parallel tools platform

developers workshop

Oak Ridge National Laboratory
23-24 May 2007

Agenda

- Wednesday 5/23
  - 09:00 - 09:30 Opening Remarks/Welcome
  - 09:30 - 10:00 Workshop Overview
  - 10:00 - 12:00 Session 1 : System and Core Services (including break)
  - 12:00 - 13:00 Lunch
  - 13:00 - 14:00 Session 1 : Continued
  - 14:00 - 17:00 Session 2 : Performance Tools (including break)
  - 17:00 - 17:30 Day 1 Wrap Up
- 18:00 Dinner
  - Calhoun's at the Marina in Lenoir City

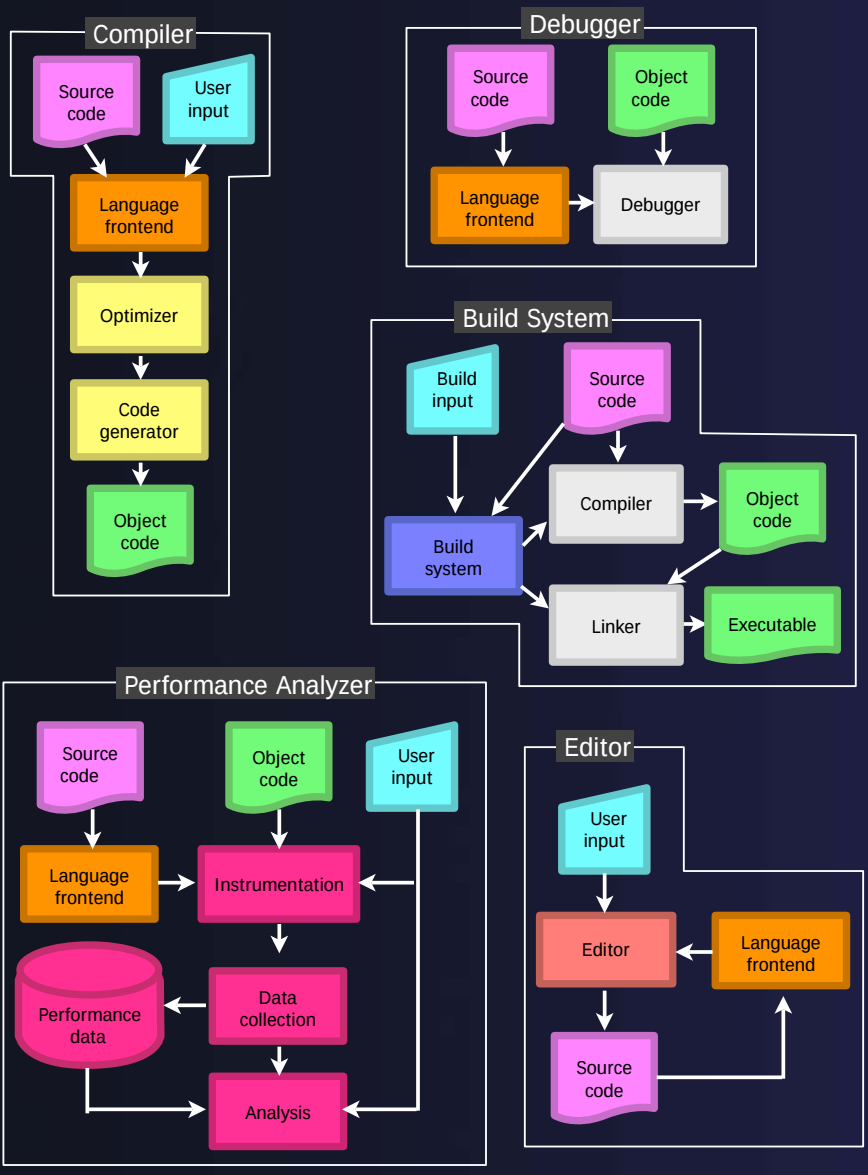
Agenda

- Thursday 5/24
  - 09:00 - 11:30 Session 3 : Debugging (including break)
  - 11:30 - 12:00 Session 4 : Languages and Language Services
  - 12:00 - 13:00 Lunch
  - 13:00 - 14:30 Session 4 : Continued
  - 14:30 - 16:30 Session 5 : PTP Roadmap (including break)
  - 16:30 - 17:00 Workshop Wrap Up

