrics CPI Breakdown		

The CPI breakdown model to be selected can be either external files or built-in files. The Change CPI breakdown model dialog is as follows:

🚳 Change CPI	Breakdown Todel	×
Select Another Select another C	CPI Breakdown Model PI breakdown model file.	4
○ File : • <u>Built-in :</u> ]	[Plugin]/metadata/cpibreakdown.xml	
	OK Cancel	

#### Export as HTML

Choose Export as HTML on the context menu, you can export CPI breakdown data into an HTML file.
😂 hpmc_cpibreakdown_)	. pmf 🗙 🌘 x1	.pmf					
CPI Breakdown Model	All Processors		Average	Pro	cessor O		~
🖃 📡 TOTAL_CPI	5.332 (100.0%)	5.332	(100.0%)	5.766	(100.0%)	5.	
CPI_CMPL_CY	0.775 (14.54%)	0.775	(14.54%)	1.071	(18.57%)	0.	
🖃 🔽 CPI GCT EMP	0.115 (2.16%)	0.115	(2.16%)	0.141	(2.44%)	0.	
✓ 10 Show	As Double		%)	0.001	(0.01%)		
			%)	0.029	(0.5%)	0.	
10 Show	As Science Double	e	%)	0.112	(1.94%)	0.	
🗆 🔛 C 🛛 🕇 Show	As Integer		%)	4.555	(78.99%)	4.	
- I				2.905	(50.37%)	2.	
Filt	er		%)	0.008	(0.13%)	0.	
Chan	ge CPI Breakdown M	Model .	%)	0.0	(0.0%)		
Expo	rt as HTML	N	%)	0.008	(0.13%)	0.	
			×)	0.0	(0.01%)		
Sele	et All		%)	2.896	(50.23%)	2.	
🖃 🗯 🖺 Сору	Selection to Clip	pboard	%)	0.188	(3.26%)	0.	
Find			%)	0.0	(0.0%)		
	. U. 400 (0.000)	0.400	(8.00%)	0.188	(3.26%)	0.	v
<						>	
Details Metrics CPI H	reakdown						

The CPI breakdown HTML file is as follows:

CPI Breakdown Model				C PX	<b>W FX</b>	
TOTAL_CPI h minipion: Total spiles fom the: p_ctr	CT2_C FL_CTC K ztäpfädit Kupäetään sysäns foru dät y cita_C Fk					11, 544
	CPX_SCT_E PTT k zxipios: copietam Takia opty (sct opty)	CFE_SCT_CFTT_EC_ESS N EXP[20]: TC:sct. dor Founday fou dat v _cct_restory_z_E1			•.•	
	The last r justic ritigen	CPZ_6CT_E FIT_BA_ FARD N 233/1501 Facult Argundentian Faculty Fau dat :			17	0.327
		CPZ_6CT_E FIT_0TBEL R maipfion: ethna CCT ptails			1. 175	1.07
	CPZ_STALL_CTC R ZIJ/201: Cupletion Itali 17:802	CPI_STAIL_LSU k minipion: 04-11 kp L00 intertmeting for da: p _C plugatell_log	CPX_STALL_SSM_ARTCCT R TRIPICS: Stall by USP Reset Founda: F _C FLE_TRLL_Righter	CPT_STALL_ISU_FMAT_ ISS Kurzipidos: 14aii by L10 Teneriotics Reject for da: P_c Playatellitat_ Inc		
				CPT_STALL_LSU_REFECT_OTHER R maipline: could be too other fagend	1. 125	0.207
			CPT_STALL_LSU_BCACHT_ ISS h mighio: Sedity the Franks size for dat e_c est_staticect_ me			1.17
			CPT_STALL_ISU_OTHERS R STRIPTOR: Stall by LIN basis interny, LIN Plank presidy		2. 151	53. 514
		CPI_STALL_FID R minister teals by for factoristic	CPT_STALL_FID_BUTH N: IIIPIGN: IIIII to any dama ami BITI TIPIG TIPIG daminantian foi da: F _C FLE_IIII.FII		•.•	1.17
		P _C PLB_STALL_FOR	CPE_STALL_FED_OTHERS R Hilpick: Stall by fit banks latency		1. 413	1. 107
		CFZ_STALL_FF9 CFZ_STALL_FF9_KXY R zzápřádki zekáží k pře k zzápřádki zekáž k prov domi de P277/22077 dostřantáva dostantáva		- ad feitifight dasfestdae	•.•	1.17
		P _C PLU_STALL_PPU	CPT_STALL_FIG_OTHERS N HILIPION: Deall by PPE bases inference		•.•	0.07
		CPT_STALL_OTHERS Is maipline: stadi by other	an lotali by Brailona and , ain	ek provády (oranyi Lil dásek), oto.)	1.013	20.389
cm					5.332	200.07

# 5.4.3.3View Temporal Chart

You can view temporal chart of the counter data in **Graph** view, by selecting **Display Temporal Chart** to enter temporal mode which is set as the default chart mode.



**Aggregation Scale** is used to scale the aggregation rate of samples. You may feel overwhelmed in front of too many samples in **Graph** View in temporal mode sometimes. And with the aggregation scale, it is easy for you to choose the number of samples to display. For example, you may choose to display 20 samples in **Graph** View:



And you may also choose to display 10 samples here:



Filter is used to filter processors.

S Filters	
Select What You Want to See	
Please select the processors that will be displayed.	
Processors	
All Processors Average Processor 0 Processor 1 Processor 2 Processor 3	Select All Deselect All
	K Cancel

In temporal mode, when you select one event in Tab **Details**, one metric in Tab **Metrics**, or one CPI component in Tab **CPI Breakdown**, the temporal information of htis item is displayed in **Graph** view. The chart type can be switched between **Multiple Bar Chart** and **Line Chart**.

• Line chart of event:







• Line chart of one metric:



• Line chart of one CPI breakdown component:



Note: This plot mode is supported by details data, metrics data, and CPI breakdown data.

# 5.4.3.4View CPI Breakdown Chart

In CPI breakdown mode, when you select one CPI component in Tab **CPI Breakdown**, CPI breakdown data can be displayed in **Graph** View. You can choose to display **CPI Children Breakdown Chart** or **CPI Leaves Breakdown Chart**. The former merely displays the sons of the selected node, while the latter displays all the leaf nodes with the selected node as the root. In this mode, chart type can be switched between **Multiple Bar Chart** and **Stacked Bar Chart**. Filter is used to filter processors.

• Display CPI leaves breakdown stacked-bar chart:



Display CPI children breakdown multiple-bar chart:



Display CPI children breakdown multiple-bar chart:



Note: This plot mode is only supported by CPI breakdown data.

# 5.4.3.5Compare Counter Data

You can click  $\stackrel{e}{=}$  on the tool bar to select several files to compare. In the pop-up dialog, you must select 2~4 target files to compare. You can either select files from the explorer, or select files from repositories (you have to refresh repositories in **Counter Repository** View in davance). Select one file, and click the **Add** Button to add it. The first file you add is regarded as the base file.

You can further specify the metrics and CPI breakdown Model to apply to the counter data, or do it later (Please refer to Change Metrics and Change CPI Breakdown Model).

The following screen snapshot displays the Select To Compare Dialog:

Select To Compare	
Select Targets to Compare Please select the targets to compare from file or repository.	
Select Target to Compare File : C Repository :	
	Add Remove Move Up Move Down Clear All
Select Metrics and CPI Breakdown Model Metrics : CPI :	Select Select
OK	Cancel

#### 5.4.3.5.1View Comparison Data

Comparison Editor is opened to display comparison result side by side. The following items can be compared:

- The total counts of one event are compared.
- The derived metrics values are compared.
- The CPI breakdown values are compared.

The following screen snapshot displays comparison data of each event from different data sources. refers to difference between the target file and the base file, while % refers to the target files' proportion of the base file.

😂 Comparison Editor	×			- 0
Events	Average (A)	Average (B)	∆ Average (B)	% Ave 🔨
PM_INST_DISP	1,039,69	1,249,861	210, 165, 486	
PM_LSU_LDF		1		
PM_FPU_FIN		1		
PM_GCT_NOSLOT_SRQ_FU	0	10 Show As	Double	
PM_FPU_FULL_CYC	0			
PM_CMPLU_STALL_REJEC	2, 788, 342	10 Show As	Science Double	-
PM_FXU_FIN		✓ 1 Show As	Integer	
PM_GRP_IC_MISS_BR_RE	5, 381, 671	- 1		
PM_LD_REF_L1		Filter		
PM_CYC	3,072,99			
PM_CMPLU_STALL_FDIV	364			
PM_GRP_MRK	8, 889, 633			
PM_DTLB_MISS		53		
PM_ITLB_MISS		90		
PM_FXUO_FIN		272, 583, 045		
PM_ST_MISS_L1		337		
PM_FPU_STF		2		
PM_MRK_FPU_FIN		0		V
	1 202 64	0 071 000	1 047 200 411	>
Details Metrics CPI	Breakdown			

The comparison is based on the sum of all processors, the average of all processors, or both. You can specify it in **Filter**.



#### 5.4.3.5.2View Temporal Comparison Chart

When Comparison Editor is opened, comparison data are displayed in **Graph** view. You can view temporal chart of the comparison data by selecting **Display Temporal Chart** to enter temporal mode which is set as the default chart mode.

The following screen snapshot displays the temporal comparison line chart of one event. The temporal comparison chart of one metric and one CPI breakdown component is much the same with that of one event. The comparison is based on the sum of all processors, the average of all processors, or both. You can specify it in **Filter**.



Note: This plot mode is supported by details data, metrics data, and CPI breakdown data.

#### 5.4.3.5.3View CPI Breakdown Comparison Chart

When Comparison Editor is opened, comparison data are displayed in **Graph** view. When you select one CPI component in Tab **CPI Breakdown**, you can choose to display **CPI Children Breakdown Chart** or **CPI Leaves Breakdown Chart**. The former merely displays the sons of the selected node, while the latter displays all the leaf nodes with the selected node as the root. In this mode, chart type can be switched between **Multiple Bar Chart** and **Stacked Bar Chart**. The comparison is based on the sum of all processors, the average of all processors, or both. You can specify it in **Filter**.



Note: This plot mode is only supported by CPI breakdown data.

