IBM Research

Eclipse
PTP - Parallel Tools Platform
Performance Tools Framework

Beth Tibbitts
tibbitts@us.ibm.com
High Productivity Tools Group, IBM Research

"This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002"

January 2008

Agenda

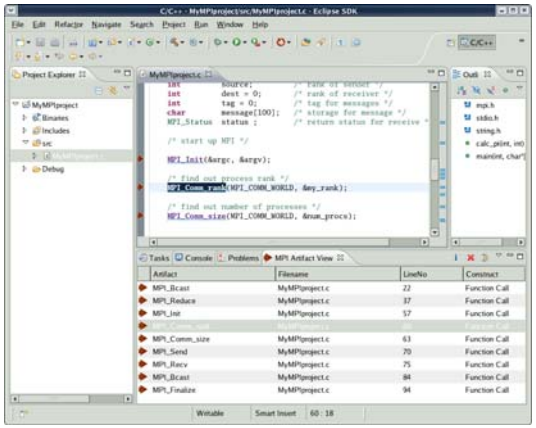
- PTP (Eclipse Parallel Tools Platform) Overview
  - PTP Goals
  - PTP Core: Runtime, debugger
  - PTP's PLDT: Analysis tools
  - PTP Community & Current Availability
- Performance Tools Framework
  - Existing tools
  - What does Eclipse integration mean?
  - Plans for Performance Framework

PTP Goals

http://eclipse.org/ptp

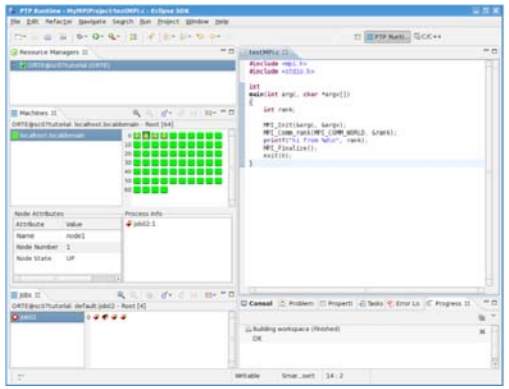
- To produce an open-source industry-strength platform that provides a highly integrated environment specifically designed for parallel application development. The project will provide:
  - a standard, portable parallel IDE that supports a wide range of parallel architectures and runtime systems
  - a scalable parallel debugger
  - *support for the integration of a wide range of parallel tools*
  - an environment that simplifies the end-user interaction with parallel systems

