Performance Tuning and Analysis Tools

Objective

Become familiar with tools integrated with PTP, to help enhance performance of parallel applications

Contents

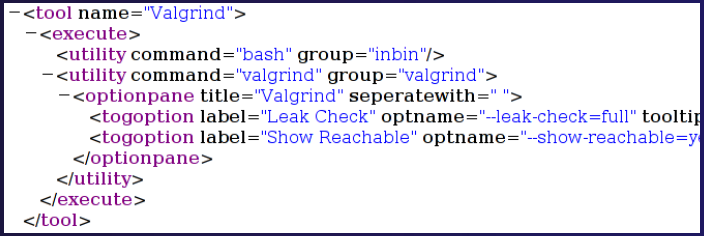
Overview of ETFw and Performance Tools

PTP/External Tools Framework

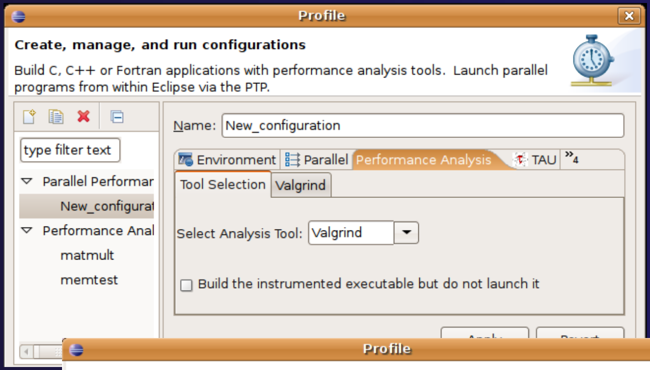
formerly "Performance Tools Framework"

Goal:

- **Reduce the "eclipse plumbing" necessary to integrate tools**
- Provide integration for instrumentation, measurement, and analysis for a variety of performance tools
    - Dynamic Tool Definitions: Workflows & UI
    - 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
    - One-click 'launch' functionality
    - Support for development tools such as TAU, PPW and others.
    - Adding new tools is much easier than developing a full Eclipse plug-in



```
−<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" toolti
          <togoption label="Show Reachable" optname="--show-reachable=y
        </optionpane>
      </utility>
    </execute>
  </tool>
```



Profile

Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

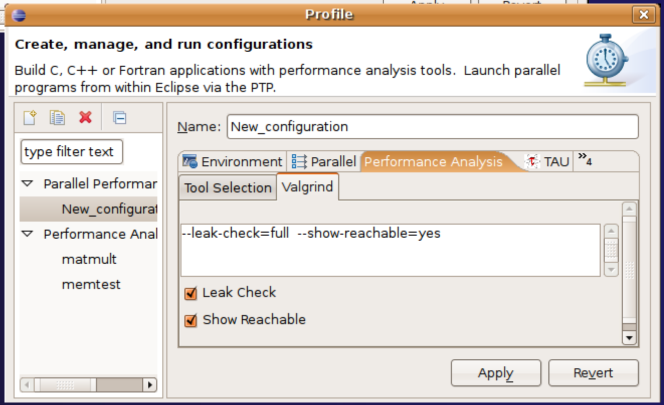
Name: New_configuration

Environment | Parallel | Performance Analysis | TAU

Tool Selection | Valgrind

Select Analysis Tool: Valgrind

☐ Build the instrumented executable but do not launch it



Profile

Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

Name: New_configuration

Environment | Parallel | Performance Analysis | TAU

Tool Selection | Valgrind

--leak-check=full --show-reachable=yes

☑ Leak Check

☑ Show Reachable

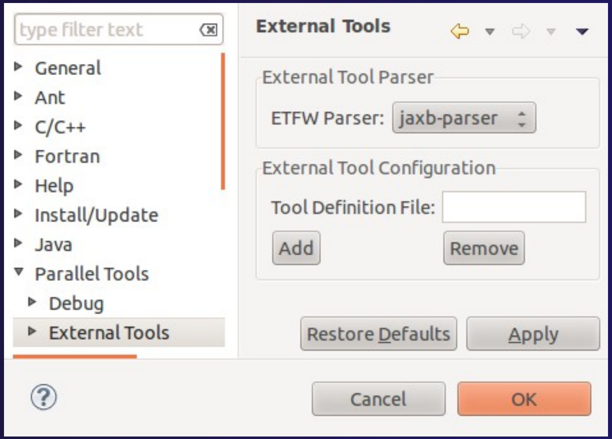
SAX and JAXB Tool Definitions

Prior implementations of ETFW used a simple SAX based schema to define tool workflows

By default workflows now use the more powerful JAXB schema that defines PTP's resource manager

Legacy workflows can still be loaded by selecting the SAX parser in PTP options

Window->Preferences->
Parallel Tools->External Tools



External Tool Parser

ETFW Parser: jaxb-parser

External Tool Configuration

Tool Definition File:

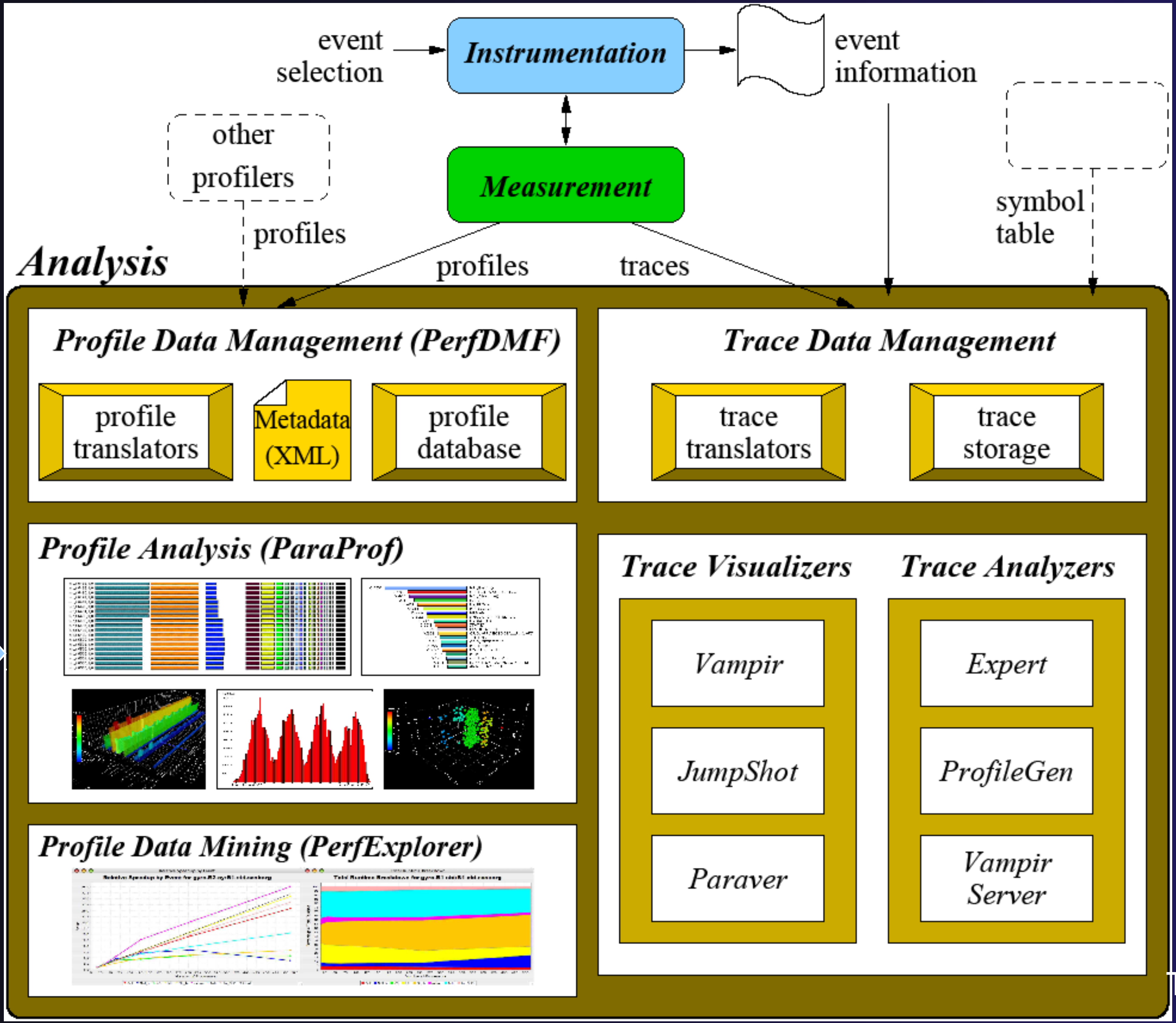
Performance Tuning and Analysis Tools - TAU

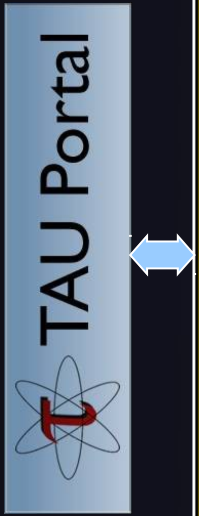
- Objective
  - Become familiar with tools integrated with PTP, to help enhance performance of parallel applications
- Contents
  - Performance Tuning and external tools:
    - PTP External Tools Framework (ETFw), TAU
    - Hands-on exercise using TAU with PTP

TAU: Tuning and Analysis Utilities

- TAU is a performance evaluation tool
- It supports parallel profiling and tracing
    - Profiling shows you how much (total) time was spent in each routine
    - Tracing shows you *when* the events take place in each process along a timeline
- TAU uses a package called PDT (Performance Database Toolkit) for automatic instrumentation of the source code
- Profiling and tracing can measure time as well as hardware performance counters from your CPU (or GPU!)
- TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
    - **paraprof** is TAU's 3D profile browser

TAU Performance System Architecture

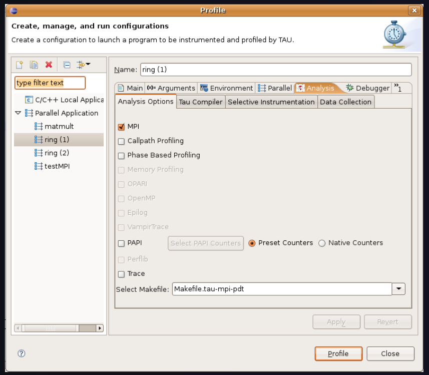




PTP TAU plug-ins

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

- TAU (Tuning and Analysis Utilities)
- First implementation of External Tools Framework (ETFw)
- Eclipse plug-ins wrap TAU functions, make them available from Eclipse
- Full GUI support for the TAU command line interface
- Performance analysis integrated with development environment


Profile

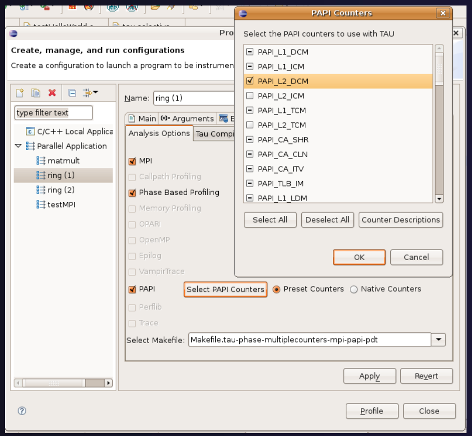
Create, manage, and run configurations

Create a configuration to launch a program to be instrumented and profiled by TAU.