# 5.4.3.6Edit Metrics/CPI Breakdown Model

#### 5.4.3.6.1 Edit Derived Metrics

- 1. You can select mon the tool bar to open one derived metrics file to edit.
- 2. You can view and edit derived metric formula in Metrics Editing dialog. The supported operation includes:
  - Add new derived metric
  - Edit derived metric
  - Delete derived metric

🚳 Letrics Editor		
<b>Metrics Editing</b> Please open an old metric file or create	a new metric :	file
File Name New Metrics File		
Metrics Name New Metrics	Description	
Processor Undefined Processor 🔻		
OS Undefined Os		
-Detail Information		
	Name	
	Description	
	Formula	
	16-14	
	Unit	· ·
New Remove Clear		
		Open Save As., Save Close

3. After modification, you can save the file or save it as another derived metrics file.

#### 5.4.3.6.2Edit CPI Breakdown Model

- 1. You can select ^[P]on the tool bar to open one CPI breakdown model definition file.
- 2. You can view and edit CPI breakdown model in **Edit CPI Breakdown Model** dialog. The supported operation includes:
  - o Add new component
  - Edit component
  - Delete component

🚭 Edit CPI Breakdown Model		×
<b>Edit CPI Breakdown Model</b> Create or edit CPI breakdown model		6
File Name: * New CPI Breakdown Model	File	
Name: CPI name	Description:	
Processor: Unspecified 💌		~
OS Family: Unspecified 💽		
Instruction Event: PM_INST_CMPL		~
Detail Info		
····· Matter Total_CPI	Name :	
	Description:	
	Formula:	
		<u> </u>
Open	Save Save As	<u>C</u> lose

3. After modification and validation, you can save the file or save it as another CPI breakdown model definition file.

# 5.4.3.7Import Counter Data File into Repository

To create Counter Analyzer Support, you should create a database connection first. Please refer to section <u>5.1.7</u> for details on how to create and manage database connections.

Then, right-click on the connection to create Counter Analyzer Support,

If you are using hsqldb connection, all existing database under this path will be listed, you can create a new one, or attach an existing one.

If you are using db2 connection, all existing schema under this database will be listed, you can create a new one, or attach an existing one.

🖨 Add Counter	Analyzer	X					
Add Counter Analyzer Add Counter Analyzer support to this connection.							
Input the new repository name, or select an existing one.							
💿 New :		]					
C Existing:	2counter counter1 counter count1 1107 cc1						
0	OK Cancel						

**Create Counter Analyzer Support** 

After Counter Analyzer Support has been created, then, you can import file into this support, open it in editor, or delete it.

Size	Date	Size Date
🕀 🗗 Local Connection (Hsqld		🖅 🖵 Local Connection (Hsqld
🖶 📇 9. 186. 10. 107		🖻 🚰 9. 186. 10. 107
🖮 🗗 My Hsqldb		🖨 🖵 My Hsqldb
🖻 🖸 Counter Analyzer 1.16	_	🚍 <u>C</u> Counter Analyzer 1.16
🚽 🛺 hpm d 🔐 Refresh	007-04-24 17:57	
a.pm 🔁 a.pm ran Import File	007-04-24 18:04	
🛄 d. pn	- 007-04-24 18:04	
🔄 🔟 c.pn 🗙 Delete	007-04-24 18:04	c.pm 🖌 Delete Terested Wile
	2007-04-24 18:04	b.pm
🖻 📊 Profile Analyzer N/A		🗄 📊 Profile Analyzer N/A

You can delete files manually to release disk space. Multi-selecting is allowed. See the picture below.

🖮 🖵 My Hsqldb		
🚍 ⊆ Counter Analyzer 1.16		
- 🔊 hpmccompare_1_2. I	2007-04-24	17:57
🔟 a. pmf	2007-04-24	18:04
🚽 🗾 d. pmf	2007-04-24	18:04
🔟 c. pn 🖕		18:04
🔟 b. pa 🚺 Open in Editor	1-24	18:04
🗄 📊 Profile 🗙 Bolote Imported Wil		
➤ Derece Imported FII	e	

This view support sorting operation. You can sort the files by name, size or date.

e: 1	<b>D</b> .
51Ze	Date

To sort files, select the sort mode on the action bar, or click the title directly.



# 5.5Trace Analyzer

Trace Analyzer visualizes Cell/B.E. traces containing information such as DMA communication, locking/unlocking activities, mailbox messages, etc.

Trace Analyzer provides several views that help the user make sense of the trace data. The trace can be plotted in a graphical view, organized by core, along a common timeline. Alternatively, the user can traverse the trace records in a textual table, Another view provides the detailed data for each kind of records, for example, lock identifier for lock operations, accessed address for DMA transfers, etc.

You can also find the Trace Analyzer User Guides from the VPA. Select **Help - Help Contents** within VPA. To get context sensitive help, press **F1** for Windows and AIX or press **Ctrl+F1** for Linux.

# 5.5.1Basic concepts

#### Events

Events are records that have no duration, for example, records describing non-stalling operations, such as releasing a lock. Events' input on performance is normally insignificant, but they may be important for understanding the application and tracking down sources of performance problems.

#### Intervals

Intervals are records that may have non-zero duration. They normally come from stalling operations, such as acquiring a lock. Intervals are often a very significant performance factor, and identifying long stalls and their

sources is an important task in performance debugging. A special case of an interval is *live interval*, that starts when an SPE thread begins to execute and ends when the thread exits.

#### 5.5.2Load an existing trace file

#### 5.5.2.1 Open Trace Analyzer Perspective

When you first start Visual Performance Analyzer after installation, the default perspective is Profile Analyzer. To start Trace Analyzer, choose **Window -> Open Perspective -> Other.** In the Select Perspective dialog choose **Trace Analyzer** and click **OK**.

The following screen capture shows the initial layout of Trace Analyzer.



#### 5.5.2.2Load in trace file

Choose File -> Open File_, and in Open File dialog select one trace file with suffix ".pe"..

#### 5.5.2.3Brief introduction to Trace Analyzer Perspective

After loading in the trace data, the Trace Analyzer Perspective displays the data in its views and editors. Going from the top left clockwise, we see:

- Navigator View
- Trace Editor showing the trace visualization by core
- **Details View** showing the details of the selected record (if any)
- Color Map View, allowing the user to view and modify color mapping for different kinds of events
- Trace Table View, which shows all the events on the trace in the order of their occurrence

### 5.5.3Navigate the Trace Analyzer Perspective

The following are tasks that you can perform to navigate around trace data within Trace Analyzer.

#### 5.5.3.1 View Trace Data Graph

A graphical presentation of the trace is shown in the editor window.



Data from each processor is displayed in a separate row, and each trace record is represented by a rectangle. Time is represented on the horizontal axis, so that the location and size of a rectangle on the horizontal axis represent the corresponding event's time and duration. The color of the rectangle represents the type of event, as defined by the Color Map View. In the rows corresponding to the SPEs, note the full-height green rectangles. They show the live intervals, i.e., the life time of SPE threads. Records represented by the shorter reddish rectangles represent events that occurred during the thread execution.

#### **Trace Editor Components**

The following figure shows the different components of the Graphical View.



Going from top to bottom, we have:

- The *marker ruler* shows where the selected record (if any) is located on the trace's timeline (look for the orange-and-white selection marker). Clicking on the marker ruler scrolls the view to make the selected event visible. Note that there are also vertical marker rulers, located in each row between the core id and the graph. These rulers show on which core the selected event occured.
- The scrollbar can be used to scroll back and forth in time.
- The *ruler bar* shows the time values, in the same units as those used in the trace.

#### **Trace Editor Tools**

When a trace is open in the Graphical View, the following toolbar is added to the standard Eclipse toolbars:



This toolbar is only active when the focus is on the Trace Editor. The following tools are available:

• Selection tool. Pick this tool and click with it a record on the Graphical View to select the record. This will scroll the Record List View to the selected record and display its details in the Record Details View.

- Zoom-in point tool. Pick this tool and click one of the graphs to zoom in while keeping the time value at the click point at the same location.
- Some out point tool. Pick this tool and click one of the graphs to zoom out while keeping the time value at the click point at the same location.
- Soom-all tool. Pick this tool and click anywhere in the graph to fit all the trace into the view.
- Section 2012 Sec

2007012	511	5703.pe 🗙				
		400000	50000	500000	70000	800000
PPE		100000			100000	
SPE 0				]		
SPE 1						
SPE 2						
SPE 3						
SPE 4						

• Trag tool. To scroll the view back and forth along the time axis, pick this tool, and hold the right mouse button pressed while dragging the graph

#### Trace Editor Coloring Conventions

The default coloring used by the Trace Editor assigns hues of blue to events and hues of red to intervals. One of the few exceptions are the live intervals, colored green. Please refer to the Color Map View for the exact color mapping of any particular editor instance.

When analyzing a trace, it is often important to distinguish between a large number of short intervals and a single long interval, which may be a good target for optimization. In order to aid in this analysis, Trace Analyzer emphasizes intervals with a border whose color is a darker hue of the event's color:



# 5.5.3.2View List of Trace Records

You can view the list of the records in the trace in the **Trace Table View**. Click on a row to select a record and see its details in the **Record Details View**. **Trace Table View** selection is also synchronized with the selection in the **Trace Editor**, so that each scrolls to and highlights the selection done in the other.

Trace Ta	able View 🛛		
Trace file:	0 in C:\Documents and Sett	ings\ps-user\	runtime-v 💌
Index	Record type	Trace file	Record N
23	SPE READ IN MBOX	0	3
24	SPE_READ_IN_MBOX	0	5
25	PPE_SPE_WRITE_IN	0	3
26	SPE_READ_IN_MBOX	0	4
27	SPE_READ_IN_MBOX	0	4
28	SPE_READ_IN_MBOX	0	6
29	PPE_SPE_WRITE_IN	0	4
30	SPE_READ_IN_MBOX	0	5
31	SPE_READ_IN_MBOX	0	5
32	SPE_MFC_GET	0	0
33	SPE_MFC_GET	0	1
34	SPE_MFC_GET	0	2
35	SPE_MFC_GET	0	3
36	SPE_MFC_READ_TAG	0	0
37	SPE_MFC_READ_TAG	0	0
38	SPE_MFC_READ_TAG	0	0
39	SPE MFC READ TAG	0	1
•			• •

#### 5.5.3.3View Record Details

The Record Details View shows the names and values for all the fields defined for the selected record.

Record View	xx		
SPE_MFC_READ	TAG_STATUS #0		
Name	Value	Туре	
Event Name	SPE_MFC_READ_TAG_STATUS	string	
Duration	5	long	
Mask	1	integer	
Physical SPE Id	0	integer	
Update Mode	2	integer	
Tag status	1	integer	
Event Count	12	integer	
EndTime	1279237	long	
StartTime	1279232	long	
EventType	322	integer	

# 5.5.3.4Change Colors used in Trace Editor

The **Color Map View** holds a list of record types and their color mapping. To change the color mapping for a particular record type, double-click the corresponding row. A color chooser dialog will open, allowing to select the new color for the record type.