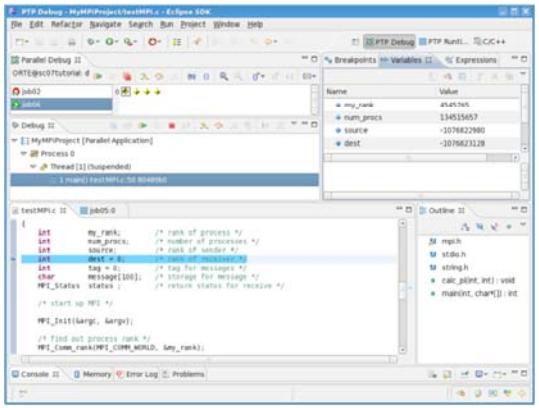
Such as: Performance Tools



Parallel Analysis Tools

Parallel Runtime monitoring





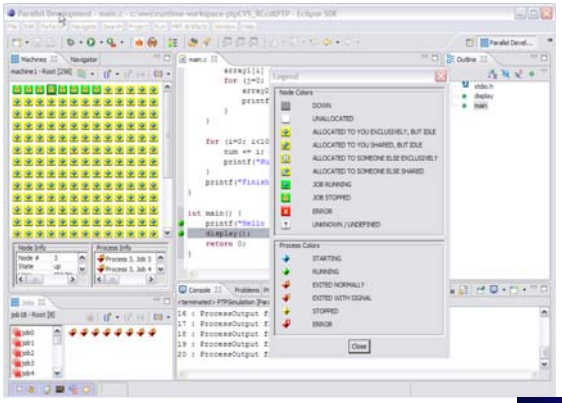
Parallel Debugger

Eclipse PTP: Parallel Tools Platform

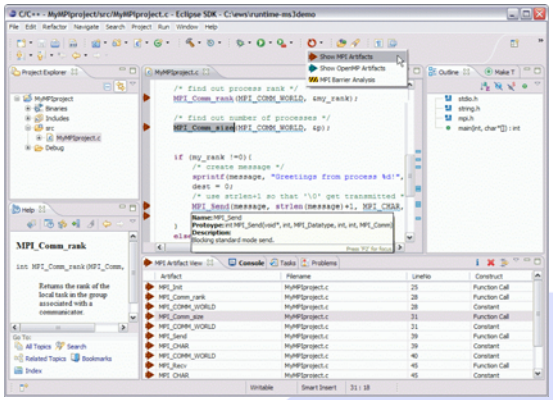
http://eclipse.org/ptp

Parallel Runtime

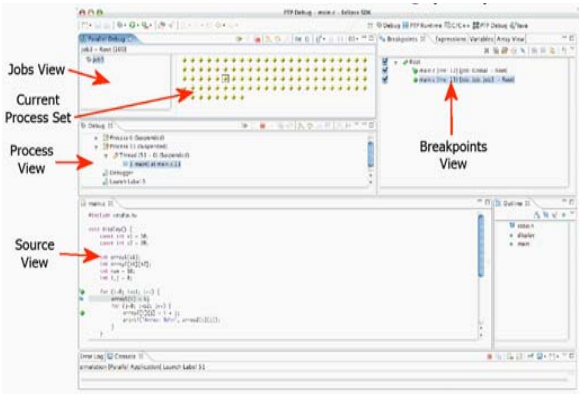
Local or Remote



Based on CDT:
C/C++ Development Tools
http://eclipse.org/cdt

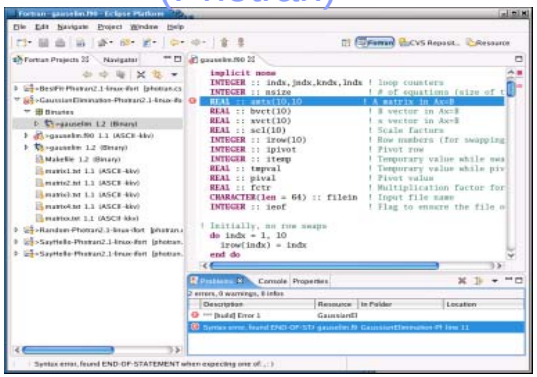


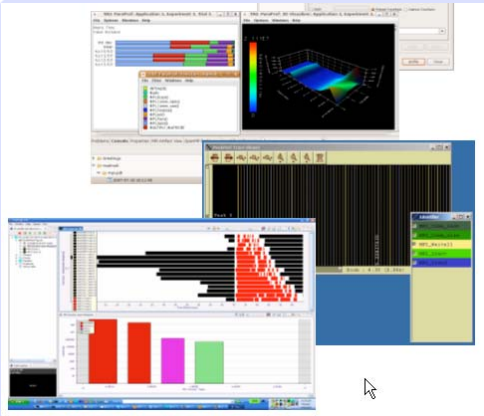
Parallel Language Dev. Tools (PLDT)



Parallel Debugger

Fortran Tools (Photran)





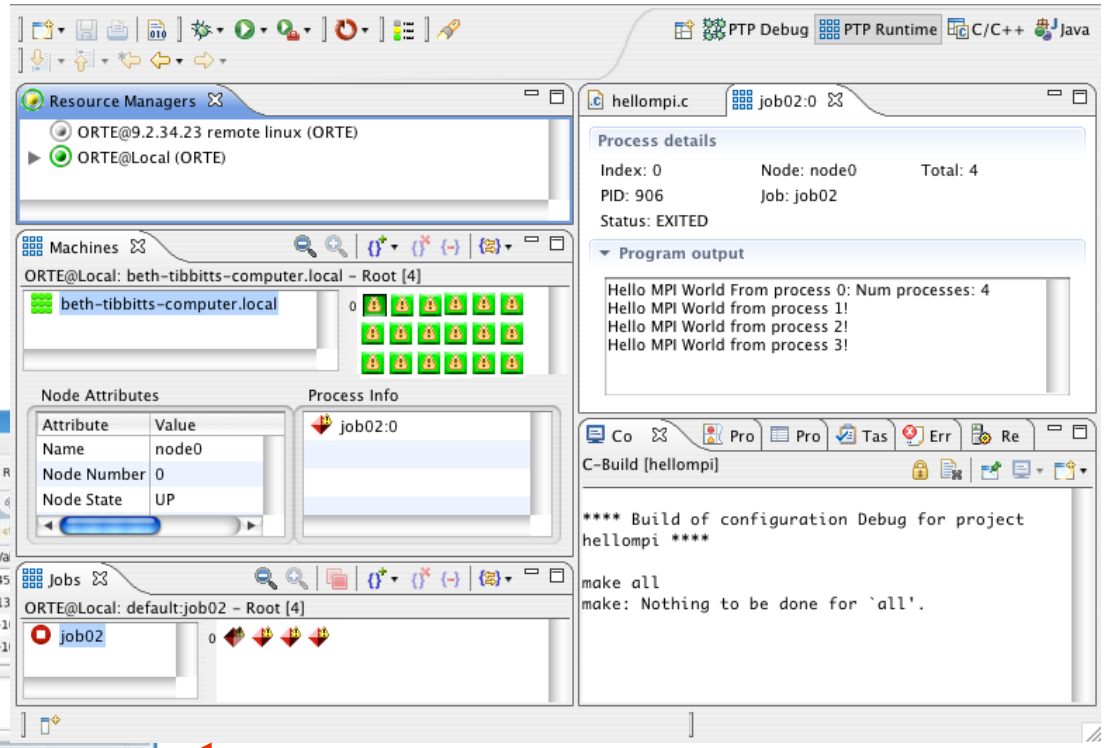
Performance Tools: IBM TuningFork, U.Oregon TAU & Paraver integration;
Future: IBM HPC toolkit, U.Tenn PAPI, etc.