Name: ring (1)

Analysis Options:
- [✓] MPI
- [ ] Callpath Profiling
- [ ] Phase Based Profiling
- [ ] Memory Profiling
- [ ] OPARI
- [ ] OpenMP
- [ ] Epilog
- [ ] VampirTrace
- [ ] PAPI — [✓ Preset Counters] Native Counters
- [ ] Perflib
- [ ] Trace

Select Makefile: Makefile.tau-mpi-pdt


PAPI Counters

Select the PAPI counters to use with TAU

- PAPI_L1_DCM
- PAPI_L1_ICM
- [✓] PAPI_L2_DCM
- [ ] PAPI_L2_ICM
- PAPI_L1_TCM
- [ ] PAPI_L2_TCM
- PAPI_CA_SHR
- PAPI_CA_CLN
- PAPI_CA_ITV
- PAPI_TLB_IM
- PAPI_L1_LDM

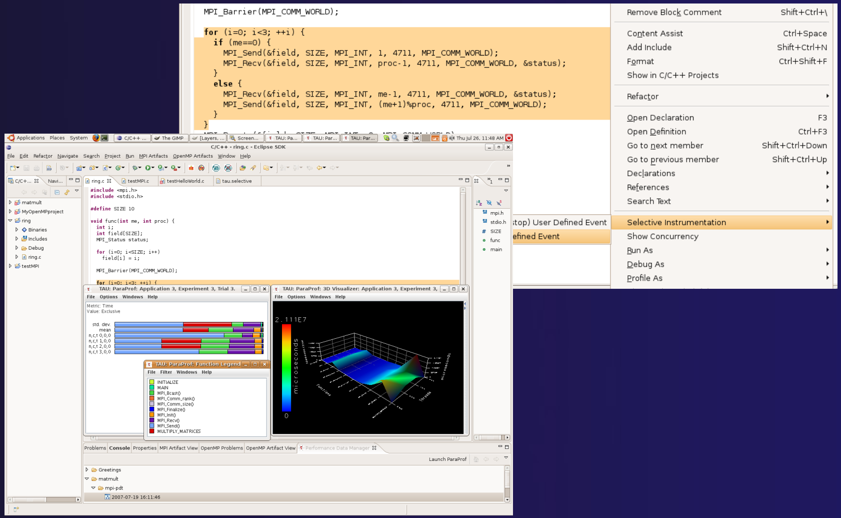
Create, manage, and run configurations

Create a configuration to launch a program to be instrumen

Name: ring (1)

Analysis Options:
- [✓] MPI
- [ ] Callpath Profiling
- [✓] Phase Based Profiling
- [ ] Memory Profiling
- [ ] OPARI
- [ ] OpenMP
- [ ] Epilog
- [ ] VampirTrace
- [✓] PAPI — [✓ Preset Counters] Native Counters
- [ ] Perflib
- [ ] Trace

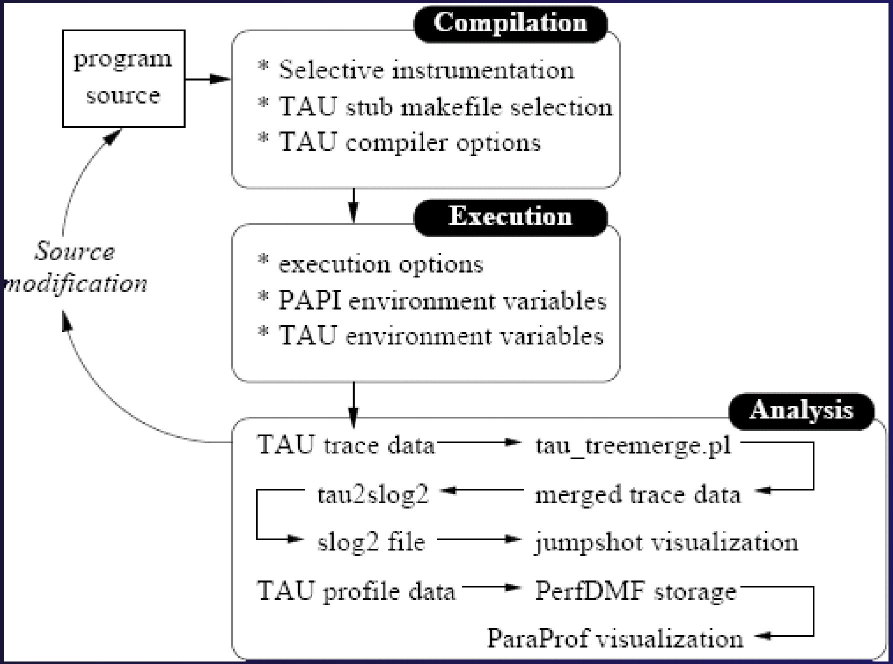
Select Makefile: Makefile.tau-phase-multiplecounters-mpi-papi-pdt


```
MPI_Barrier(MPI_COMM_WORLD);

for (i=0; i<3; ++i) {
  if (me==0) {
    MPI_Send(&field, SIZE, MPI_INT, 1, 4711, MPI_COMM_WORLD);
    MPI_Recv(&field, SIZE, MPI_INT, proc-1, 4711, MPI_COMM_WORLD, &status);
  }
  else {
    MPI_Recv(&field, SIZE, MPI_INT, me-1, 4711, MPI_COMM_WORLD, &status);
    MPI_Send(&field, SIZE, MPI_INT, (me+1)%proc, 4711, MPI_COMM_WORLD);
  }
}
```
Remove Block Comment Shift+Ctrl+\
Content Assist Ctrl+Space
Add Include Shift+Ctrl+N
Format Ctrl+Shift+F
Show in C/C++ Projects
Refactor
Open Declaration F3
Open Definition Ctrl+F3
Go to next member Shift+Ctrl+Down
Go to previous member Shift+Ctrl+Up
Declarations
References
Search Text
Selective Instrumentation
Show Concurrency
Run As
Debug As
Profile As
top) User Defined Event
efined Event
C/C++ - ring.c - Eclipse SDK
testHelloWorld.c
tau.selective
mpi.h
stdio.h
SIZE
func
main
TAU: ParaProf: Application 3, Experiment 3, Trial 3.
TAU: ParaProf: 3D Visualizer: Application 3, Experiment 3,
File Options Windows Help
2.111E7
microseconds
TAU: ParaProf: Function Legend
Thu Jul 26, 11:48 AM

