



Collecting and Analyzing PEX Trace Profile Data



Jeremy Arnold

IBM Rochester Lab

September 2002

Performance Explorer

While Performance Explorer (PEX) is generally used for advanced performance analysis, some PEX functionality is easy to use. Collecting and analyzing trace profile data is a relatively easy way to find out where the CPU time is being spent when running your application. (Note: To analyze PEX data requires that the PT1 LPP (Performance Tools Licensed Program Product) be installed on your system. It's possible to collect PEX data without PT1 installed, but the Print PEX Report (PRTPEXRPT) command, which allows the data to be printed in a readable format is only available as part of PT1.)

The first step in collecting Trace Profile data is to create a PEX definition. The definition for a Trace Profile prior to V5R2 can be created as:

```
ADDPEXDFN DFN(TPROF5) TYPE(*TRACE) JOB(*ALL) TASK(*ALL)
          MAXSTG(100000) INTERVAL(5) TRCTYPE(*SLTEVT) SLTEVT(*YES)
          BASEVT(( *PMCO))
```

In V5R2, the definition was simplified a bit:

```
ADDPEXDFN DFN(TPROF5) TYPE(*PROFILE) PRFTYPE(*JOB) JOB(*ALL)
          TASK(*ALL) MAXSTG(100000) INTERVAL(5)
```

I won't explain each parameter in these commands, but there are a couple you may wish to modify. For instance, if you have multiple JVMs running on the system, it may be helpful to specify your job name in the JOB parameter (instead of *ALL). This tells PEX to collect events for just the job you're interested in. The INTERVAL(5) parameter specifies that an event should be generated for every 5 milliseconds of CPU time. Using large values (like 10 to 20) reduces PEX's overhead, but requires a longer collection to get accurate data. Smaller values (like 1 to 5) have a bit more overhead, but can be used for shorter collections. I typically start with 5 milliseconds on large systems (4-way or above) or 1 millisecond on smaller systems and use longer intervals only if the overhead appears to be impacting the run. The "5" in the definition name ("TPROF5") matches the interval--this is simply a convention to remember what the definition is for.

To start a PEX collection, once your workload is running in a steady state, issue the following command:

```
STRPEX SSNID(mytprof) DFN(TPROF5)
```

The SSNID parameter specifies the session ID, which is the name of the collection and should be named something you can remember.

End PEX by issuing:

```
ENDPEX SSNID(*SELECT)
```

Because this uses the default session ID, "*SELECT," you'll see a list of the currently active PEX collections, which should include your collection. It'll also show the current event count. Press F5 to refresh the count. You should generally try to collect around 100,000 events. (Note: More events may result in more accurate data if you're printing reports by *PROCEDURE.) This could take

anywhere from a couple of minutes to about 20 minutes, depending on system size, activity and the interval specified on the PEX definition. Once the necessary number of events has been collected, select your session to end it. It may take a couple of minutes to end.

Next, print the data using Print PEX Report (PRTPEXRPT) and specifying the session ID used in the STRPEX command for the "MBR" parameter:

```
PRTPEXRPT MBR(mytprof) TYPE(*PROFILE) PROFILEOPT(*SAMPLECOUNT *PROGRAM)
```

In some cases you may want to use "*MODULE" or "*PROCEDURE" in place of "*PROGRAM." Printing the report using *PROGRAM shows the amount of CPU time spent in each program running on the system. Printing using *MODULE or *PROCEDURE summarizes by module or procedure level instead. Examining the profile at a program level can be helpful for getting an overall view of system behavior, while examining the procedure level can help identify more specific problems.

The report will be printed to a spool file called "QPVPERPT." The report contains multiple sections, which can make it difficult to find the section you're interested in. For trace profiles, the most useful section is the actual profile data. The easiest way to find this is to search for "Histogram." This will appear several times at the beginning of the file in the Library section, but if you search again you should reach the desired information. To verify that you're in the correct section, examine the Map Flag column, which should always be "= =" or "+ +".

Here is a description of the various columns:

Histogram	This gives a graphical view of which programs/modules/procedures are taking the most CPU time.
Hit Cnt	This tells how many "hits" PEX had on this program/module/procedure. The "Total Hits" is included in the summary information near the top of the report.
Hit %	This gives the number of hits as a percentage of the total number of hits. This is generally the most useful column to examine.
Cum %	This column gives the hit percentage of this program/module/procedure and all of the ones above it.
Start Addr	This is the address of the program/module/procedure. This isn't generally useful.
Map Flag	This will be "= =" for code running above the MI (XPF and application code), and "+ +" for code running below the MI (LIC code).
Stmt Nbr	The statement number isn't generally useful for Java programs.
Name	This is the name of the program/module/procedure.

Resources

More information on running PEX can be found online (<http://www.ibm.com/series/perfmgmt/resource.htm>, select "Performance Tools for iSeries Reference" (SC41-5340) and click on Chapter 11 "Performance Explorer.")

About the Author

Jeremy Arnold is a staff software engineer with the IBM Server Group in the Rochester (Minn.) Lab. Jeremy specializes in Java and WebSphere performance. He can be reached at arnoldje@us.ibm.com.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM iSeries PEX. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

©Copyright International Business Machines, 2002. All rights reserved. This paper last revised on: September 17, 2002.