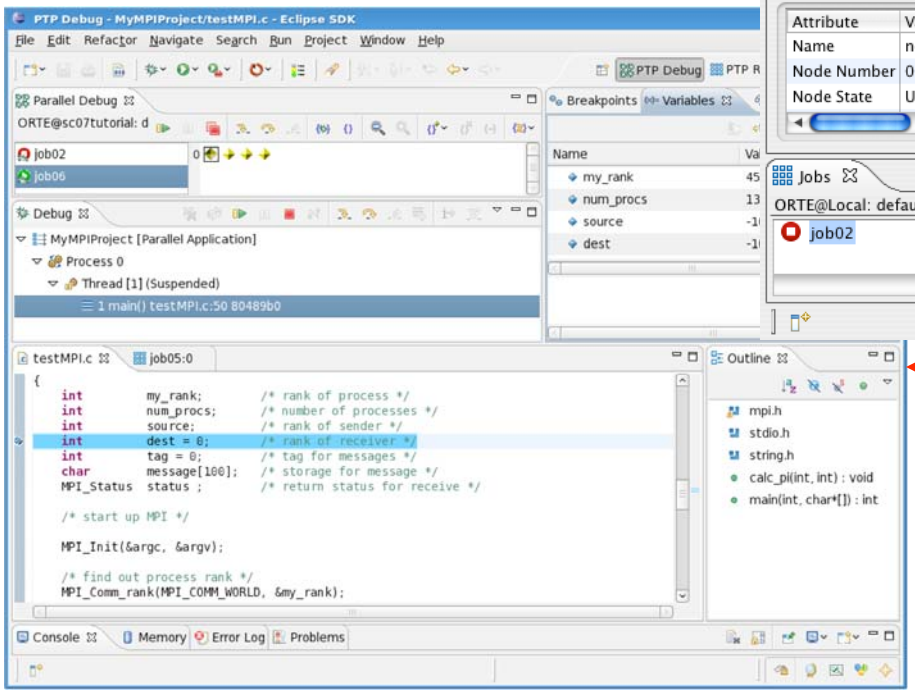
http://eclipse.org/photran

PTP Core Features

Parallel Runtime Monitoring

- Local and Remote
- Support for job schedulers
- Individual or combined console output

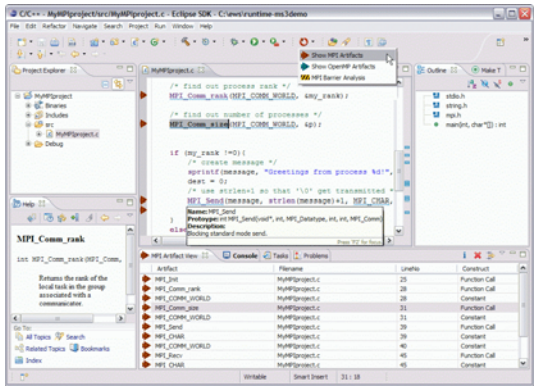




Parallel Debugger

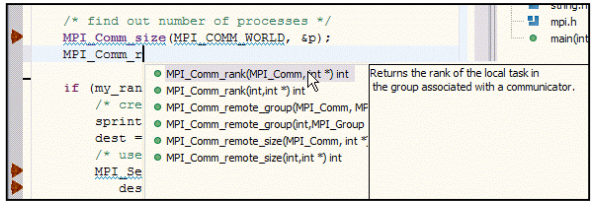
- Monitor/launch/debug on multiple systems
- Control 1 or many processes: step, breakpoint, variable monitoring
- See task or aggregated output

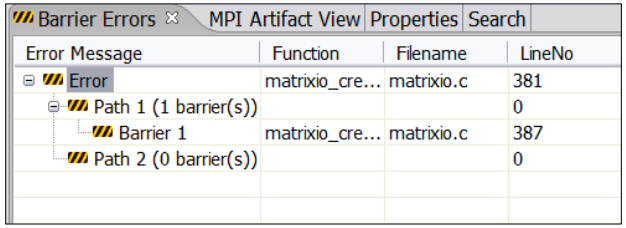
PTP / Parallel Language Development Tools - Analysis tools