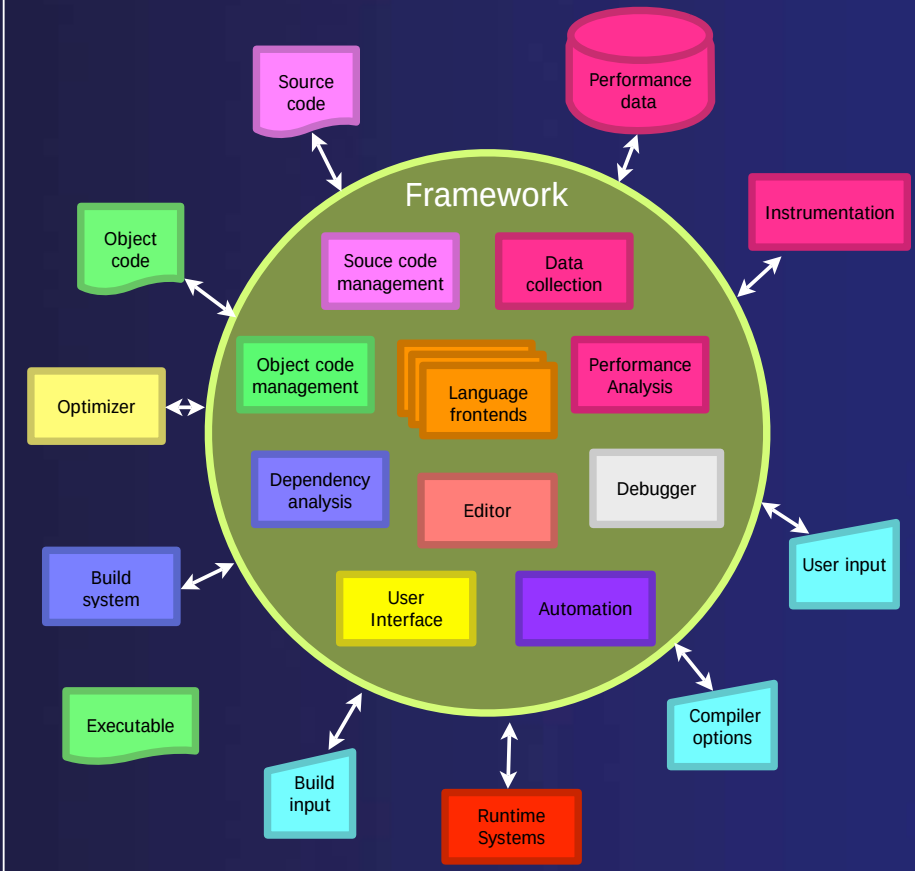
PTP Goals

- Provide a rich set of tools for concurrent/parallel programming
- Provide a uniform, consistent, development environment across platforms and architectures
- Provide a framework to simplify the development and integration of parallel tools
- Provide a robust, scalable, platform that meets the challenges of petascale application development

Existing Tools Paradigm



New Tools Paradigm



Workshop Goals

- Identify improvements to PTP that will ensure a greater level of adoption
- Identify ways of improving and simplifying tool integration in Eclipse
- Identify opportunities for common frameworks and services
- Establish project roles and responsibilities
- Address any concerns about Eclipse as an application development platform

Workshop Outline

- 4 key topics
  - Core Services
  - Performance Tools
  - Debugging
  - Languages and Language Services
- Each session will deal with a separate topic
- Sessions will comprise
  - Short presentations
  - Small group discussions
  - Synthesizing outcomes and prioritizing actions
- Last session will be to develop a PTP roadmap

Workshop Ground Rules

- Commit to the agenda and timetable
- Focus on issues not personalities
- Take personal responsibility for outcomes
- Listen, don’t dominate, and don’t interrupt
- Be brief, daring, honest, and open

Core Services

- What can we do better?
- How can we improve usability and functionality?
- How can we improve tool integration?
- What services are missing?
- What types of architectures should we support?
- Are remote services important?