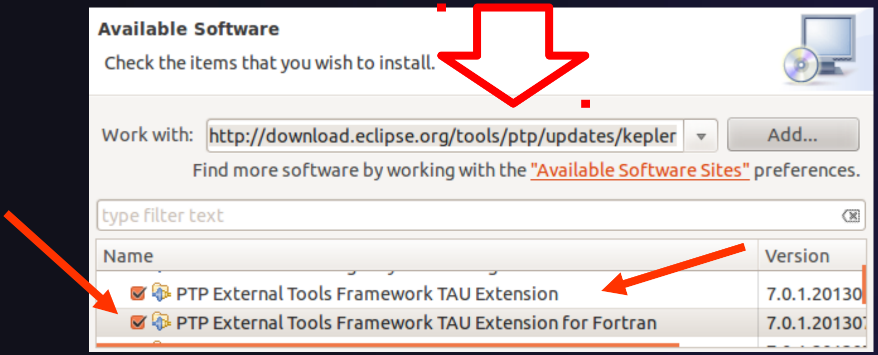
TAU Integration with PTP

- TAU: Tuning and Analysis Utilities
  - Performance data collection and analysis for HPC codes
  - Numerous features
  - Command line interface
- The TAU Workflow:
  - Instrumentation
  - Execution
  - Analysis



TAU PTP Installation

- This tutorial assumes that the TAU extensions for PTP are installed – they are not included in the "Eclipse for Parallel Application Developers"
- The installation section (Module 1) shows how to install TAU and other features from the PTP update site – be sure TAU was selected



Available Software

Check the items that you wish to install.

Work with: http://download.eclipse.org/tools/ptp/updates/kepler

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ PTP External Tools Framework TAU Extension	7.0.1.20130
☑ PTP External Tools Framework TAU Extension for Fortran	7.0.1.20130

To confirm:
- Help>Install New Software…
- Select the link "What is already installed" at the bottom of the dialog
- You should see the TAU Extension

Installing TAU Analysis Tools

- The TAU plugin can use ParaProf for visual analysis and TauDB for organization of profiles
- To install these utilities on Mac or Linux platforms:
  - Download (browser, curl or wget) tau.uoregon.edu/tautools.tgz
  - tar -zxf tautools.tgz
  - cd tautools-2.22.2
  - ./configure
  - Set path as shown (launch eclipse from this environment)
  - Run taudb_configure and follow the instructions
- Java WebStart: tau.uoregon.edu/paraprof
- TAU Installation, downloads and instructions: tau.uoregon.edu

Assumptions

- Obtain and install TAU*
  - Download at tau.uoregon.edu
  - The website includes setup and user guides
- Set up the $PATH on the remote machine*
  - For TAU you should be able to run ‘which pprof’ on a remote login and see a result from your TAU bin directory
  - On trestles.sdsc.edu this is accomplished by loading the tau module in the environment manager for the build and launch configurations
- Include ‘eclipse.inc’ in the makefile*
  - Create an empty eclipse.inc file in the same directory as the makefile
  - Place ‘include eclipse.inc’ in the makefile after regular compiler definitions
  - ETFw will modify eclipse.inc to set CC/CXX/FC variables

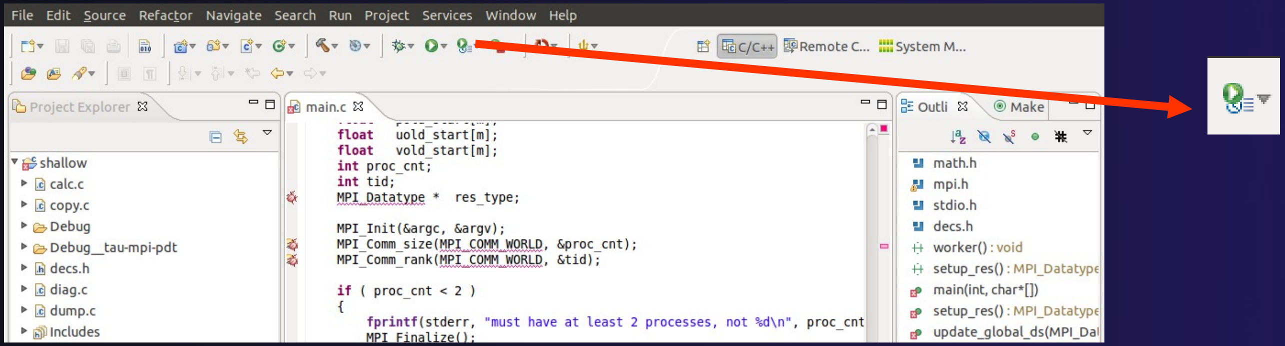
* SC tutorial: this has been done for you

Selective Instrumentation

- By default tau provides timing data for each subroutine of your application
- Selective instrumentation allows you to include/exclude code from analysis and control additional analysis features
  - Include/exclude source files or routines
  - Add timers and phases around routines or arbitrary code
  - Instrument loops
  - Note that some instrumentation features require the PDT
- Right click on calc.c, init.c, diag.c go to the Selective Instrumention option and select Instrument Loops
- Note the creation of tau.selective (refresh if needed)

