

Objectives/Overview

Objective

Enable Performance Engineering

- By Application behavior characterization and profiling
- By Performance measurement
- By Performance observation at Method and Line Level

Local and Distributed Java Performance Engineering

Enable Adopters to extend TPTP Performance Tools

TPTP Java Profiling

- Broadly useful for performance analysis and for gaining a deeper understanding of a Java program
- Passively collect performance data of your application
- Enables you to profile and interact with your applications, to work with profiling resources, and to examine your applications for performance and memory usage problems
- Help you to visualize and understand your program execution, pinpoint the operations that take the most resource, as well as to explore patterns of program behavior
- Enables you to test your application's performance early in the programming development cycle for improvements
- TPTP Java Profiler currently supports JVMPI interface.
- Next Generation Java profiler available as Technology Preview June, 2006.

Java Performance Tools in TPTP

TPTP Java Performance Tools

Profiling

Native Interface

- Java VM Profiling Interface (JVMPI)
- Java VM Tooling Interface (JVMTI)

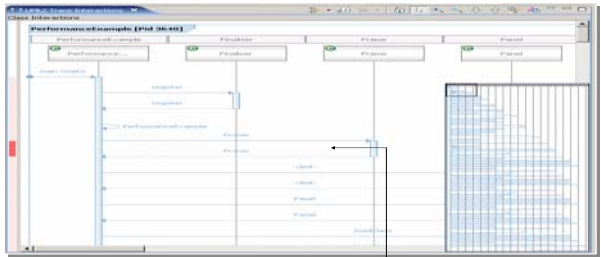
Java Management Extensions

- JMX Instrumentation

Java Byte-Code Manipulation

- Static Instrumentation (Probekit)
- Dynamic Instrumentation (Dynamic Probekit)

Reconstruct application flow from events



Measure method execution

TPTP Java Profiler

- TPTP Java Profiler is used to identify performance details such as classes or methods responsible for the poor execution performance, also be used to analyze application heap and find memory leaks
- Applications under test can reside in Eclipse workspace or binaries on file system
- Profile
- Visual analysis of performance data
- Performance measurement data collection mode
 - Aggregate
 - Fine grained

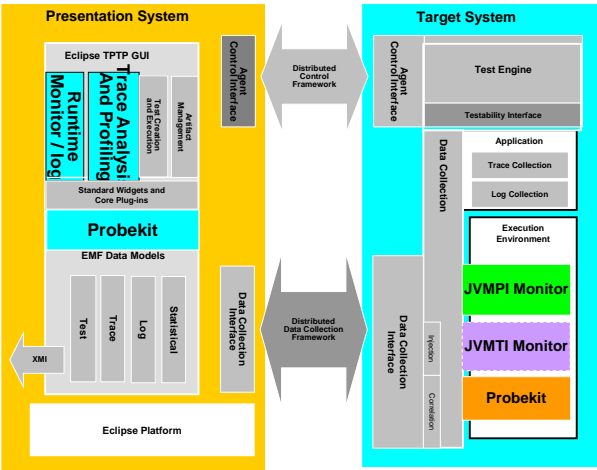
Java Performance Engineering

- Profile and measure performance of your application
- Observe Performance at Module and Line Level
- Optimize Poor performing code
- Tune your application
 - Increase Production code performance
 - Mitigate Performance issues