📩 Color Map View 🛛	- E
Color Map: CELL	•
Name	Color 🔺
PPE_SPE_DESTROY_GROUP	
PPE_SPE_CREATE_THREAD	
PPE_SPE_KILL	
PPE_SPE_READ_OUT_MBOX	
PPE_SPE_READ_OUT_INTR_MBOX	
PPE_SPE_WRITE_IN_MBOX	
PPE_SPE_WRITE_SIGNAL	
PPE_SPE_MFC_PUT	
PPE_SPE_MFC_PUTB	
PPE_SPE_MFC_PUTF	
PPE_SPE_MFC_GET	
PPE_SPE_MFC_GETB	
PPE_SPE_MFC_GETF	
PPE SPE MEC READ TAG STATUS IMM	

Note that the interval types, which are emphasized with a darker border in the Trace Editor, are also shown with a border in the Color Map View:

PPE_SPE_WAIT

# 6.Appendix A - sample profiling session

This process will walk through the typical usage of VPA to analyze problems, using the **Profile Analyzer plug-in** and some existing data. Where can you download the existing data? Look for the **Sample ETM File** at the bottom of the web page: <u>http://www.alphaworks.ibm.com/tech/vpa/download</u>.

The picture below shows the file being saved to a workspace directory.



If you don't have the Profile Analyzer directory, try this:

Profile Analyzer - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
] 📬 🕶 🔚 🗁   🗛 🕶   🛷   🦇 🗇 🖛 🔿 👻
№ Navigator X         Profiling Resources         □
Profile Analyzer
Click on the Profiling Resources tab, somehow this causes it to create the missing directory.
Then click on the Navigator tab to return to the view we want.
Profi e Analyzer - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
№- Navigator X         Profiling Resources         □
Profile Analyzer

Once we have the Profile Analyzer folder in view, it is sometimes necessary to REFRESH it. Just Right click and select REFRESH.





If you don't have a Navigator tab, adding a new view is easy with VPA! Just select Window => Show View => Other...



Select Navigator. It will open it as a tab in the bottom right window. Just drag it to the top left where we want it.

Show View	
Ant     Ant     Basic     Bookmarks     Classic Search     Console     Co	

You will then need to navigate to D:\eclipse\workspace\Profile Analyzer to see your data.

Open the SAMPLE ETM by double clicking on it in the Navigator view.

Profile Analyzer - Eclips	se SDK
File Edit Refactor Navigate Searc	h Project Run Window Help
] 📬 🕶 📄 🕒 ] 💁 🖌 🖋 ] 🍫 🗇	$\bullet \bullet \Rightarrow \bullet$
🔁 Navigator 🛛 Profiling Resources	,
	(> @   □ 🔄 🔽
Profile Analyzer .project pi_config.cfg .t ^µ sample.etm	

The initial view will look something like this:

Profile Analyzer - sample.etm - Eclipse SDF	(	
File Edit Refactor Navigate Search Project Run Window	Help	
] 📸 🕶 🔄 💁 🚽 🖉 🖉 😓 🗢 🛪 🔿 🔻		🗈 📊 Profile Anal ×
😪 Navigator 🛛 🖓 Profiling Resources 🛛 🖓 🗖	💾 sample.etm 🛛	- 8)
(→ → @   = 😓 🗸	sample.etm	
🖃 🗁 Profile Analyzer	🖃 🔁 Profile sample.etm (11028 ticks)	. % Process Thread 🔨
.project	Process > Thread > Module 220	08 20.02 wait_a014 wait TID 45079
pi_config.cfg	Process > Module 218	34 19.80 wait_b016 wait TID 49177
sample.etm		70 19.68 wait_c018 wait TID 57373
	128	33 11.63 java_410f4 java TID 577623
	64	12 5.82 java_410f4 java TID 577623
	53	32 4.82 wait_b016 wait TID 49177
	52	23 4.74 Wait_a014 Wait TID 45079
	40	00 4.17 Walt_C018 Walt 11D 57373
	22	20 2.90 Java_41014 Java 110 577023
	15	58 1 43 java 410f4 java TID 577623
	<	
	Proc	ess > Thread > Module
Complex Distribut M Deschard Calls Counterry No. 💷	🛗 Disassembly/Offsets 🛛 🛛 Source Code Compiler Lis	sting Java/Classes Hier ²⁷ 4
Samples Distribut 23 Resolved Calls Counters 2	No symbol is selected.	🛢 🎭 🗁 🖌 🎽 🗢
<u> </u>		
wait a014wait b016		
wait_col8 java_410f4		
wait_2004		
Other		
25.0% 24.90% 24.19% 23.24% 2.01% 0.05%		
Process > I nread > Module - I otal ticks: 11028, time share: 100.00		
Process > I nread > Module: All Immediate children		

Expand Process > Module on the top right window by clicking the + sign :

🔡 sample.etm 🛛			
sample.etm			
🖃 💦 Profile sample.etm (11028 ticks)	Ti	%	Process
Process > Thread > Module	2208	20.02	wait_a014
	2184	19.80	wait_b016
	2170	19.68	wait_c018
	1283	11.63	java_410f
	642	5.82	java_410f
	<b>E</b> 22	4.00	

Once expanded, we can see that on the 4-way test system, our single threaded Java test case left 3 of the processors idling, leaving the ticks nicely divided amongst the processors. The Java test case (Process ID 410f4) took 23.24% of the total ticks. A *tick* is a sampled address recording where the system was executing code. JITCODE is where the work for this Java test case is being done, and we expect this due to Jitting of the Java methods.

🔡 sample.etm 🛛				
sample.etm				
Profile sample.etm (11028 ticks)	Ti	%	Module	
Process > I nread > Module	1283	50.06	JITCODE	
• wait_a014(2757 ticks/25.00%)	320	12.49	JITCODE	
wait_b016(2746 ticks/24.90%)	158	6.16	JITCODE	
wait_c018(2668 ticks/24.19%)     iava_410f4(2563 ticks/23.24%)	80	3.12	JITCODE	
wait_2004(288 ticks/2.61%)	6	0.43	/usr/java14/jre/bin/classic/libjvr /usr/lib/libc.a[shr.o]	
gil_e01c(2 ticks/0.02%)	3	0.12	/usr/lib/libpthreads.a[shr_xpg5.	
trcstop_410f6(2 ticks/0.02%)     netm_d01a(1 ticks/0.01%)	3	0.12	/usr/lib/libpthreads.a[shr_xpg5.	
	2	0.12	/usr/lib/libpthreads.a[shr_xpg5.	~
	<		>	
	Total:	ticks (25	563), time share (23.24%)	

The bar chart on the bottom left of VPA shows this well. Within the Java Process, we took a full 96.87% of the ticks in Jitted code. (If not shown, just click the bar chart icon on the Samples Distribution tab) :

Samples Distribut	Resolved Calls	Counters	»2		
				6	$\overline{}$
JIT CODE	Other				
96.87%	3.12%				
Process: java_410f4 - Process: All Process ur	Total ticks: 2563, time nder java_410f4	e share: 23	3.24%		

Almost everything with VPA is a mouse click away!!!

Double clicking any tab will change to a maximized view of that tab. Double clicking the same tab will change it back.

Try it by double clicking the tab for the window on the top right:

🔡 sample.etm 🛛	_			
sample.etm Double click on tab				
🖃 🖓 Profile sample.etm 🕞 👘		%	Module	^
🕀 👯 Process > Thread > Module	1283	50.06	JITCODE	
Process > Module	642	25.05	JITCODE	
🕀 💽 wait_a014(2757 ticks/25.00%)	320	12.49	JITCODE	
wait_b016(2746 ticks/24.90%)	158	6.16	JITCODE	
mait_c018(2668 ticks/24.19%)	80	3.12	JITCODE	
java_410f4(2563 ticks/23.24%)	11	0.43	/usr/java14/jre/bin/classic/libjvr	
🕀 🙋 wait_2004(288 ticks/2.61%)	6	0.23	/usr/lib/libc.a[shr.o]	
	3	0.12	/usr/lib/libpthreads.a[shr_xpg5.	
Itrcstop_410f6(2 ticks/0.02%)	3	0.12	/usr/lib/libpthreads.a[shr_xpg5.	
	3	0.12	/usr/lib/libpthreads.a[shr_xpg5.	
	2	0.08	/unix	*
⊡ <mark>.</mark>	<	1111	>	
	Total:	ticks (25	63), time share (23.24%)	

This makes it easier to see everything.

e Edit Refactor Navigate Search Project Run Window Help				
i 🕶 🔜 🕒 🛛 🗛 🕶 🖉 🦑 🖢 🌾 🔶 👻				🗈 📊 Profile Anal
sample.etm 🕱				-
nple.etm				
- Profile sample.etm (11028 ticks)	Tim	%	Module	Symbol
ET Process > Thread > Module	1283	50.06	ITCODE	thirthdwo()I
Process > Module	642	25.05	ITCODE	sixteen()I
wait_a014(2757 ticks/25.00%)	320	12.49	ITCODE	eight()I
wait_b016(2746 ticks/24.90%)	158	6.16	ITTCODE	four()I
wait_c018(2668 ticks/24.19%)	80	3.12	ITCODE	twoOI
🕂 💽 java_410f4(2563 ticks/23.24%)	11	0.43	/usr/iava14/ire/bin/classic/libivm.a	.mmipExecuteJava
🗉 🥘 wait_2004(288 ticks/2.61%)	6	0.23	/usr/lib/libc.a[shr.o]	noSymbol
gil_e01c(2 ticks/0.02%)	3	0.12	/usr/lib/libpthreads.a[shr xpq5.o]	.pthread getspecific 1
trcstop_410f6(2 ticks/0.02%)	3	0.12	/usr/lib/libpthreads.a[shr_xpq5.o]	.pthread mutex unloc
inetm_d01a(1 ticks/0.01%)	3	0.12	/usr/lib/libpthreads.a[shr_xpq5.o]	.global_unlock_ppc_m
🗄 💽 sh_410f4(1 ticks/0.01%)	2	0.08	/unix	.vm_handle
ie-™ Modules	2	0.08	/unix	.trchook64
	2	0.08	/unix	sc_msr_2_point
	1	0.04	/usr/lib/libpthreads.a[shr_xpg5.o]	.pthread_mutex_tryloc
	1	0.04	/usr/lib/libc.a[shr.o]	.fwrite
	1	0.04	/usr/lib/libc.a[shr.o]	.fflush_unlocked
	1	0.04	/usr/lib/drivers/pci/phxentdd[phxentdd64]	.rx_handler
	1	0.04	/usr/lib/drivers/netinet[netinet64]	.in_cksum
	1	0.04	/usr/java14/jre/bin/libzip.a	.inflate_fast
	1	0.04	/usr/java14/jre/bin/libzip.a	moveeq
	1	0.04	/usr/java14/jre/bin/libzip.a	.readCEN
	1	0.04	/usr/java14/jre/bin/libzip.a	.huft_build
	1	0.04	/usr/java14/jre/bin/libjitc.a	.transform_codeattr
	1	0.04	/usr/java14/jre/bin/libjitc.a	.SAME_VARREF
	1	0.04	/usr/java14/jre/bin/libjitc.a	.Initialize_const_table
	1	0.04	/usr/java14/jre/bin/libjitc.a	.dopt_require_term_st
	1	0.04	/usr/java14/jre/bin/libjitc.a	.dopt_regenerate_dag
	1	0.04	/usr/java14/jre/bin/libjitc.a	.dopt_intern_operation
	1	0.04	/usr/java14/jre/bin/libjitc.a	.Deadstore_Init_Datafle
	1	0.04	/usr/Java14/jre/bin/libjitc.a	.CostBenefitAnalysis
	<	_	1111	>

Double click the same tab to go back to the 4 pane view:



Select PID 410f4 and click the + to expand and see the modules within this process:



As we noted earlier, the JITCODE module is showing the most ticks as expected for the Java process. On the right hand side we see a breakdown of ticks for the JITCODE module.