Begin Profile Configuration

- The ETFw uses the same run configurations and resource managers as debugging/launching
- Click on the ‘Run’ menu or the right side of the Profile button

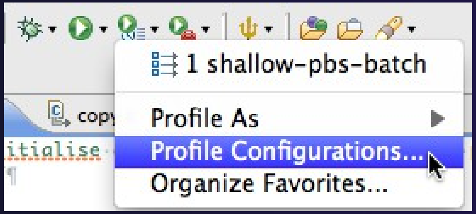


```
float   uold_start[m];
float   vold_start[m];
int proc_cnt;
int tid;
MPI_Datatype *  res_type;

MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &proc_cnt);
MPI_Comm_rank(MPI_COMM_WORLD, &tid);

if ( proc_cnt < 2 )
{
    fprintf(stderr, "must have at least 2 processes, not %d\n", proc_cnt
    MPI_Finalize();
```

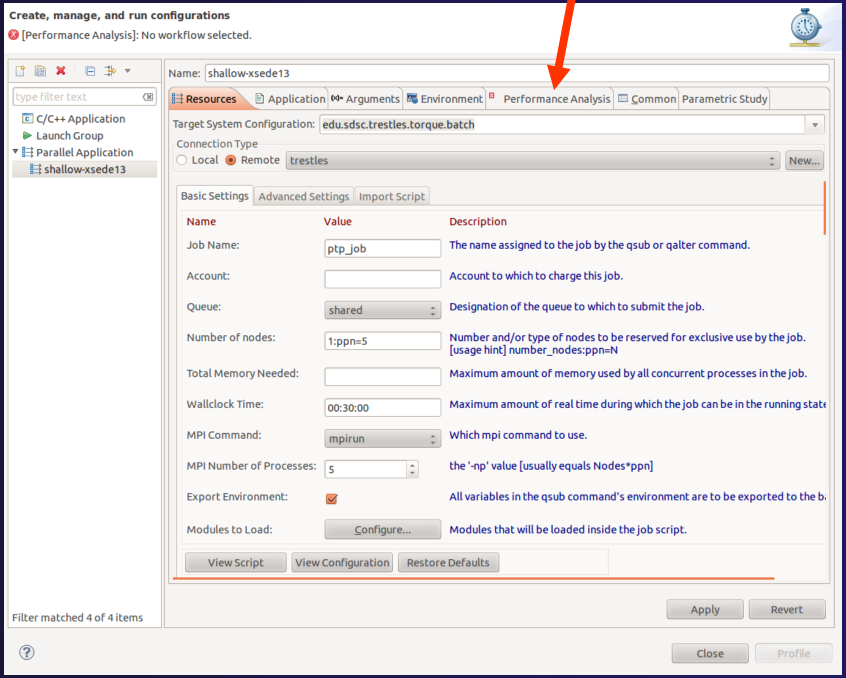
- From the dropdown menu select ‘Profile configurations…’



Select Configuration

- Select the shallow configuration prepared earlier
- The Resource and Application configuration tabs require little or no modification
  - We are using the same resource manager and Torque settings
  - Since we are using a makefile project the application will be rebuilt in and run from the previously selected location

Performance Analysis tab is present in the **Profile Configurations** dialog



Create, manage, and run configurations

[Performance Analysis]: No workflow selected.

Name: shallow-xsede13

Target System Configuration: edu.sdsc.trestles.torque.batch

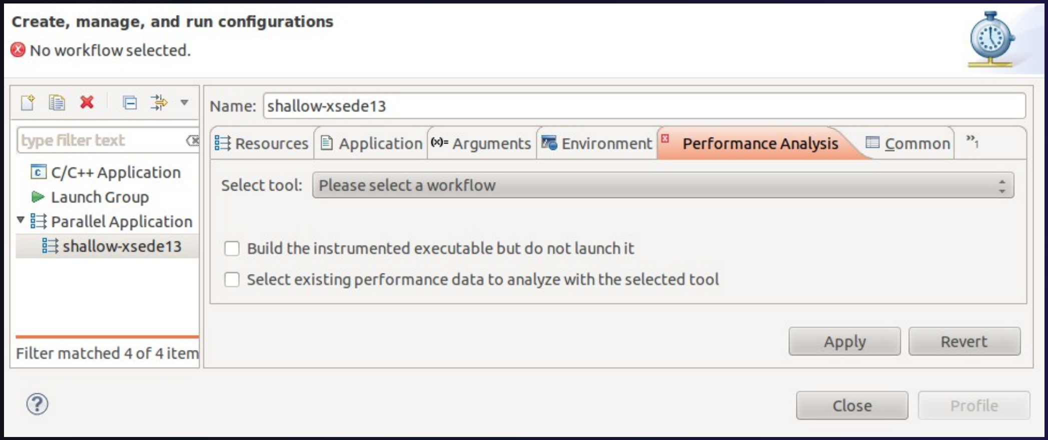
Connection Type: ○ Local ● Remote trestles

Name	Value	Description
Job Name:	ptp_job	The name assigned to the job by the qsub or qalter command.
Account:		Account to which to charge this job.
Queue:	shared	Designation of the queue to which to submit the job.
Number of nodes:	1:ppn=5	Number and/or type of nodes to be reserved for exclusive use by the job. [usage hint] number_nodes:ppn=N
Total Memory Needed:		Maximum amount of memory used by all concurrent processes in the job.
Wallclock Time:	00:30:00	Maximum amount of real time during which the job can be in the running state
MPI Command:	mpirun	Which mpi command to use.
MPI Number of Processes:	5	the '-np' value [usually equals Nodes*ppn]
Export Environment:	✓	All variables in the qsub command's environment are to be exported to the b
Modules to Load:	Configure...	Modules that will be loaded inside the job script.