Performance Tools

- What common infrastructure is needed for performance analysis/optimization?
- What will make integration of performance tools easier?
- How can tools maintain differentiation, but meet the consistency goal?
- Relationship between PTP and TPTP?

Debugging

- How are petascale applications going to be debugged?
- Are developers satisfied with the current debugging tools?
- What other types of tools could aid debugging concurrent/parallel applications?
- How could debugging be better integrated with other tools?

Languages & Language Services

- How can existing languages be used to meet petascale application requirements?
- How can tools help in:
  - Preserving legacy language investment
  - Transitioning to new languages
  - Enabling cross-language applications
- What becomes possible when a complete language front-end is available to other tools?

Myths & Concerns About Eclipse

Myth #1: Emacs Rules

- Emacs is the best editor ever, why would I want to use a big clunky IDE?
- The correct tool should be used for every job
  - Traditional editors may be appropriate for some tasks
- Eclipse provides functionality that is more appropriate for large team-based software development projects
- Sophisticated, language sensitive tools, achieve productivity improvements that are not possible with traditional environments

Myth #2: Java Only

- Eclipse is a Java development environment, so I have to develop my applications in Java
- Eclipse was originally designed for Java development, but now provides a rich, multi-language, ecosystem
- Eclipse supports Java, C, C++, Fortran, COBOL, Python, Perl, PHP, JavaScript, Ruby, Ada, and more...
- Eclipse allows you to choose the most appropriate language for your application

Myth #3: Poor Performance

- Java is too slow, so Eclipse is sluggish
- Early versions of Eclipse were slow
- Significant improvements have been achieved in:
  - Eclipse platform performance
  - JVM performance
  - Reducing memory usage
- Combined with improvements in hardware speed means there is no discernible difference between Eclipse and native applications

Myth #4: Clunky Look & Feel

- UI development in Java requires Swing, which has a clunky look and feel
- Eclipse and Eclipse-based Java applications use the standard widget toolkit (SWT) not Swing
- SWT uses native widgets to achieve:
  - Native look and feel
  - Better performance

Myth #5: Large Project Problems

- I've heard that Eclipse has problems for projects with >500K lines of code and thousands of source files
- Early versions of the C/C++ tools did have some problems with large projects
  - This is no longer the case
- Eclipse is regularly used to build the Linux kernel (5M lines of code) and other large projects such as Apache

Myth #6: Too Complicated

- The Eclipse interface is too complicated and difficult to learn
- The Eclipse interface can be overwhelming for the novice user
- However, the interface is designed for consistency, so once a user becomes familiar with the concepts there are usually no problems
- Eclipse's plugin architecture also allows unneeded functionality to be removed if necessary

Myth #7: Hard To Install

- I downloaded Eclipse, but then I had to download a bunch of other stuff before I could use it
- Eclipse is separated into different components corresponding to the top level projects
- Downloading the Eclipse SDK only provides Java functionality
- Other tools must be installed separately using the update manager
- EasyEclipse now provides pre-packaged Eclipse distributions

Other Concerns?

- What might prevent people from adopting Eclipse?
- I'll never get my group to accept it
- I'll never get my management to accept it
- Software security/from trusted source
- Eclipse doesn't fit into our way of doing things
- Have to write Eclipse-based tools in Java
- Doesn't support multi-language projects
- Scalability of PTP and PLDT
- Why don't you spell Photran Fortran?
- On-line documentation for plug-in development is not very good and out of date or non existent