Clicking on the JITCODE Module expands the window on the right, showing a breakdown of ticks in the JITCODE. The hottest symbols are shown in descending order. Notice that thirtytwo() is the biggest contributor at almost 52%:



If you look at the Samples Distribution view in the lower left corner, you will see that the information now corresponds to the JITCODE symbols. You can also switch between pie and bar charts.



Double click on thirtytwo() and notice that the disassembly/Offset information in the bottom right window is now filled in for this method. Clicking on any other method on the top right will change everything we are seeing in the graph and in this disassembly to correspond to the method selected:

Offeet	Butes	Disassembly	Ticks	Demarks	
011500	01-1555-		 IICAS	Remarks	 
0x00000000	91elffic	stw r15,-4(r1)			-
0x0000004	39e10000	addi r15,r1,+0			_
0x00000008	3821fff0	addi r1,r1,-16			
0x000000c	7c2e0808	tw TO_LGT,r14,r1			=
0x0000010	39600000	li r11,+0			_
0x00000014	3d403571	lis r10,0x3571			_
0x0000018	814aaf88	lwz r10,-20600(r10)			
0x000001c	3d205f5e	lis r9,0x5f5e			
0x0000020	61291000	ori r9,r9,0x1000			
0x0000024	7d2a4a14	add r9,r10,r9			
👃 0x00000028	4800000c	b 0x34			
0x0000002c	6000000	ori r0,r0,0x0			
0x0000030	6000000	ori r0,r0,0x0			
0x0000034	3d405f5e	lis r10,0x5f5e			
0x0000038	614a1000	ori r10,r10,0x1000			
0x000003c	7f8b5000	cmp cr7,0,r11,r10			
0-00000040	409c000c	be BO IF NOT CR7 LT 0x4c			

We need to get the Disassembly/Offset Information as a full screen, so double click that tab:



#### We now see a full window view of this:

Profile An	alyzer - san	nple.etm - Eclipse SDK							_	
File Edit Refact	tor Navigate S	earch Project Run Window Help								
📬 🕶 🖪 📥 🗍	9a • ] 🛷 ] 🥲	$\phi \star \phi \star$							🗈 <u> ii</u> Profile An	ial
Disassembly/C	ffsets 🛞 Sou	rce Code Compiler Listing Java/Classes Hi	ierarchy Call-Gr	aph Calle	ers/Descendants	Console	Profile Comparison	Disassembly Comparison		
Offerste ferri thirthd		= 410f4)			-,			,,		1.1.6
Onsets for: thirtyt	wo()i(Process Jav	a 41014)							🖷 🐿   🍝 🗲	
Offset	Bytes	Disassembly		Ticks	Remarks					^
0x00000000	91e1fffc	stw r15,-4(r1)								
0x0000004	39e10000	addi r15,r1,+0								
0x0000008	3821fff0	addi r1,r1,-16								
0x000000c	7c2e0808	tw TO_LGT, r14, r1								
0x00000010	39600000	li r11,+0								
0x00000014	3d403571	lis r10,0x3571								
0x00000018	814aaf88	lwz r10,-20600(r10)								
0x0000001c	3d205f5e	lis r9,0x5f5e								
0x00000020	61291000	ori r9,r9,0x1000								
0x00000024	7d2a4a14	add r9,r10,r9								
🚴 0x0000028	4800000c	b 0x34								
0x000002c	6000000	ori r0,r0,0x0								
0x0000030	6000000	ori r0,r0,0x0								
0x0000034	3d405f5e	lis r10,0x5f5e								=
0x0000038	614a1000	ori r10,r10,0x1000								
0x000003c	7f8b5000	cmp cr7,0,r11,r10								
🚴 0x0000040	409c000c	bc BO_IF_NOT,CR7_LT,0x4c								
0x0000044	396b0001	addi r11,r11,+1								
雀 0x0000048	4bffffec	b 0x34								
0x000004c	38690000	addi r3,r9,+0								
0x0000050	3d603571	lis r11,0x3571								
0x0000054	906baf88	stw r3,-20600(r11)								
0x0000058	382£0000	addi r1,r15,+0								
0x000005c	81effffc	lwz r15,-4(r15)								
0x0000060	4e800020	blr								
0x0000064	38000001	li r0,+1								
0x0000068	90100170	stw r0,+368(r16)								
0x000006c	382ffff0	addi r1,r15,-16								
0x0000070	7c2e0808	tw TO_LGT,r14,r1								
0x0000074	81610000	lwz r11,+0(r1)								
0x0000078	3c603571	lis r3,0x3571								
0x000007c	8063af88	lwz r3,-20600(r3)								
0x0000080	3d405f5e	lis r10,0x5f5e	32.11	412						
0x0000084	614a1000	ori r10,r10,0x1000							1	× 1

Notice the colored bar on the right. This is the Hotness Bar.

Red is used for any symbol that represents at least 20% of the total tick count.

Magenta is used for any symbol that represents between 5% and 20% of the total tick count.

Blue is used for any symbol that uses less than 5% of the total tick count.

Darker shades show higher activity than lighter shades.

Offset	Bytes	Disassembly	-	Ticks	Remarks	
0x0000060	4e800020	blr				
0x0000064	38000001	li r0,+1				Hotness bar for
0x0000068	90100170	stw r0,+368(r16)				symbol thirtytwo()
0x000006c	382ffff0	addi r1,r15,-16				Symbol timitytwo()
0x00000070	7c2e0808	tw TO_LGT, r14, r1				7
0x0000074	81610000	lwz r11,+0(r1)				
0x00000078	3c603571	lis r3,0x3571				
0x0000007c	8063af88	lwz r3,-20600(r3)				
0x0000080	3d405f5e	lis r10,0x5f5e	32.11	412		
0x0000084	614a1000	ori r10,r10,0x1000				
0x0000088	7f8b5000	cmp cr7,0,r11,r10				
C 0x000008c	409cffc4	be BO_IF_NOT, CR7_LT, 0x50	35.62	457		
0x00000090	38630001	addi r3,r3,+1	32.27	414		=
0x0000094	396b0001	addi r11,r11,+1				
C 0x00000098	4bffffe8	b 0x80				
0x000009c	ccccccc3570	JITCODE MB 0x3570ad40				
0x000000a4	35238f3c	addic. r9,r3,-0x70c4				

We want to see the code with the most activity, so click on the % column to do a quick sort:

Offset	Bytes	Disassembly		Ticks	Remarks		~
	409cffc4	be BO_IF_NOT,CR7_LT,0x50	35.62	457			
0x00000090	38630001	addi r3,r3,+1	32.27	414			
0x0000080	3d405f5e	lis r10,0x5f5e	32.11	412			
0x00000a4	35238f3c	addic. r9,r3,-0x70c4					=
0x000009c	ccccccc3570	JITCODE MB 0x3570ad40					
🔁 0х0000098	4bffffe8	b 0x80					
0x0000094	396b0001	addi r11, r1 Click on % columi	n				
0x0000088	7f8b5000	cmp cr7,0,1 to sort highest					
0x0000084	614a1000	ori r10, r10 percentage of tick	s				
0x000007c	8063af88	1wz r3,-206 towards the top					
0x0000078	3c603571	lis r3,0x35					
0x0000074	81610000	lwz r11,+0(r1)					
0x00000070	7c2e0808	tw TO_LGT,r14,r1					
0x000006c	382ffff0	addi r1,r15,-16					
0x0000068	90100170	stw r0,+368(r16)					
0x0000064	38000001	li r0,+1					
0x0000060	4e800020	blr				[	-

If you would like to use other Profile Analyzer views that is not currently shown. Simply go to Window->Show View -> Other ...

Profile Analyzer - sample.etm - Eclipse SDK								
File Edit Refactor Navigate Search Project Run	Window Help							
	New Window New Editor							
← → Q	Open Perspective							
Profile Analyzer □ .project □ pi_config.cfg	Show View Other /24 Customize Perspective Save Perspective As Reset Perspective Close Perspective Close All Perspectives Navigation Preferences (24 Other /24 O							

Expand Profile Analyzer to see all the available views ...



And as an example .. click on Temporal Profiling to load the view...



And you should now have a screen that looks like this ...