Filter matched 4 of 4 items

Select Tool/Workflow

- Select the **Performance Analysis** tab and choose the TAU tool set in the 'Select Tool' dropdown box
  - Other tools may be available, either installed as plug-ins or loaded from workflow definition XML files
  - Configuration sub-panes appear depending on the selected tool



Create, manage, and run configurations

No workflow selected.

Name: shallow-xsede13

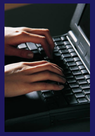
Select tool: Please select a workflow

Build the instrumented executable but do not launch it

Select existing performance data to analyze with the selected tool

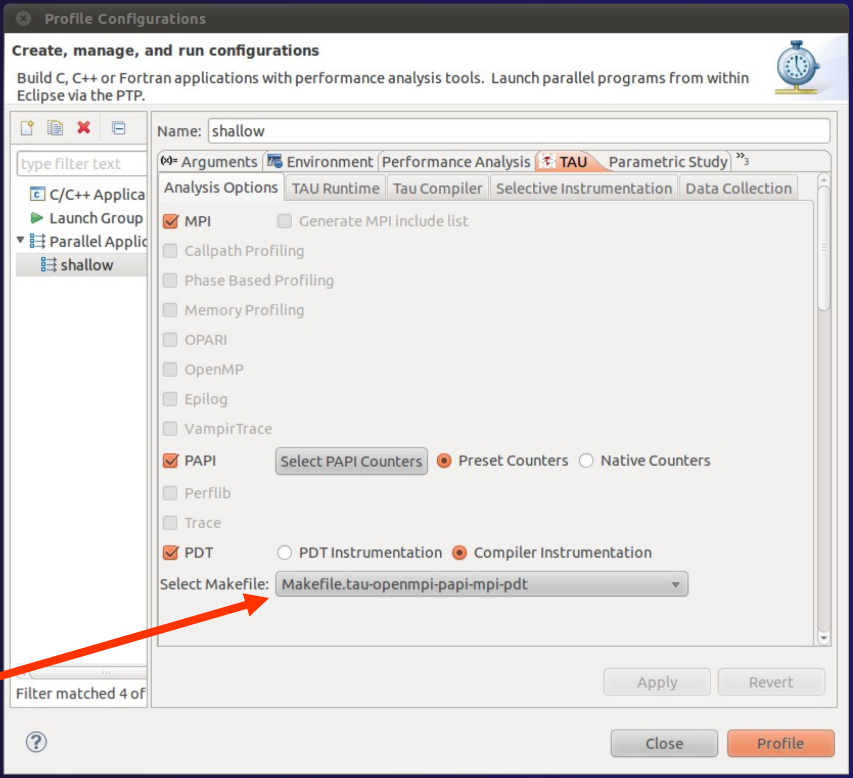
Filter matched 4 of 4 item

Tabs may be hidden if the window is too small



Select TAU Configuration

- Choose the TAU Makefile tab
  - All TAU configurations in remote installation are available
  - Check MPI and PDT checkboxes to filter listed makefiles
  - Make your selection in the **Select Makefile:** dropdown box
  - Select Makefile.tau-mpi-pdt



Profile Configurations

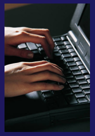
Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

Name: shallow

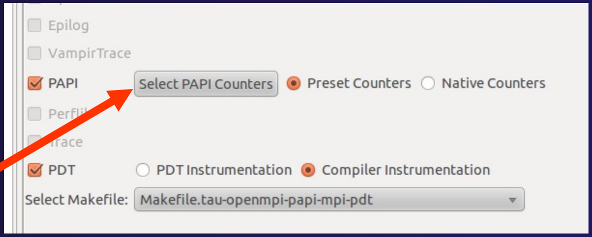
- ☑ MPI — ☐ Generate MPI include list
- ☐ Callpath Profiling
- ☐ Phase Based Profiling
- ☐ Memory Profiling
- ☐ OPARI
- ☐ OpenMP
- ☐ Epilog
- ☐ VampirTrace
- ☑ PAPI — Select PAPI Counters — [✓ Preset Counters] Native Counters
- ☐ Perflib
- ☐ Trace
- ☑ PDT — PDT Instrumentation [✓ Compiler Instrumentation]

Select Makefile: Makefile.tau-openmpi-papi-mpi-pdt

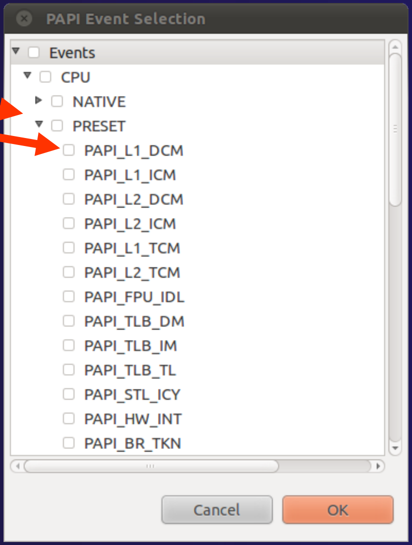


Choose PAPI Hardware Counters

- When a PAPI-enabled TAU configuration is selected the PAPI Counter tool becomes available
  - Select the 'Select PAPI Counters' button to open the tool
  - Open the PRESET subtree
  - Select PAPI_L1_DCM (Data cache misses)
  - Scroll down to select PAPI_FP_INS (Floating point instructions)
  - Invalid selections are automatically excluded
  - Select **OK**
  - **Not available on trestles.sdsc.edu**

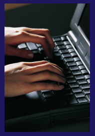


Epilog
VampirTrace
[✓] PAPI   Select PAPI Counters   [✓ Preset Counters]   Native Counters
Perfl
Trace
[✓] PDT   PDT Instrumentation   [✓ Compiler Instrumentation]
Select Makefile: Makefile.tau-openmpi-papi-mpi-pdt