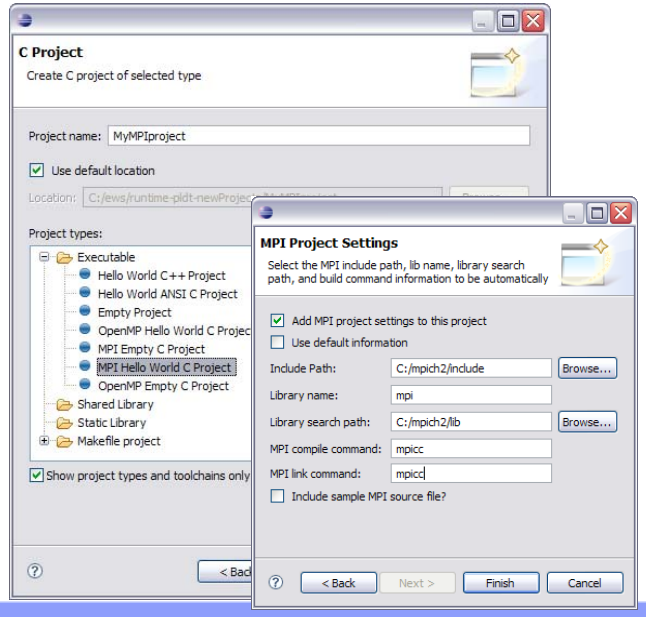


Current PLDT – available at eclipse.org

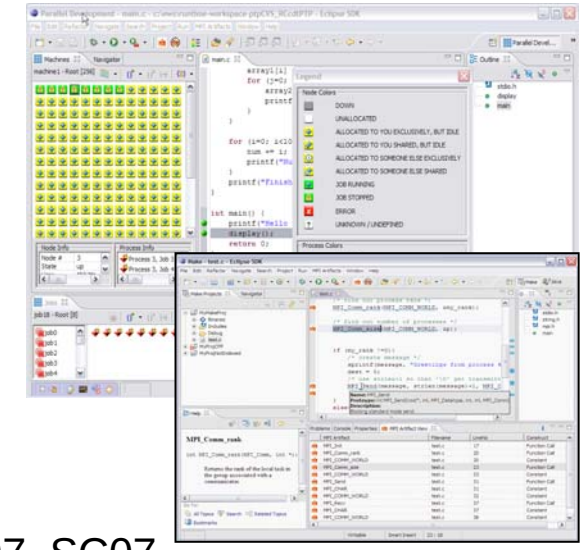
- Tools to assist developers in developing parallel codes
- Assistance tools: Identify MPI/OpenMP/LAPI "artifacts", help (hover, content assist, etc.), wizard for new MPI project creation
- Static Analysis tools:
  OpenMP concurrency analysis
  MPI barrier analysis to detect deadlocks





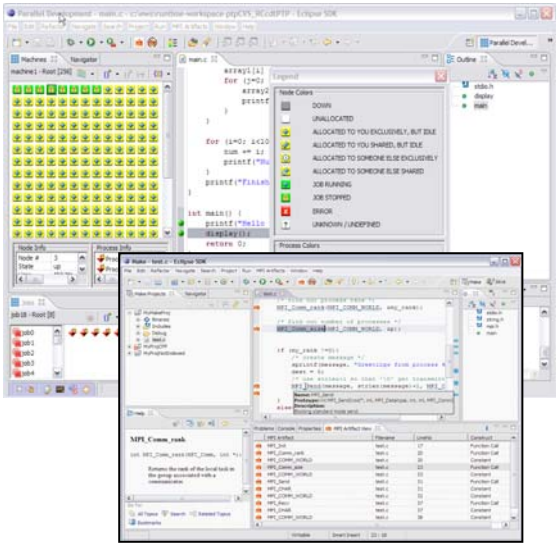


PTP Community & Availability



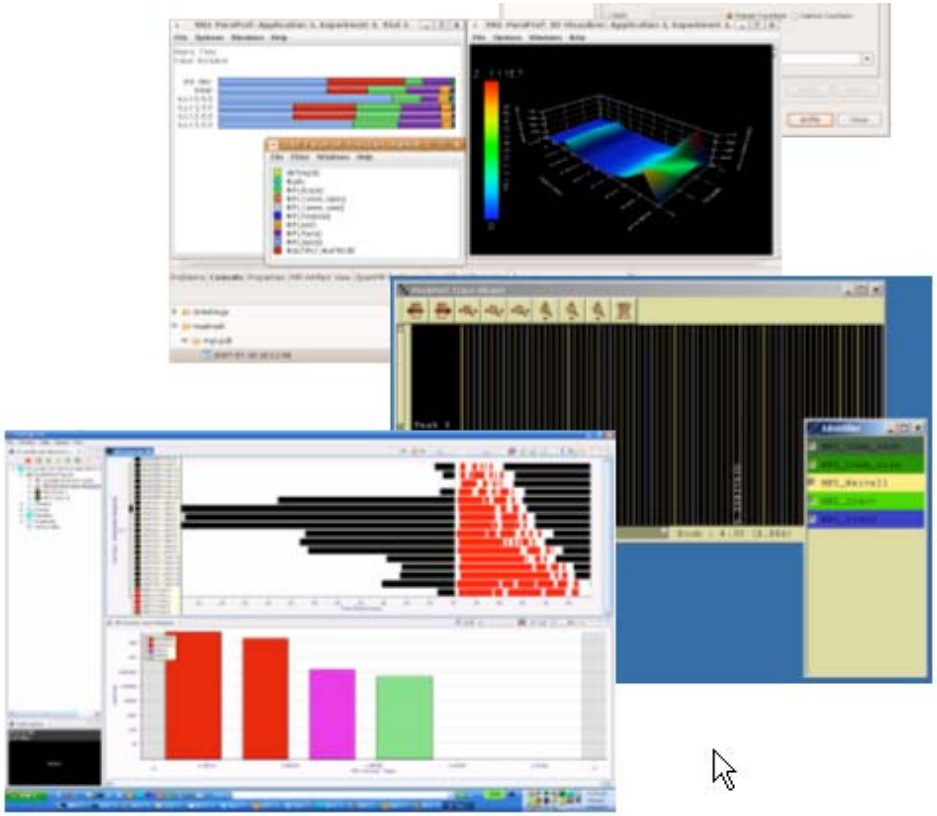
- Building the PTP Community
  - PTP Download statistics: over 2000
  - Website & Wiki: http://eclipse.org/ptp
  - Mailing lists: for users and developers
  - Events
    - PTP tutorials @LACSI 06, SC06, ORNL 5/07, OSCON 07, SC07
    - PTP Workshop @ORNL May ’07
    - SC07: PTP Tutorial, BOF, and demo’d at IBM booth
    - 2008 tutorial plans: HPCSW, OSCON, SC08, NCSA
- Academic Adoption
  - Calvin College, University of Kentucky, LSU
- Availability: PTP 2.0 “early release” available now at http://eclipse.org/ptp
- Related Communities
  - Scalable Tools Communication Infrastructure (STCI) open-source project