#!/bin/sh #

# #

{

}

{

## 7.Appendix B – run.tprof_xml.sh

The following script file should be copied to your AIX system. It is required when you create an AIX profile configuration in Profile Analyzer.

```
# Remote launch script for AIX tprof with XML generator
Transfer-encoding: chunked
function CheckForHelp
  _RUN_TPROF_HELP=
  case $1 in
    "?")
      ;&
    "-?")
       _RUN_TPROF_HELP=short ;;
    "??")
      ;&;
    "-??")
      ;&
    "/??")
       _RUN_TPROF_HELP=long ;;
  esac
  if [[ $_RUN_TPROF_HELP != "" ]]; then
    ShowHelp
  fi
function ShowHelp
  echo "run.tprof_xml.sh: Execute AIX tprof and generates XML format data file"
  echo
  echo "Syntax:"
  echo " run.tprof_xml.sh <--start | --stop | --full> <-b #MB> <-s ##> <-r ##> <-R> <-l> <-N> {-m time | -m event -e
## -c ##} <--timedata <--buckets ##>>"
  echo
  echo "Where:"
  echo "-----"
  echo " --start
                   Start tracing only."
  echo
  echo " --stop
                   Stop tracing and generate the XML file."
  echo
  echo " --full
                  Run full trace session and generate the XML file"
  echo
  echo " -b ##
                   Set trace buffer size to ## MB."
  echo
  echo " -s ##
                   Automatically start tracing after ## seconds."
  echo
  echo " -r ##
                   Run (trace) for ## seconds."
DLM
                                                     Alphaworks
```

```
echo
                  Enable PURR weighting."
  echo "-R
                 * If you set -R option when using --start, you must also"
  echo "
  echo "
                  set it when using --stop."
  echo
  echo " -l
                 Turn on binary instructions collecting for tprof2xml."
  echo
  echo " -N
                  Turn on source line number info collecting for tprof2xml."
  echo
  echo " -m {time,event} Select profiling type."
  echo "
                 * -m time to use timer based profiling."
                * -m event to use event based profiling."
  echo "
  echo
  echo " -E ##
                   Specify the event to profile."
                 * this option is only valid when -m event is given."
  echo "
  echo
  echo " -c ##
                  Specify the sampling frequency to use."
  echo
  echo " --timedata Enable time data generation in XML."
  echo
  echo " --buckets ## Buckets number for time data. If not specified, default"
  echo "
                number is 1800."
  echo "
                * --timedata and --buckets options are only valid for"
  echo "
                  the --full or --stop option."
  echo
  exit 0
}
function DefaultOption
{
  _SCRIPT_NAME=$1
  _TPROF_XML_BUFSIZE=""
  _TPROF_XML_START_DELAY=""
  _TPROF_XML_RUN_TIME=2147483647
  _TPROF_XML_START=""
  _TPROF_XML_STOP=""
  _TPROF_XML_FULL=""
  TPROF XML PURR=""
  TPROF XML TIMEDATA=""
  _TPROF_XML_BUCKETS=""
  _TPROF_XML_INSTR=""
  _TPROF_XML_SRCLINE=""
  _TPROF_XML_TYPE=""
  TPROF XML EVENT=""
   TPROF XML FREQ=""
}
function ParseCmdLine
{
  case $1 in
    "-b")
      ;&
    "--buffersize")
      HandleBufferSize $1 $2 ;;
    "-s")
      HandleS $1 $2 ;;
```

```
"-r" )
      HandleR $1 $2 ;;
    "--stop")
      _TPROF_XML_STOP=1 ;;
    "--start" )
       _TPROF_XML_START=1 ;;
    "--full" )
       _TPROF_XML_FULL=1 ;;
    "-R")
       _TPROF_XML_PURR=1 ;;
    "--timedata")
      _TPROF_XML_TIMEDATA=1 ;;
    "--buckets")
      HandleBuckets $1 $2 ;;
    "-l" )
       _TPROF_XML_INSTR=1 ;;
    "-N")
       _TPROF_XML_SRCLINE=1 ;;
    "-m")
      HandleType $1 $2 ;;
    "-e")
      HandleEvent $1 $2 ;;
    "-c" )
      HandleFrequency $1 $2;;
    *)
      echo
      echo "ERROR:" "$1 is not a valid argument. Enter \"run.tprof_xml.sh ? \" for help."
      echo
      exit 1 ;;
  esac
}
function HandleBufferSize
{
  if [[ $2 = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Missing buffer size with --buffersize option. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_BUFSIZE=$2
function HandleS
{
  if [[ $2 = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Missing time delay with -s option. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_START_DELAY=$2
function HandleR
  if [[ $2 = "" ]] ; then
```

}

}

{

```
echo
    echo "ERROR:" $ SCRIPT NAME": Missing run time with -r option. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_RUN_TIME=$2
}
function HandleBuckets
{
  if [[ $2 = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Missing buckets number. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_BUCKETS=$2
}
function HandleType
{
  if [[ $2 = "time" || $2 = "event" ]]; then
    _TPROF_XML_TYPE=$2
  fi
  if [[ $_TPROF_XML_TYPE = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Invalid profiling type. Only -m time and -m event are supported. Quitting."
    echo
    exit 1
  fi
}
function HandleEvent
{
  if [[ $2 = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Missing event with -e option. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_EVENT=$2
}
function HandleFrequency
{
  if [[ $2 = "" ]] ; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": Missing event frequency with -c option. Quitting."
    echo
    exit 1
  fi
   _TPROF_XML_FREQ=$2
}
function PostProcessing
{
  if [[ ! -f rse_trace.trc || ! -f rse_trace.syms ]] ; then
```

```
echo
    echo "ERROR:" $ SCRIPT NAME": tprof trace files not found. Quitting"
    echo
    exit 1
  fi
  #
  # call the XML generator
  #
  OPTIONS=""
  if [[ $_TPROF_XML_PURR != "" ]] ; then
    OPTIONS=$OPTIONS" -R "
  fi
  if [[ $_TPROF_XML_TIMEDATA != "" ]]; then
    OPTIONS=$OPTIONS" --timedata
  fi
  if [[ $_TPROF_XML_BUCKETS != "" ]] ; then
    OPTIONS=$OPTIONS" --buckets "$_TPROF_XML_BUCKETS
  fi
  /usr/lib/perf/tprof2xml $OPTIONS -c rse_trace.trc -s rse_trace.syms -o out.etm
  echo '## done get the file##'
}
function Cleanup
{
  rm -f rse_trace.*
  unset _SCRIPT_NAME
  unset _TPROF_XML_BUFSIZE
  unset _TPROF_XML_START_DELAY
  unset _TPROF_XML_RUN_TIME
  unset TPROF XML STOP
  unset _TPROF_XML_START
  unset _TPROF_XML_FULL
 unset TPROF_XML_PURR
  unset _TPROF_XML_TIMEDATA
  unset TPROF XML BUCKETS
  unset TPROF XML INSTR
  unset _TPROF_XML_SRCLINE
  unset _TPROF_XML_TYPE
  unset _TPROF_XML_EVENT
  unset _TPROF_XML_FREQ
}
#
# Script Entry ...
#
# (1) parse for cmd line options
# (2) validate options
# (3) invoke proper sub command: START | STOP | FULL
#
# START & FULL ... kick off Tprof in background
#
# STOP ... waits for background Tprof to exit and then runs tprof2xml ... output is OUT.ETM file
#
```

```
CheckForHelp $1
DefaultOption $0
while [[ $1 != "" ]]
do
  ParseCmdLine $1 $2
  if [[ $1 = "-b" || $1 = "-s" || $1 = "-r" || $1 = "--buckets" || $1 = "-m" || $1 = "-e" || $1 = "-c" ]] ; then
    shift
  fi
  shift
done
###
### Validate Options
###
if [[$ TPROF XML START = "" && $ TPROF XML STOP = "" && $ TPROF XML FULL = "" ]]; then
  ShowHelp
  exit 1
fi
if [[ $_TPROF_XML_TYPE = "" ]] ; then
  echo
  echo "ERROR:" $_SCRIPT_NAME": Profiling type must be specified by -m time or -m event. Quitting."
  echo
  exit 1
fi
if [[ $ TPROF XML TYPE = "time" ]]; then
  if [[ $_TPROF_XML_EVENT != "" || $_TPROF_XML_FREQ != "" ]] ; then
    echo
    echo "ERROR:" $ SCRIPT NAME": -m time can not work with -e or -c options. Quitting."
    echo
    exit 1
  fi
fi
if [[ $ TPROF XML TYPE = "event" ]]; then
  if [[ $_TPROF_XML_EVENT = "" || $_TPROF_XML_FREQ = "" ]] ; then
    echo
    echo "ERROR:" $ SCRIPT NAME": -m event must be given together with -e and -c options. Quitting."
    echo
    exit 1
  fi
fi
if [[ $_TPROF_XML_EVENT != "" || $_TPROF_XML_FREQ != "" ]] ; then
  if [[ $_TPROF_XML_TYPE != "event" ]]; then
    echo
    echo "ERROR:" $_SCRIPT_NAME": -e or -c must be given together with -m event. Quitting."
    echo
    exit 1
  fi
fi
###
### STOP Command
###
```

```
if [[ $_TPROF_XML_STOP != "" ]] ; then
   SLEEP PID=`ps -e | awk '$NF == "sleep" { print $1; exit }'`
  if [[ $_SLEEP_PID = "" ]] ; then
    echo "ERROR:" "PID not found."
    exit 1
  fi
  kill $_SLEEP_PID
  ###
  ### wait for tprof to finish
  ###
  _TPROF_PID=`ps -e | awk '$NF == "tprof" { print $1; exit }'`
  while [[ $_TPROF_PID != "" ]]
  do
    sleep 2
    _TPROF_PID=`ps -e | awk '$NF == "tprof" { print $1; exit }'`
  done
  ###
  ### do post process
  ###
  PostProcessing
  Cleanup
  echo
  echo Done
  echo
  echo '##_done_stop_tools##'
  exit 0
fi
#
# START or FULL command
#
if [[ $_TPROF_XML_START != "" || $_TPROF_XML_FULL != "" ]] ; then
  ###
  ### kill previous not finished tprof process
  ###
  _TPROF_PID=`ps -e | awk '$NF == "tprof" { print $1; exit }'`
  while [[ $_TPROF_PID != "" ]]
  do
    /usr/bin/trcstop
    kill $_TPROF_PID
    sleep 1
    _TPROF_PID=`ps -e | awk '$NF == "tprof" { print $1; exit }'`
  done
  ###
  ### kill the sleep process launched by tprof
  ###
  _SLEEP_PID=`ps -e | awk '$NF == "sleep" { print $1; exit }'`
  while [[ $ SLEEP PID != "" ]]
  do
```

```
kill $_SLEEP_PID
    sleep 1
    _SLEEP_PID=`ps -e | awk '$NF == "sleep" { print $1; exit }'`
  done
fi
###
### start delay
###
if [[ $_TPROF_XML_START_DELAY != "" ]]; then
  sleep $_TPROF_XML_START_DELAY
fi
###
### Launch tprof
###
rm -f rse_trace.*
rm -f out.etm
OPTIONS=""
if [[ $_TPROF_XML_PURR != "" ]]; then
  OPTIONS=$OPTIONS" -R "
fi
if [[ $_TPROF_XML_INSTR != "" ]]; then
  OPTIONS=$OPTIONS" -I "
fi
if [[ $_TPROF_XML_SRCLINE != "" ]]; then
  OPTIONS=$OPTIONS" -N "
fi
if [[ $_TPROF_XML_TYPE = "event" ]]; then
  OPTIONS=$OPTIONS" -E "$_TPROF_XML_EVENT" -f "$_TPROF_XML_FREQ" "
fi
if [[ $_TPROF_XML_START != "" ]]; then
  tprof -eukj -A -F $OPTIONS -r rse_trace -x sleep $_TPROF_XML_RUN_TIME &
  echo '##_done_start_tools##'
  exit 0
else
  echo '## done start tools##'
  tprof -eukj -X -A -F $OPTIONS -r rse_trace -x sleep $_TPROF_XML_RUN_TIME
fi
PostProcessing
Cleanup
echo
echo Done
echo
echo '##_done_start_tools##'
exit 0
```