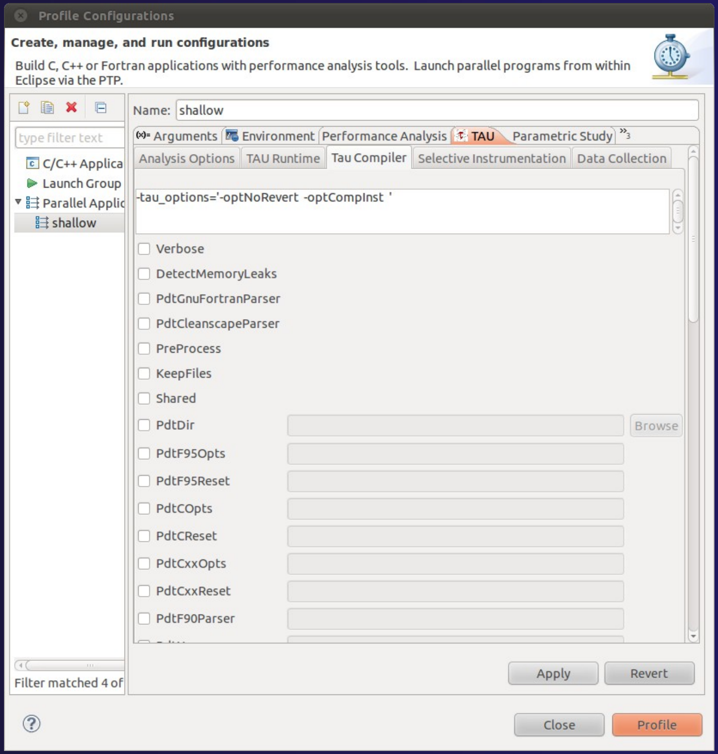
PAPI Event Selection

- Events
  - CPU
    - NATIVE
    - PRESET
      - PAPI_L1_DCM
      - PAPI_L1_ICM
      - PAPI_L2_DCM
      - PAPI_L2_ICM
      - PAPI_L1_TCM
      - PAPI_L2_TCM
      - PAPI_FPU_IDL
      - PAPI_TLB_DM
      - PAPI_TLB_IM
      - PAPI_TLB_TL
      - PAPI_STL_ICY
      - PAPI_HW_INT
      - PAPI_BR_TKN



Compiler Options

- TAU Compiler Options
  - Set arguments to TAU compiler scripts
  - Control instrumentation and compilation behavior
  - Verbose shows activity of compiler wrapper
  - KeepFiles retains instrumented source
  - PreProcess handles C type ifdefs in fortran
- In the Selective Instrumentation tab select Internal then hit Apply
- Scroll to bottom of the Tau Compiler tab and activate TauSelectFile to use tau.selective


Profile Configurations

Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

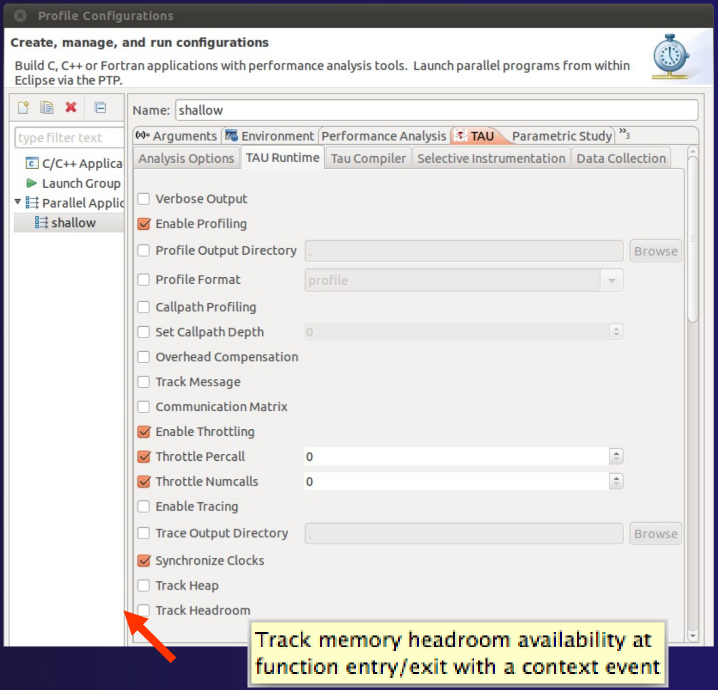
Name: shallow

```
-tau_options='-optNoRevert -optCompInst '
```

- [ ] Verbose
- [ ] DetectMemoryLeaks
- [ ] PdtGnuFortranParser
- [ ] PdtCleanscapeParser
- [ ] PreProcess
- [ ] KeepFiles
- [ ] Shared
- [ ] PdtDir
- [ ] PdtF95Opts
- [ ] PdtF95Reset
- [ ] PdtCOpts
- [ ] PdtCReset
- [ ] PdtCxxOpts
- [ ] PdtCxxReset
- [ ] PdtF90Parser

Runtime Options

- TAU Runtime options
  - Set environment variables used by TAU
  - Control data collection behavior
  - Verbose provides debugging info
  - Callpath shows call stack placement of events
  - Throttling reduces overhead
  - Tracing generates execution timelines
- Set Profile Format to merged


Profile Configurations

Create, manage, and run configurations

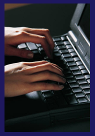
Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

Name: shallow

- [ ] Verbose Output
- [✓] Enable Profiling
- [ ] Profile Output Directory: .
- [ ] Profile Format: profile
- [ ] Callpath Profiling
- [ ] Set Callpath Depth: 0
- [ ] Overhead Compensation
- [ ] Track Message
- [ ] Communication Matrix
- [✓] Enable Throttling
- [✓] Throttle Percall: 0
- [✓] Throttle Numcalls: 0
- [ ] Enable Tracing
- [ ] Trace Output Directory: .
- [✓] Synchronize Clocks
- [ ] Track Heap
- [ ] Track Headroom