PTP contributor Community



- Committers:
  - LANL, IBM, Monash Univ, U. Oregon
- Contributors:
  - Oak Ridge National Lab: PTP Core & parallel debugger
- Interested/started
  - U.Tenn: getting starting with plugins for PAPI
  - U.Houston: contributing compiler analysis feedback (Fortran)
  - BSC: integrating Performance tools
  - MCSC: Munich Computational Sciences Center / consortium
- Interested/discussing
  - HP: Remote & PLDT
  - Georgia Tech: interested in PLDT
  - RENCI/NCSA: interested in integrating performance tools
  - Univ. Florida: UPC and PGAS tools incl. performance tools
- … and growing

PTP Performance Tools



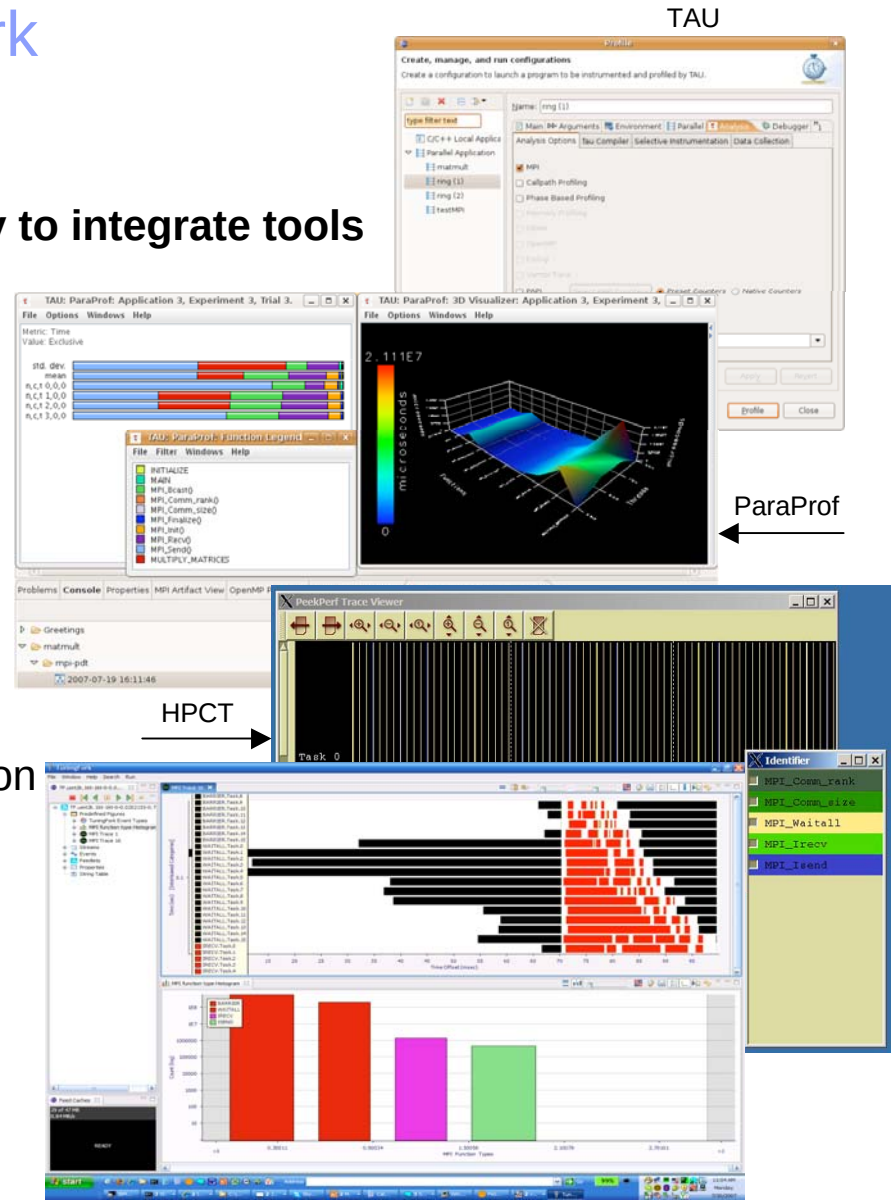
PTP / Performance framework

Goal:

Reduce the "eclipse plumbing" necessary to integrate tools

What we have now:

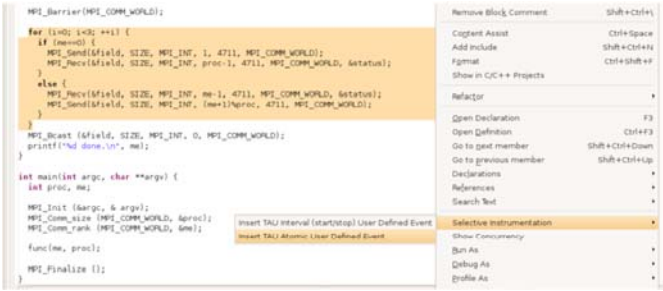
- TAU performance plug-ins
  - Wrappers for non-Eclipse features: TAU, ParaProf
  - General tool-launching feature (XML)
  - UI builder (XML)
- TuningFork visualization
  - Add-ins include source code instrumentation
- Analysis views link items/events to source code lines (from PLDT)
  - Can be generalized for reuse

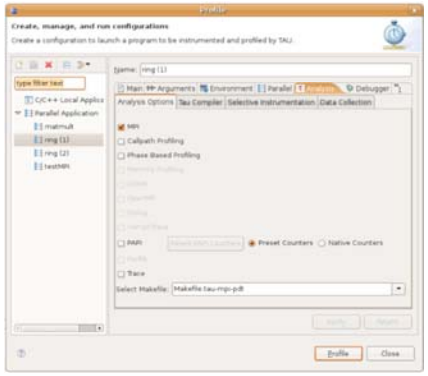


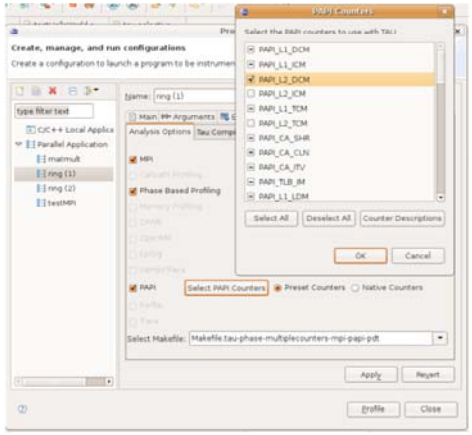
TAU plug-ins contributed to PTP

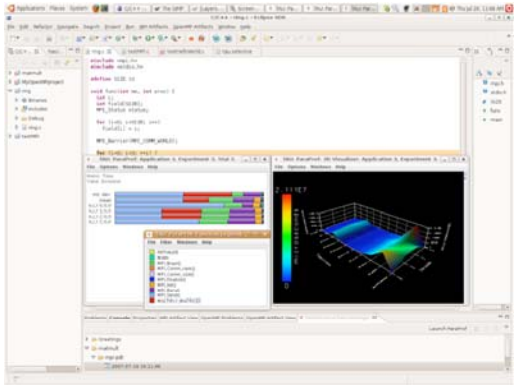
- Contributed by U. Oregon
- TAU (Tuning and Analysis Utilities)
- Eclipse plug-ins wrap TAU functions, make them available from Eclipse
- Other plug-ins launch Paraprof from Eclipse too