## 8.Appendix C – run.tprof_e.cmd

The following cmd file should be copied to your Performance Inspector for Windows bin directory. It is required when you create Windows profile configuration in Profile Analyzer.

```
@setlocal
@echo off
  set TPROF POST RC=0
  echo.
  echo ***** run.tprof v2.06 *****
  echo.
  goto CheckForHelp
:Help
  echo run.tprof: Coordinates the TPROF procedure
  echo.
  echo Syntax:
  echo ------
  echo run.tprof ^<-a ^| -t ^| -p^> ^<-b #MB^> ^<-m [time ^| event]^> ^<-s ##^> ^<-r ##^>
               Options valid with '-m time':
  echo
                 ^<-c #ticks_per_second^>
  echo
               Options valid with '-m event':
  echo
  echo
                 ^<-e event name^> ^<-c #events^>
  echo.
  if "%_RUN_TPROF_HELP%" == "long" goto ContinueHelp
  echo Enter "run.tprof ??" for more extensive help.
  echo.
  goto Exit
:ContinueHelp
  echo Where:
  echo -----
  echo -a
               Run entire TPROF procedure.
               * !!! THIS IS THE DEFAULT AND NEED NOT BE SPECIFIED !!!
  echo
               * Performs the trace and post-processing steps.
  echo
               * Produces a raw trace file (swtrace.nrm2) and a TPROF
  echo
  echo
                report file (tprof.out).
               * This option is mutually exclusive with -t and -p.
  echo
  echo.
  echo -t
               Run tracing only.
  echo
               * Performs only the tracing (ie. data collection)
                step of the TPROF procedure.
  echo
               * Produces a raw trace file (swtrace.nrm2).
  echo
  echo
               * The trace will not be post-processed. You can
                post-process the trace at a later time by invoking
  echo
                run.tprof with the -p option.
  echo
               * This option is mutually exclusive with -a and -p.
  echo
  echo.
  echo -p
               Run post-processing only.
               * Performs only the post-processing step of the TPROF
  echo
```

echo	procedure.
echo	* It requires a trace file (swtrace.nrm2) and assumes
echo	that you've already done the tracing step, either by
echo	itself (via a previous -t) or in a previous
echo	complete run (via -a).
echo	* Produces a TPROF report file (tprof.out).
echo	* This option is mutually exclusive with -a and -t.
echo.	
echo	-b ## Set trace buffer size to ## MB **PER PROCESSOR**
echo	* You can also set this value using the
echo	IBMPERE TPROF BLIEFER SIZE environment variable
echo	in the toolsenv cmd configuration file
echo	
echo	-m mode Set the TPROF mode
echo	* Valid modes are:
echo	- TIME
echo	This causes TPROF to run in TIME-based mode
echo	In this mode sampling is based on time and the time
echo	hetween samples is constant. The sampling rate
echo	can set via the -c ontion
echo	
echo	This causes TPROF to run in FV/FNT-based mode
ocho	In this mode compling is based on event counts and
echo	the time between camples is not constant. The campling
echo	avent count can set via the contiant. The sampling
echo	* Default mode is time
echo	* You can also get this value using the
echo	IDUCATION SEL TINS VALUE USING THE
echo	IDMFERF_IFROF_MODE environment variable in the
echo	tooisenv.chu conngulation nie.
echo.	a event. Set event nome when in $EV/ENT$ based made
echo	-e event Set event hame when in Event Jobsed mode.
echo	event is the name of an event. Valid event names
ecno	can be obtained by entring the SWIRACE EVENT LIST
ecno	command.
echo	- Event names are not case sensitive. INSTR is the same
echo	As most count can be actuained the coentian
ecno	- An event count can be set using the -c option.
echo	* Very easy also part this value using the
ecno	You can also set this value using the
ecno	IBMPERF_IPROF_EVENT environment variable in the
ecno	toolsenv.cmd configuration file.
ecno.	
ecno	-C ## Set tick rate (TIME-based) or event count (EVENT-based).
ecno	"If mode is TIME then ## represents the desired TPROF tick
ecno	rate, in ticks/second.
ecno	- Default tick rate is 128 ticks/second.
ecno	- You can also set this value using the
ecno	IBMPERF_IPROF_IICK_RATE environment variable in the
echo	toolsenv.cmd configuration file.
echo	fit mode is EVENT then ## represent the number of events
ecno	atter which a TPROF tick will occur.
echo	- Default event count is 10,000,000 events.
echo	- You can also set this value using the
echo	IBMPERF_IPROF_EVENIONI environment variable in the
echo	toolsenv.cmd configuration file.
echo	Setting the rate too high may affect overall system
ecno	performance.

echo		* Setting the rate too low may not collect enough samples.
echo.		
echo	-s ##	Automatically start tracing after ## seconds.
echo		* You WILL NOT be prompted for when to start tracing.
echo		Instead, tracing will start automatically after a
echo		delay of ## seconds.
echo		* If you DO NOT specify the -r option, you will be
echo		prompted for when to stop tracing.
echo		* If you specify 0 (zero), or a non-numeric value.
echo		tracing is started immediately.
echo		* If you use this option your application should be
echo		started and warmed up by the time tracing is started.
echo		* Only applicable with -a or -t. Ignored otherwise.
echo		* Default is to prompt for when to start tracing.
echo.		
echo	-r ##	Run (trace) for ## seconds
echo	•	* You WILL NOT be prompted for when to stop tracing
echo		Instead, tracing will stopt automatically after a
echo		delay of ## seconds from the time tracing was started
echo		* If you DO NOT specify the -s option, tracing is stopped
echo		## seconds after you press ENTER to start the trace
echo		* If you specify $\Omega$ (zero) or a pop-numeric value
echo		tracing is stopped immediately after being started
echo		* If you use this option your application should
echo		complete the scenario you want to trace in the
echo		amount of time you specify with this option
echo		* Only applicable with a or at lapored otherwise
ocho		* Default is to prompt for when to stop tracing
		Default is to prompt for when to stop tracing.
~~~~		
ecno.	Notos	
echo l	Notes:	
echo echo echo	Notes:	and are parted from left to right. If you specify an
echo l echo - echo - echo	Notes: • * Opti	ons are parsed from left to right. If you specify an
echo echo echo echo	Notes: * Opti optic	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used.
echo echo echo echo echo echo	Notes: * Opti optic * -a, -	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options:
echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed.
echo. echo echo echo echo echo echo echo	Notes: * Optic optic * -a, - only * -b, -	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd'
echo. echo echo echo echo echo echo echo echo	Notes: * Optic optic * -a, - only * -b, - envi	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables.
echo. echo echo echo echo echo echo echo echo	Notes: * Optio optic * -a, - only * -b, - envi * The	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode.
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode.
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s al	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed a rate (ticks/second) in TIME-based mode, or an event at in EVENT-based mode. ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s al proc	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed a rate (ticks/second) in TIME-based mode, or an event it in EVENT-based mode. ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s al proc scart	ions are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want
echo. echo echo echo echo echo echo echo echo	Notes: * Optic optic * -a, - only * -b, - envi * The to be cour * -e is * -s an proc sc Tl * V-lite	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed a a rate (ticks/second) in TIME-based mode, or an event at in EVENT-based mode. is ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option).
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s all proc scer to TI * Value	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. is ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations:
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scerto TI * Valie No co	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event at in EVENT-based mode. is ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scerto TI * Valion No co	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. is ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace.
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scerto TI * Valia No co -a	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. is ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options.
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scert to TI * Valie No co -a	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r.
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scern to TI * Valie No co -a -t	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event nt in EVENT-based mode. ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing.
echo. echo echo echo echo echo echo echo echo	Notes: * Option option * -a, - only * -b, - envin * The to be cour * -e is * -s an proce scerto to TI * Valie No co -a -t	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event nt in EVENT-based mode. ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing. Trace start/stop controlled by -s/-r.
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s al proc scer to TI * Valie No co -a -t -p	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed a rate (ticks/second) in TIME-based mode, or an event at in EVENT-based mode. ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing. Trace start/stop controlled by -s/-r. Post-process only. No tracing.
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s al proc scer to TI * Valie No co -a -t -p	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event at in EVENT-based mode. ignored if the current mode is TIME. and -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing. Trace start/stop controlled by -s/-r. Post-process only. No tracing.
echo. echo echo echo echo echo echo echo echo	Notes: * Opti optic * -a, - only * -b, - envi * The to be cour * -e is * -s an proc scer to TI * Valie No c -p No -	ons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event in EVENT-based mode. is ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF redure and have an idea of how long it takes for the nario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing. Trace start/stop controlled by -s/-r. Post-process only. No tracing.
echo. echo echo echo echo echo echo echo echo	Notes: * Optic optic * -a, - only * -b, - envi * The to be cour * -e is * -s an proc scer to TI * Valie No c -a -t -p No - -s ##	Cons are parsed from left to right. If you specify an on more than once, the rightmost option is used. t and -p are exclusive options: one of them is allowed. m, -c and -e override the values set in 'toolsenv.cmd' ronment variables. meaning of -c varies depending on the mode. It is assumed e a rate (ticks/second) in TIME-based mode, or an event of in EVENT-based mode. is ignored if the current mode is TIME. nd -r are useful if you are automating the TPROF wedure and have an idea of how long it takes for the hario to start up (-s option) and how long you want PROF for (-r option). d option combinations: options Run entire procedure. Prompt to start and stop trace. Same as no options. Trace start/stop controlled by -s/-r. Trace only. No post-processing. Trace start/stop controlled by -s/-r. Post-process only. No tracing.

echo -r ## Prompt to start tracing. Automatically stop echo tracing ## seconds after tracing started. echo -s ## -r ## Automatically start tracing in ## seconds. Automatically stop tracing ## seconds after echo echo tracing started. * Windows batch language does not provide a way to check echo whether a value is a valid number or not. As such, the echo values entered with the -s, -r, -b and -c options are not echo checked. A non-numeric value is the same as entering 0 (zero). echo echo * run.tprof will not run if environment variable IBMPERF TOOLS PATH isn't set. echo echo * All customization should be done via TOOLSENV.CMD. You should echo not need to modify this command file. echo. echo Examples: echo ----echo 1) run.tprof Runs the entire TPROF procedure. You are prompted to start echo echo and stop the trace. Generates TPROF report. Same as: 'run.tprof -a' echo echo. echo 2) run.tprof -s 10 -r 60 Runs the entire TPROF procedure. Tracing starts automatically echo after a 10 seconds delay. Tracing stops automatically 60 echo seconds after being started. Generates TPROF report. There echo echo are no prompts - everything happens automatically. echo Same as: 'run.tprof -a -s 10 -r 60' echo. echo 3) run.tprof -t Runs the trace step of the TPROF procedure. You are echo echo prompted to start and stop the trace. Does not generate a echo TPROF report. echo. echo 4) run.tprof -t -s 10 -r 60 Runs the trace step of the TPROF procedure. Tracing starts echo echo automatically after a 10 seconds delay. Tracing stops automatically 60 seconds after being started. There are no echo echo prompts - everything happens automatically. echo. echo 5) run.tprof -t -s 10 Runs the trace step of the TPROF procedure. Tracing starts echo echo automatically after a 10 seconds delay. You are prompted for echo when to stop tracing. echo. echo 6) run.tprof -p echo Runs the post-processing step of the TPROF procedure. echo You are prompted to start and stop the trace. Only generates echo a TPROF report. echo. echo 7) run.tprof -m event -e l2 read refs -c 100000 s 10 -r 60 echo Runs the entire TPROF procedure. Sampling is EVENT-based, using event L2_READ_REFS and sampling every 100,000 occurences. echo Tracing starts automatically after a 10 seconds delay. Tracing echo echo stops automatically 60 seconds after being started. Generates echo TPROF report. There are no prompts - everything happens echo automatically. echo.

goto Exit

Rem Command line parsing :CheckForHelp Rem ** Rem ** See if they want help Rem ** SET RUN TPROF HELP= if "%1" == "?" SET _RUN_TPROF_HELP=short if "%1" == "-?" SET _RUN_TPROF_HELP=short if "%1" == "/?" SET RUN TPROF HELP=short if "%1" == "??" SET _RUN_TPROF_HELP=long if "%1" == "-??" SET_RUN_TPROF_HELP=long if "%1" == "/??" SET_RUN_TPROF_HELP=long if not "%_RUN_TPROF_HELP%" == "" goto Help Rem ** Rem ** Set defaults for command line options Rem ** SET _TPROF_START_DELAY= SET _TPROF_RUN_TIME= SET TPROF RUN TYPE= SET _TPROF_BUFSIZE= SET _TPROF_MODE= SET_TPROF_EVENT= SET _TPROF_CNT= Rem ** Rem ** Parse command line Rem ** :ParseCmdLine if "%1" == "" goto DoneParsing if "%1" == "-a" goto HandleAll if "%1" == "-A" goto HandleAll if "%1" == "-t" goto HandleTrace if "%1" == "-T" goto HandleTrace if "%1" == "-p" goto HandlePost if "%1" == "-P" goto HandlePost if "%1" == "-s" goto HandleS if "%1" == "-S" goto HandleS if "%1" == "-r" goto HandleR if "%1" == "-R" goto HandleR if "%1" == "-b" goto HandleB if "%1" == "-B" goto HandleB if "%1" == "-m" goto HandleM if "%1" == "-M" goto HandleM if "%1" == "-e" goto HandleE if "%1" == "-E" goto HandleE if "%1" == "-c" goto HandleC if "%1" == "-C" goto HandleC echo. echo "%1" is not a valid argument. Enter "run.tprof ?" for help.

goto Exit :ParseContinue shift goto ParseCmdLine Rem ** Rem ** Handle -a optiom Rem ** :HandleAll if not "%_TPROF_RUN_TYPE%" == "" goto All999 SET TPROF RUN TYPE=-a goto ParseContinue :All999 if "%_TPROF_RUN_TYPE%" == "-a" goto ParseContinue echo. echo run.tprof: %1 not allowed with %_TPROF_RUN_TYPE% option. Quitting. echo. goto Exit Rem ** Rem ** Handle -t optiom Rem ** :HandleTrace if not "% TPROF RUN TYPE%" == "" goto Trace999 SET _TPROF_RUN_TYPE=-t goto ParseContinue :Trace999 if "%_TPROF_RUN_TYPE%" == "-t" goto ParseContinue echo. echo run.tprof: %1 not allowed with %_TPROF_RUN_TYPE% option. Quitting. echo. goto Exit Rem ** Rem ** Handle -p optiom Rem ** :HandlePost if not "% TPROF RUN TYPE%" == "" goto Post999 SET TPROF RUN TYPE=-p goto ParseContinue :Post999 if "%_TPROF_RUN_TYPE%" == "-p" goto ParseContinue echo. echo run.tprof: %1 not allowed with %_TPROF_RUN_TYPE% option. Quitting. echo. goto Exit Rem ** Rem ** Handle -s option Rem ** :HandleS shift if "%1" == "" goto S999 set _TPROF_START_DELAY=%1 goto ParseContinue :S999 echo.

DLM