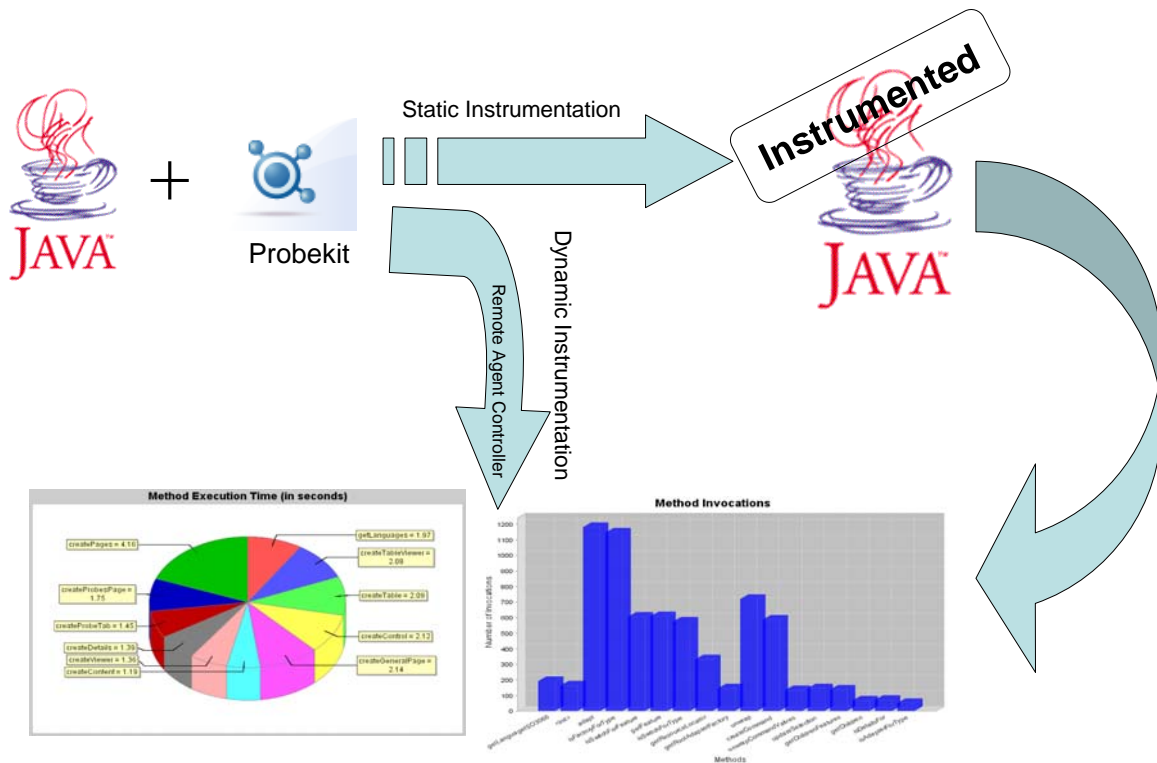
Generate analysis reports of your application performance



Eclipse TPTP Architecture



TPTP Probekit



Runtime Data on the Instrumented Application

What is Probekit?

Probekit is a framework included in the TPTP project, which provides the tools for writing probes and applying them to Java applications. Probes are reusable Java code fragments which can be used to collect runtime information.

Why use Probekit?

The runtime data collected from an application can be used for a wide variety of things such as:

- Profiling
- Debugging
- Reverse Engineering

Please see <http://www.eclipse.org/tptp> for more information

Future of TPTP

Java 5+ Profiling

- Based on Java Virtual Machine Tooling Interface (JVMTI)
- Profile your applications dynamically at runtime using JVMTI with minimal intrusion
- Sampling and Instrumentation of the JVM
- Technology Preview – June, 2006
 - Profile your application in Total or Selectively by applying filters
- General Availability - December, 2006
 - Heap Profiling
 - Thread Profiling
 - Extensive Platform coverage
- Future
 - Profile and Debug in a single flow
 - Java 6 support

Low Overhead Data Collection

Statistical Sampling

- Sample your Java application and the JVM
- Data collection is interval based
- Good for measuring CPU usage of methods

Build to Manage Toolkit

- A set of tools for managing/monitoring Java applications
- Provides JMX/CBE/ARM instrumentation

Dynamic Probekit

- Instrument classes or JARs on-the-fly with minimum overhead
- Instrumentation is done in memory
- No clean up work required at the end because .class files are not modified
- Available in TPTP 4.2

Line Level Coverage

- Indicates the hit and missed lines during a profiling session
- Useful for determining whether JUnit test cases provide 100% coverage
- Available in TPTP 4.2