http://www.cs.uoregon.edu/research/tau/home.php







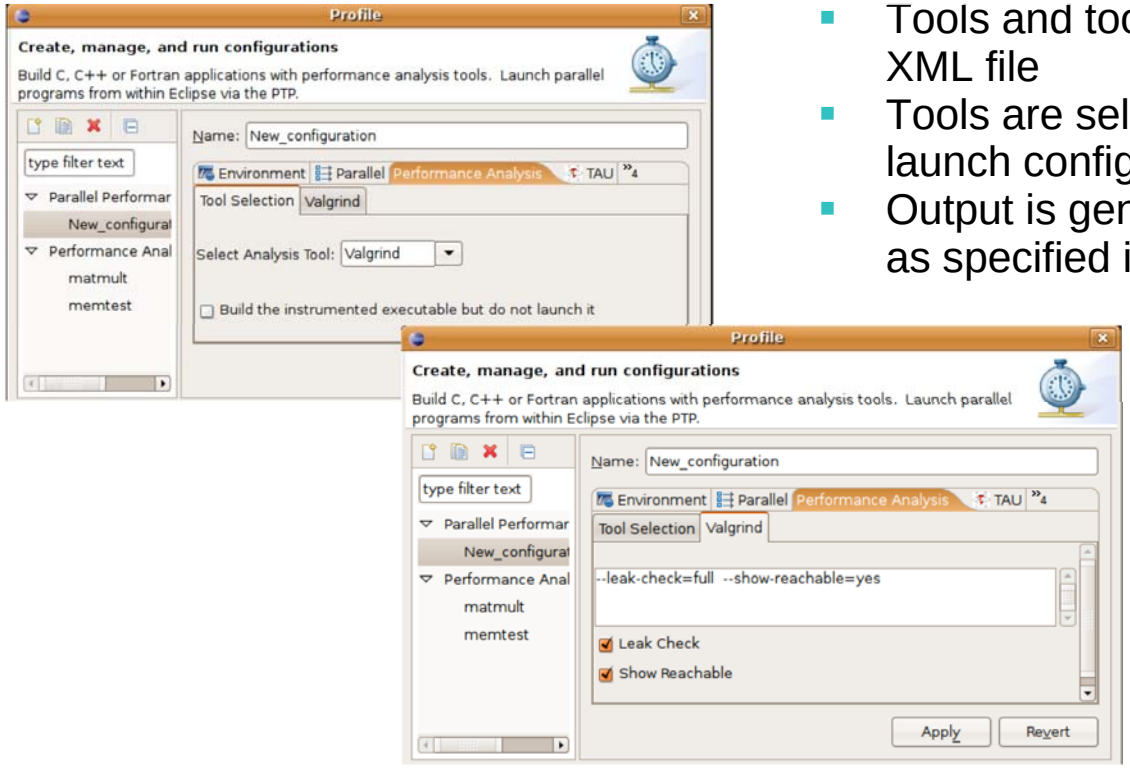


TAU Plug-in features to seed the Performance Analysis Framework

- Began as an Eclipse-based interface for the TAU performance analysis system
- Encapsulates existing command-line based performance analysis tools within the Eclipse environment
- Tool definition file allows generation of tool-specific GUI elements without Eclipse plug-in development
- Tool workflows define compilation, execution, data collection, processing and visualization steps of performance analysis
- Compatible with Photran and CDT projects and with PTP parallel application launching

Dynamic Tool Definitions: Workflows & UI

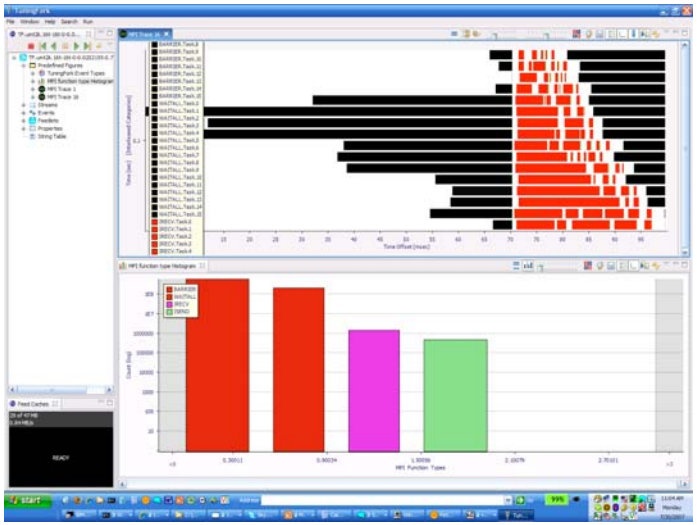
```
-<tool name="Valgrind">
  -<execute>
     <utility command="bash" group="inbin"/>
    -<utility command="valgrind" group="valgrind">
     -<optionpane title="Valgrind" seperatewith=" ">
        <togoption label="Leak Check" optname="--leak-check=full" tooltip="Full memory leak check"/>
        <togoption label="Show Reachable" optname="--show-reachable=yes" tooltip="Show reachable units"/>
      </optionpane>
     </utility>
   </execute>
 </tool>
```



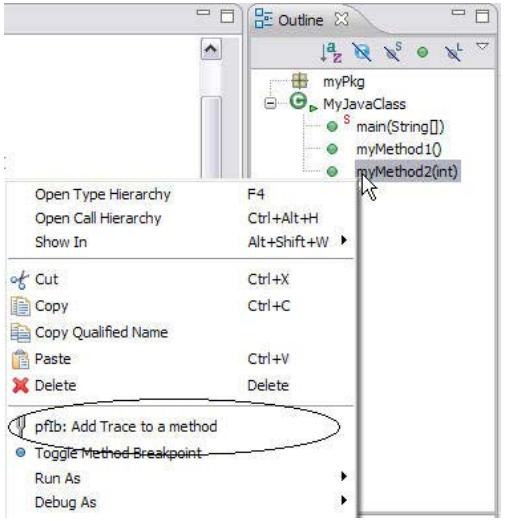
- Tools and tool workflows are specified in an XML file
- Tools are selected and configured in the launch configuration window
- Output is generated, managed and analyzed as specified in the workflow

TuningFork

- Rich graphical Eclipse RCP app developed by IBM Research
- Originally designed for real-time Java traces
- MPI traces and other HPC uses developed
- http://www.alphaworks.ibm.com/tech/tuningfork



- Integration plans:
  - Source Code Instrumentation
    - Automatic and selective instrumentation of Java first, C/C++ later
    - Visible and invisible (modify source code or not)
    - Markers & Views for tracking
  - Source code integration
    - Run TuningFork in SDK (vs. RCP)
    - Link visualization entities to src code lines
  - Early client for Performance Framework



PTP / Performance framework

Goal:
Reduce the "eclipse plumbing" necessary to integrate tools

What we need:

- Help for others interested in integrating tools
  - TAU, ParaProf/ U.Oregon (Wyatt Spear)
  - TuningFork / IBM (Beth Tibbitts)
  - PAPI/ U.Tenn (Dan Terpstra, Shirley Moore)
  - HPC toolkit, Paraver / BSC / IBM
- Reusable parts for building new tools
  - UIs for user interaction
  - Views for source code integration
  - Source code instrumentation tools
  - …etc.