```
echo run.tprof: Missing time delay with -s option. Quitting.
   echo.
   goto Exit
Rem **
Rem ** Handle -r option
Rem **
:HandleR
   shift
   if "%1" == "" goto R999
   set TPROF RUN TIME=%1
   goto ParseContinue
:R999
   echo.
   echo run.tprof: Missing run time with -r option. Quitting.
   echo.
   goto Exit
Rem **
Rem ** Handle -b option
Rem **
:HandleB
   shift
   if "%1" == "" goto B999
   set _TPROF_BUFSIZE=%1
   goto ParseContinue
:B999
   echo.
   echo run.tprof: Missing buffer size with -b option. Quitting.
   echo.
   goto Exit
Rem **
Rem ** Handle -m option
Rem **
:HandleM
   shift
   if "%1" == "" goto M999
   if "%1" == "time" goto M100
   if "%1" == "Time" goto M100
if "%1" == "TIME" goto M100
   if "%1" == "event" goto M100
   if "%1" == "Event" goto M100
   if "%1" == "EVENT" goto M100
   echo.
   echo run.tprof: Invaid mode with -m option. Must be TIME or EVENT. Quitting.
   echo.
   goto Exit
:M100
   set _TPROF_MODE=%1
   goto ParseContinue
:M999
   echo.
   echo run.tprof: Missing mode with -m option. Quitting.
   echo.
   goto Exit
```

Alphaworks

Rem ** Rem ** Handle -e option Rem ** :HandleE shift if "%1" == "" goto E999 set _TPROF_EVENT=%1 goto ParseContinue :E999 echo. echo run.tprof: Missing event name with -e option. Quitting. echo. goto Exit Rem ** Rem ** Handle -c option Rem ** :HandleC shift if "%1" == "" goto C999 set TPROF CNT=%1 goto ParseContinue :C999 echo. echo run.tprof: Missing count with -c option. Quitting. echo. goto Exit Rem ** Rem ** Done parsing Rem ** :DoneParsing set TPROF POST RC=0 Rem Set environment Rem ** Rem ** Try running TOOLSENV.CMD and hope it can be found along PATH. Rem ** It could be found in the current directory if the user has Rem ** added it to PATH. Rem ** echo run.tprof: Attempting to run TOOLSENV.CMD (along PATH) ... call TOOLSENV.CMD > NUL 2>NUL if not "%IBMPERF TOOLS PATH%" == "" goto Start echo run.tprof: ... either not found or didn't at least set IBMPERF_TOOLS_PATH echo. Rem ** Rem ** Not found along PATH ... Rem ** Now try running TOOLSENV.CMD from the current directory. Rem **

echo. if not exist .\TOOLSENV.CMD goto TryBin echo run.tprof: Attempting to run TOOLSENV.CMD (current directory) ... call .\TOOLSENV.CMD > NUL 2>NUL if not "%IBMPERF TOOLS PATH%" == "" goto Start echo run.tprof: ... either not found or didn't at least set IBMPERF TOOLS PATH :TryBin Rem ** Rem ** Not in the current directory or didn't set IBMPERF_TOOLS_PATH ... Rem ** Now check if TOOLSENV.CMD is in 'bin' directory, in case user is Rem ** running from the tools root directory. Rem ** if not exist bin\TOOLSENV.CMD goto GiveUp cd bin echo. echo run.tprof: Attempting to run bin\TOOLSENV.CMD (bin subdirectory) ... call TOOLSENV.CMD > NUL 2>NUL if not "%IBMPERF_TOOLS_PATH%" == "" goto Start echo run.tprof: ... either not found or didn't at least set IBMPERF_TOOLS_PATH :GiveUp Rem ** Rem ** TOOLSENV.CMD nowhere to be found *OR* it was found but it did Rem ** not set IBMPERF TOOLS PATH. Either way it's an error. Rem ** echo. echo run.tprof: Environment variable IBMPERF_TOOLS_PATH is not set. echo. echo Either no copy of TOOLSENV.CMD was found or IBMPERF_TOOLS_PATH has echo not been set. You must either: echo - Change to the 'bin' subdirectory in the directory where the echo tools were installed and run run.tprof from there, or echo - Copy TOOLSENV.CMD from the 'bin' subdirectory in the directory echo where the tools were installed to this directory and then echo re-run run.tprof. echo In any case, make sure TOOLSENV.CMD has been updated correctly if echo you haven't already done so. echo If you don't have a TOOLSENV.CMD run TINSTALL and one will be echo generated for you. goto Exit :Start Rem ** Rem ** Set defaults Rem ** echo. if "%IBMPERF DATA PATH%" == "" set IBMPERF DATA PATH=%IBMPERF TOOLS PATH% if "%IBMPERF_JITA2N_PATH%" == "" set IBMPERF_JITA2N_PATH=%IBMPERF_DATA_PATH% if "%IBMPERF_JITA2N_NAME_ROOT%" == "" set IBMPERF_JITA2N_NAME_ROOT=log if "%IBMPERF_TPROF_REPORT_FILENAME%" == "" set IBMPERF_TPROF_REPORT_FILENAME=tprof.out if "%IBMPERF_TPROF_MODE%" == "" set IBMPERF_TPROF_MODE=TIME

if "%IBMPERF_TPROF_MODE%" == "" set IBMPERF_TPROF_MODE=TIME if not "%_TPROF_MODE%" == "" set IBMPERF_TPROF_MODE=%_TPROF_MODE%

if "%IBMPERF_TPROF_TICK_RATE%" == "" set IBMPERF_TPROF_TICK_RATE=128 TPROF EVENT%" == "" set IBMPERF TPROF EVENT=INSTR if "%IBMPERF if "%IBMPERF_TPROF_EVENTCNT%" == "" set IBMPERF_TPROF_EVENT=10000000 if not "% TPROF EVENT%" == "" set IBMPERF TPROF EVENT=% TPROF EVENT% if "%IBMPERF_TPROF_MODE%" == "time" set IBMPERF_TPROF_MODE=TIME if "%IBMPERF_TPROF_MODE%" == "Time" set IBMPERF_TPROF_MODE=TIME if "%IBMPERF_TPROF_MODE%" == "event" set IBMPERF_TPROF_MODE=EVENT if "%IBMPERF_TPROF_MODE%" == "Event" set IBMPERF_TPROF_MODE=EVENT if "%IBMPERF TPROF MODE%" == "TIME" (if not "% TPROF CNT%" == "" set IBMPERF TPROF TICK RATE=% TPROF CNT%) else (if not "% TPROF CNT%" == "" set IBMPERF TPROF EVENTCNT=% TPROF CNT%) if "%IBMPERF TPROF MAJOR CODES%" == "" set IBMPERF TPROF MAJOR CODES=16 25 if "%IBMPERF_TPROF_BUFFER_SIZE%" == "" set IBMPERF_TPROF_BUFFER_SIZE=3 if not "% TPROF BUFSIZE%" == "" set IBMPERF TPROF BUFFER SIZE=% TPROF BUFSIZE% set IBMPERF TPROF POST OPTIONS=-off -clip 0 -show -a2nrdup -a2nmmi **MMI** if "%IBMPERF SCREEN LINES%" == "" set IBMPERF SCREEN LINES=0 if "%IBMPERF REMOVE JITA2N FILES%" == "yes" set IBMPERF REMOVE JITA2N FILES=YES if "%IBMPERF REMOVE JITA2N FILES%" == "YES" set IBMPERF REMOVE JITA2N FILES=YES if "%IBMPERF REMOVE JITA2N FILES%" == "no" set IBMPERF REMOVE JITA2N FILES=NO if "%IBMPERF_REMOVE_JITA2N_FILES%" == "NO" set IBMPERF_REMOVE_JITA2N_FILES=NO if "%IBMPERF REMOVE JITA2N FILES%" == "" set IBMPERF REMOVE JITA2N FILES=YES if "% LP%" == "" set LP=%IBMPERF JITA2N PATH%\%IBMPERF JITA2N NAME ROOT% Rem ** Rem ** Change screen size if required Rem ** if %IBMPERF_SCREEN_LINES% EQU 0 goto NoSizeChange @echo on MODE CON: LINES=%IBMPERF SCREEN LINES% @echo off :NoSizeChange Rem Display environment Rem ** Rem ** Display what we're using Rem ** echo. echo IBMPERF_TOOLS_PATH = %IBMPERF_TOOLS_PATH% echo IBMPERF_DATA_PATH = %IBMPERF_DATA_PATH% echo IBMPERF_SYMBOLS_PATH = %IBMPERF_SYMBOLS_PATH% echo IBMPERF TPROF BUFFER SIZE = %IBMPERF TPROF BUFFER SIZE%

echo IBMPERF_TPROF_MODE = %IBMPERF_TPROF_MODE% if "%IBMPERF TPROF MODE%" == "TIME" (echo IBMPERF_TPROF_TICK_RATE = %IBMPERF TPROF TICK RATE%) else (echo IBMPERF_TPROF_EVENT = %IBMPERF TPROF EVENT% = %IBMPERF_TPROF_EVENTCNT% echo IBMPERF_TPROF_EVENTCNT) echo IBMPERF_TPROF_MAJOR_CODES = %IBMPERF_TPROF_MAJOR_CODES% echo IBMPERF_TPROF_POST_OPTIONS = %IBMPERF_TPROF_POST_OPTIONS% echo IBMPERF_TPROF_REPORT_FILENAME = %IBMPERF_TPROF_REPORT_FILENAME% echo IBMPERF JITA2N PATH = %IBMPERF JITA2N PATH% echo IBMPERF JITA2N NAME ROOT = %IBMPERF JITA2N NAME ROOT% echo IBMPERF_REMOVE_JITA2N_FILES = %IBMPERF_REMOVE_JITA2N_FILES% = %IBMPERF_SCREEN LINES% echo IBMPERF SCREEN LINES if not "% TPROF START DELAY%" == "" echo TPROF Start Delay = % TPROF START DELAY% Seconds if not "%_TPROF_RUN_TIME%" == "" echo TPROF Run Time = % TPROF RUN TIME% Seconds echo Current Directory = %CD% echo Current Date and Time = %DATE% at %TIME% if "%_TPROF_RUN%" == "trace" echo ***** Running trace portion only ***** if "%_TPROF_RUN%" == "post" echo ***** Running post-processing portion only ***** echo. Rem Do the work Rem ******* Rem ** Rem ** If they only want to post-process then go do that Rem ** if "% TPROF RUN TYPE%" == "-p" goto SwtraceOff :Trace Rem ** Rem ** Get ready to trace ... Rem ** echo. echo ** echo ** Initializing SWTRACE ... echo ** @echo on SWTRACE INIT /S %IBMPERF TPROF BUFFER SIZE% @echo off if not "%errorlevel%" == "99" goto Continue100 echo. echo Looks like PERFDD.SYS isn't loaded. echo Did you run \"TINSTALL\" successfully? goto Exit if not "%errorlevel%" == "0" goto SwtraceError :Continue100 if "%IBMPERF_TPROF_MODE%" == "EVENT" goto SetEvent if "%IBMPERF_TPROF_MODE%" == "event" goto SetEvent if "%IBMPERF TPROF MODE%" == "Event" goto SetEvent

Rem ** Rem ** Set up for TIME-based profiling Rem ** @echo on SWTRACE SETRATE %IBMPERF_TPROF_TICK_RATE% @if not "%errorlevel%" == "0" @goto SwtraceError goto Continue110 :SetEvent Rem ** Rem ** Set up for EVENT-based profiling Rem ** @echo on SWTRACE EVENT %IBMPERF_TPROF_EVENT% -c %IBMPERF_TPROF_EVENTCNT% @if not "%errorlevel%" == "0" @goto SwtraceError :Continue110 @echo on SWTRACE ENABLE % IBMPERF_TPROF_MAJOR_CODES% @if not "%errorlevel%" == "0" @goto SwtraceError @echo off Rem ** Rem ** Delete jita2n* and jtnm* files if needed Rem ** if "%IBMPERF REMOVE JITA2N FILES%" == "NO" goto Continue120 @echo on ERASE %_LP%-jita2n* > NUL 2>NUL ERASE %_LP%-jtnm* > NUL 2>NUL @echo off :Continue120 @echo off Rem ** Rem ** Remove other log* files if needed Rem ** if "%IBMPERF JPROF LOG FILES TO REMOVE%" == "" goto Continue125 @echo on ERASE %IBMPERF_JPROF_LOG_FILES_TO_REMOVE% > NUL 2>NUL @echo off :Continue125 Rem ** Rem ** About to start tracing ... Rem ** @echo off if "%_TPROF_START_DELAY%" == "" goto StartPrompt echo. echo ***** echo ***** Will automatically start tracing in %_TPROF_START_DELAY% seconds ... echo ***** if not "%_TPROF_START_DELAY%" == "0" ASK /z %_TPROF_START_DELAY% echo. echo Trace started at %TIME% ... goto SwtraceOn

:StartPrompt @echo off echo. echo Get your application loaded and running ... echo. echo If TPROFing a java application you must invoke java using the '-Xrun' echo option. This allows jitted method information to be obtained. echo. echo Invoke java as follows: echo java -Xrunjprof:jita2n,threadinfo,fnm=%_LP%,pidx AppName echo. echo. echo ***** echo ***** @rem ASK /w ***** Press ENTER when ready to begin tracing or Ctrl-C to quit ... echo. echo Trace started at %TIME% ... :SwtraceOn @echo on SWTRACE ON @if not "%errorlevel%" == "0" goto SwtraceError @echo off Rem * * Rem * User is now tracing ... Rem * @echo ##_done_start_tools## if "%_TPROF_RUN_TIME%" == "" goto ExitStart echo. echo ***** echo ***** Will automatically stop tracing in %_TPROF_RUN_TIME% seconds ... echo ***** echo. ASK /z %_TPROF_RUN_TIME% goto SwtraceOff :StopPrompt echo. echo ***** echo ***** ASK /w ***** Tracing. Press ENTER when ready to stop ... :SwtraceOff @echo on SWTRACE OFF @if not "%errorlevel%" == "0" goto SwtraceError @echo off echo.

echo ** echo ** Collecting trace ... echo ** echo. echo Trace ended at %TIME% ... @echo on SWTRACE GET %IBMPERF_DATA_PATH%\swtrace.nrm2 @if not "%errorlevel%" == "0" goto SwtraceError SWTRACE DISABLE SWTRACE FREE @echo off Rem ** Rem ** If they only wanted to trace then we're done Rem ** if "% TPROF RUN TYPE%" == "-t" goto Continue160 Rem ** Rem ** This is where post-processing begins ... Rem ** :PostProcess SET TPROF POST JDIR=-jdir % LP% if exist % LP%-jita2n* goto Continue140 if exist %_LP%-jtnm* goto Continue140 echo. echo No jita2n* or jtnm* file(s) found. Assuming no java ... SET TPROF POST JDIR= :Continue140 @echo off echo. echo ** echo ** Generating TPROF report ... echo ** if "%IBMPERF_SYMBOLS_PATH%" == "" goto Continue145 @echo on SET A2N SEARCH PATH=%IBMPERF SYMBOLS PATH% @echo off :Continue145 @echo on @echo IBMPERF DATA PATH: %IBMPERF DATA PATH% @echo IBMPERF TOOLS PATH: %IBMPERF TOOLS PATH% POST -off -r %IBMPERF_DATA_PATH%\swtrace.nrm2 %_TPROF_POST_JDIR% %IBMPERF_TPROF_POST_OPTIONS% @set TPROF POST RC=%errorlevel% @CALL mergetprof.cmd tprof_e.out @rem if "%IBMPERF DATA PATH%" == "%CD%" @goto Continue150 @rem MOVE /Y tprof.out %IBMPERF_DATA_PATH%\%IBMPERF_TPROF_REPORT_FILENAME% @rem MOVE /Y tprof_e.out %IBMPERF_DATA_PATH%\%IBMPERF_TPROF_REPORT_FILENAME% @rem ** @rem ** Copy all the other post junk to the data directory @rem ** @rem MOVE /Y a2n.err %IBMPERF DATA PATH%\. > NUL 2>NUL @rem @MOVE /Y a2n.mod %IBMPERF DATA PATH%\. > NUL 2>NUL

