Using the Eclipse TPTP Java Profiler

Chris Elford (chris.l.elford@intel.com)
Content cheerfully stolen from presentations by many others
(Most from EclipseCon2007)

December 10, 2007
Portland, OR

Eclipse TPTP Overview

- An Eclipse top level project since 2004
  - Rooted in earlier Hyades project started in 2002

- TPTP Participated in Calisto, Europa releases
  - Participating in Ganymede as well

- Composed of 4 sub projects:
  - **Platform**
  - **Trace and Profiling**
  - Test
  - Monitoring

$\left.\begin{matrix}\text{Platform}\\\text{Trace and Profiling}\end{matrix}\right\}$ TPTP Profiler developed/maintained by these teams

Eclipse TPTP Profiling Tool Overview

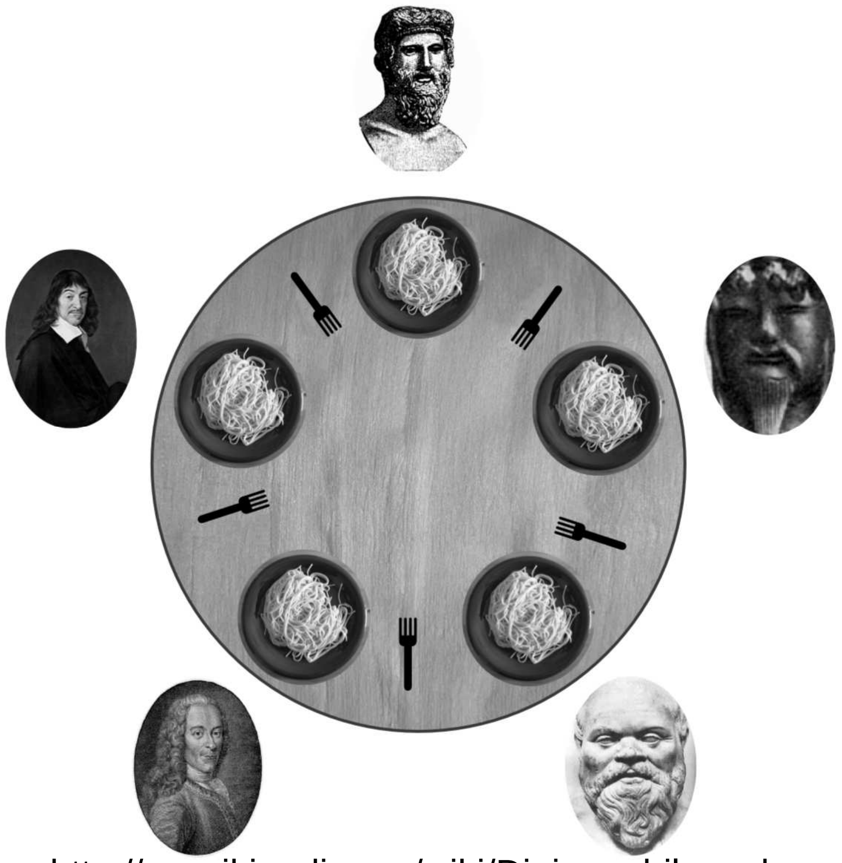
- Broadly useful for performance analysis and for gaining a deeper understanding of a Java program
- Consists of the Profiling and Logging Perspective and a number of graphical and tabular views
- Help you to visualize and understand your program execution and threading behavior, pinpoint the operations 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

The New Java Profiler

- Set of libraries to attach to JVM & record Java App behavior
- Extensible framework: core runtime component (Martini); agent managed by the Agent Controller (JPIAgent); set set of data collection libraries built on top of the Martini runtime.
- Identify performance details (e.g.,): classes or methods responsible for execution bottlenecks; analyze application heap to find memory leaks; visualize threading behavior.
- Output in the form of XML fragments (XML4Profiling)
- Can be launched from the Eclipse IDE or as a standalone program using Java command-line options
- Applications under test can reside in Eclipse workspace, binaries on file system, or hosted in a J2EE application server.

Thread Profiler Demo

- Classic Dining Philosophers problem
- 5 Philosophers around a table with 5 forks between them.
- Three basic states
  - Thinking
  - (Hungry) Want to eat (acquire 2 locks)
    - Acquire left fork first then right fork
  - Eating (then release locks)
- Deadlock opportunity
- Many solutions yield live lock



http://en.wikipedia.org/wiki/Dining_philosophers

Execution Time Demo

- Read and internalize a product catalog
- 24 Products

```
<?xml version="1.0" encoding="UTF-8"?>
<content>
        <product name="Apple" price="1.99" type="Golden" production="Canada"/>
</content>
```

- Very quick execution using “Default” filters
  - Ignore core java classes; concentrate on application
    - Limits overhead of collection
  - Rich controls over filtering allowed at different times
    - data collection time; data load time; data analysis time
  - A statistical mode also available (less overhead;less rich)
- Lets quickly see where time goes

TPTP Resources (profiling and beyond)

- Learn and Try
  - http://www.eclipse.org/tptp/home/documents/conferences/eclipseCon2007
  - http://www.eclipse.org/tptp/home/documents/conferences/eclipseCon2007/%283669%29%20Profiling%20Java%20applications%20using%20Eclipse%20TPTP%20v1_1.htm
- Webs and Wikis
  - http://eclipse.org/TPTP
  - http://wiki.eclipse.org/TPTP
- Downloads and Updates
  - http://www.eclipse.org/tptp/home/downloads
  - http://www.eclipse.org/tptp/home/downloads/updateManager.php
- News and Mail
  - http://www.eclipse.org/tptp/home/project_info/general/mailnews.php
- Use and Participate
  - http://wiki.eclipse.org/TPTP_User_Experiences_Profiling
  - http://www.eclipse.org/tptp/home/project_info/general

- backup

Limitations

- Data volume
  - With full details and larger apps, a great amount of data can be generated; overloading the perspectives on 32bit platforms
  - Data currently stored in memory for view creation
    - Will eventually resolve this.
  - Clever filtering solutions
  - Clever use of Pause/resume
- Overhead of data collection (peturbation)
  - Statistical mode
  - Clever filtering solutions
  - Binary instead of just XML (Ganymede feature)
- Ease of use – some users find interfaces confusing
  - Trying to improve with user feedback…
  - http://wiki.eclipse.org/TPTP_User_Experiences_Profiling

The New Java Profiler (JVMTI) vs. JVMPI

- Java 5.0 has introduced new standards for profiling
  - Prior to Java 5.0 (Java 1.4-) the standards and the interfaces to support profiling (JVMPI) were experimental
  - The new standard (JVMTI) is an innovative solution to profiling and enables you control precisely which parts of an application are profiled
  - Eclipse TPTP has embraced this innovation and the new Java profiler is based on this new standard
- Old Java profiler based on (JVMPI) is still available for backward compatibility
  - It will be supported in Java 5.0, but is not supported on Java 6.0+
  - The JVMPI based profiler provides numerous features currently not available in the new Java Profiler
- The new Java profiler will eventually replace the JVMPI profiler as the standard TPTP profiler