Framework Integration points:

1. Instrumentation, automatic and selective
2. Build, may be transparent to user
3. Launch with instrumentation
4. Management of profile/trace data
5. Analysis/visualization launch

sometimes these overlap

What does (Eclipse) integration mean?

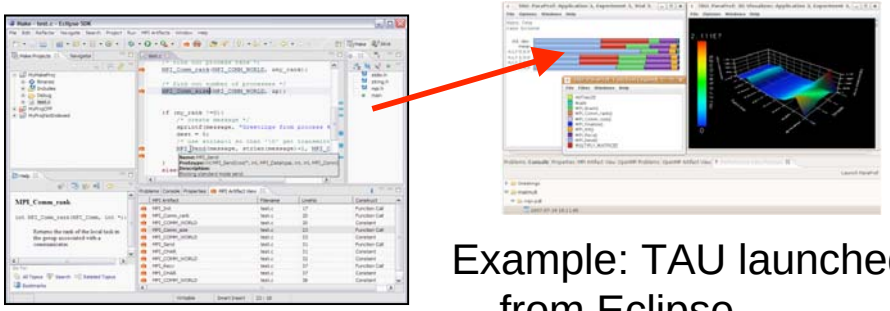
Loose Integration

Tight Integration

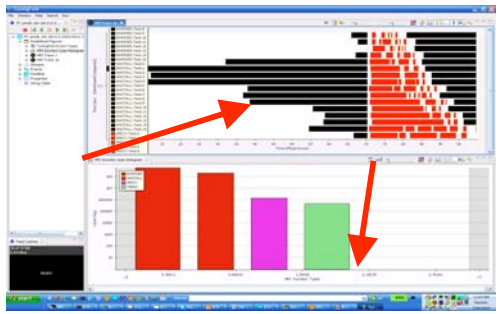
Paraprof **TAU** **TF** **PTP,PLDT**

- Launch external tools from Eclipse UI
- Easiest reuse of existing tools

- Tools run as Eclipse plug-ins
- Easiest interaction / communication with other eclipse tools



Example: TAU launched from Eclipse



Example: TuningFork (and PTP already)

PTP Performance Framework
- Availability Plans for PTP 2.0 (1Q08)

Basic Performance Framework functionality

- Initially will be mostly external launch of existing tools (more “loose” integration)

One sample/reference implementation

- Probably TAU - initially may still have more TAU-specific code

The basic functionality will include:

- Workflow launch: launching external tools at different points in the development workflow (compilation, execution, data collection, processing, and visualization)
- UI launch specifications from XML (build Eclipse GUI)
- Launching a binary through CDT or PTP
- Interaction between Eclipse (e.g. source code) and analysis data collected by external tools

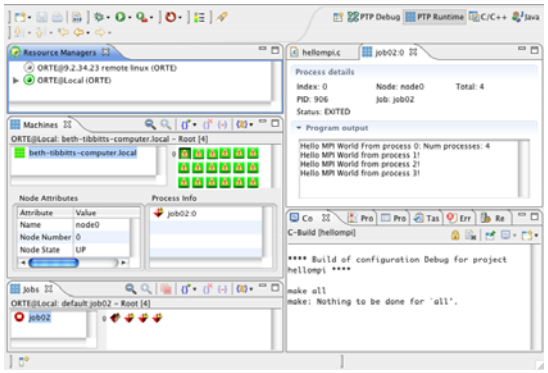
Future Work

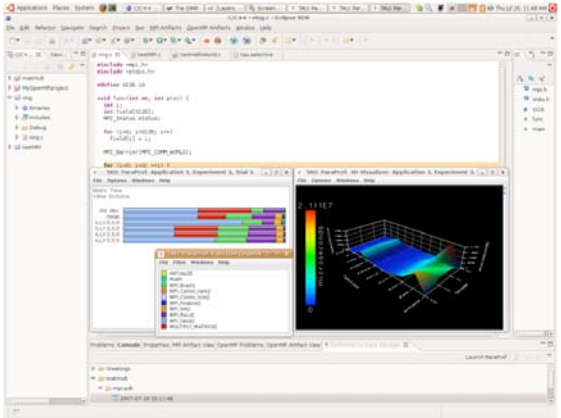
- Availability of existing features generically:
  Eclipse extension points, reusable and callable classes

- Generalization of TAU specific features
  - Selective instrumentation of files, routines and specified source lines
  - PAPI hardware counter specification
  - Profile database and web-portal interoperation

- Logical/iterative performance workflows
- Performance visualization and analysis features integrated directly into Eclipse (e.g. Tuning Fork)
- More flexible instrumentation features (both source-based and not)

PTP - Performance Framework Conclusions





- Goal: make it easier to integrate tools with Eclipse/PTP
- Leverage existing work for specific tools

- Questions?
- Comments?
- Suggestions?
- Volunteers? (This *is* an open-source project!)