```
@rem @MOVE /Y a2n.proc %IBMPERF_DATA_PATH%\. > NUL 2>NUL
 @rem @MOVE /Y post.msg %IBMPERF DATA PATH%\. > NUL 2>NUL
 @rem @MOVE /Y ptree %IBMPERF_DATA_PATH%\. > NUL 2>NUL
 @rem @MOVE /Y dbfile %IBMPERF_DATA_PATH%\. > NUL 2>NUL
 @rem @MOVE /Y arc
                    %IBMPERF DATA PATH%\. > NUL 2>NUL
:Continue150
 @echo off
Rem **
Rem ** Got here because were were doing -a or -p
Rem **
 if not "% TPROF POST RC%" == "0" goto PostError
 echo.
      echo 3
 echo TProf report "%IBMPERF_DATA_PATH%\%IBMPERF_TPROF_REPORT_FILENAME%" generated.
 @echo ##_done_get_the_file##
 goto Exit
:Continue160
 @echo off
Rem **
Rem ** Got here because were were doing -t
Rem **
 echo.
      echo *
 echo Trace file "%IBMPERF DATA PATH%\swtrace.nrm2" generated.
 goto Exit
:PostError
 @echo off
 echo.
 echo **ERROR** **ERROR** **ERROR**
 echo POST command error. TProf report may not have been generated.
 echo.
 echo - Look for report file "%IBMPERF_DATA_PATH%\%IBMPERF_TPROF_REPORT_FILENAME%".
 echo - Go over all screen output.
 echo - Look in file "%IBMPERF DATA PATH%post.msg" for additional
 echo error information.
 echo **ERROR** **ERROR** **ERROR**
 echo.
 goto Exit
:SwtraceError
 @echo off
 echo.
 echo **ERROR** **ERROR** **ERROR**
 echo SWTRACE command error. Quitting.
 echo **ERROR** **ERROR** **ERROR**
 echo.
 goto Exit
:Exit
 set _TPROF_START_DELAY=
 set TPROF RUN TIME=
 set TPROF RUN TYPE=
```

set _TPROF_POST_JDIR= set _TPROF_BUFSIZE= set _TPROF_MODE= set _TPROF_EVENT= set _TPROF_CNT= set _TPROF_POST_RC=

@echo ##_done_stop_tools##

:ExitStart @endlocal