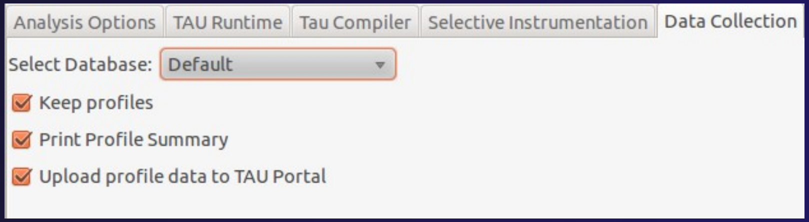
Track memory headroom availability at function entry/exit with a context event

Hover help



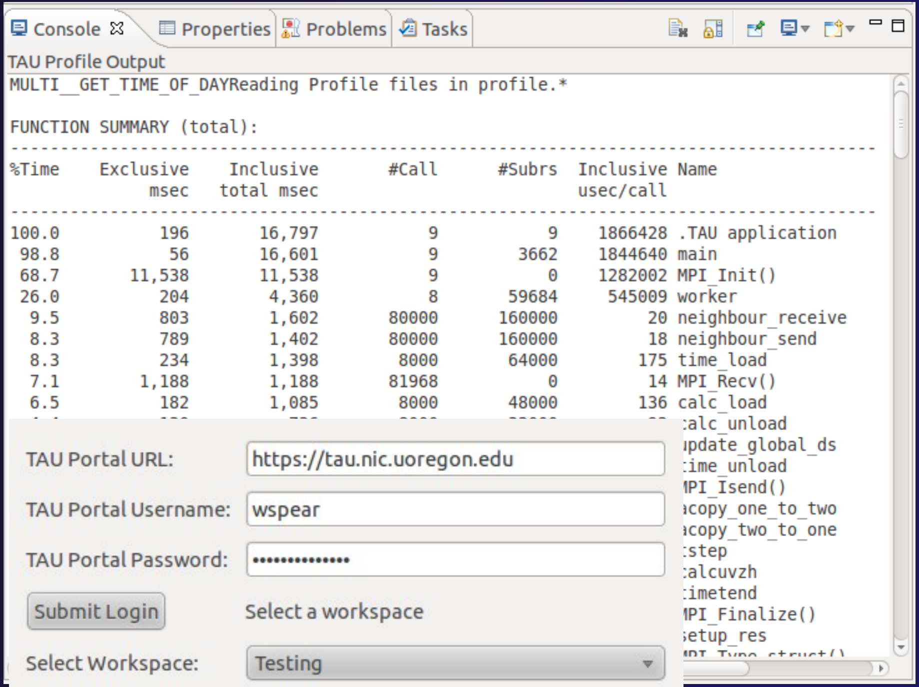
Working with Profiles

- Profiles are uploaded to selected database
- A text summary may be printed to the console
- Profiles may be uploaded to the TAU Portal for viewing online
  - tau.nic.uoregon.edu
- Profiles may be copied to your workspace and loaded in ParaProf from the command line. Select Keep Profiles



Select Database: Default

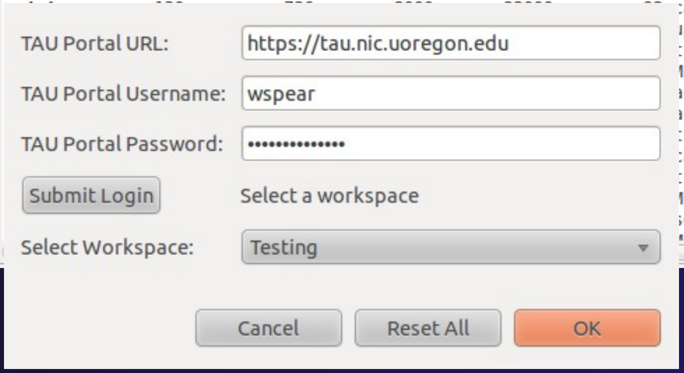
- [✓] Keep profiles
- [✓] Print Profile Summary
- [✓] Upload profile data to TAU Portal



TAU Profile Output

```
MULTI__GET_TIME_OF_DAYReading Profile files in profile.*

FUNCTION SUMMARY (total):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0          196       16,797           9           9    1866428 .TAU application
 98.8           56       16,601           9        3662    1844640 main
 68.7       11,538       11,538           9           0    1282002 MPI_Init()
 26.0          204        4,360           8       59684     545009 worker
  9.5          803        1,602       80000      160000         20 neighbour_receive
  8.3          789        1,402       80000      160000         18 neighbour_send
  8.3          234        1,398        8000       64000        175 time_load
  7.1        1,188        1,188       81968           0         14 MPI_Recv()
  6.5          182        1,085        8000       48000        136 calc_load
                                                                     :alc_unload
                                                                     ıpdate_global_ds
                                                                     :ime_unload
                                                                     IPI_Isend()
                                                                     ıcopy_one_to_two
                                                                     ıcopy_two_to_one
                                                                     :step
                                                                     :alcuvzh
                                                                     :imetend
                                                                     IPI_Finalize()
                                                                     ;etup_res
```



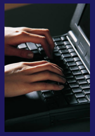
TAU Portal URL: https://tau.nic.uoregon.edu

TAU Portal Username: wspear

TAU Portal Password: ••••••••••••••

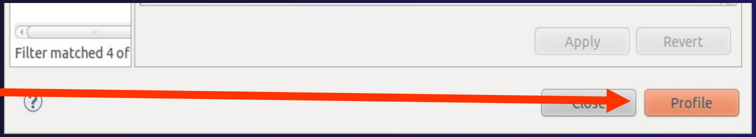
Submit Login — Select a workspace

Select Workspace: Testing

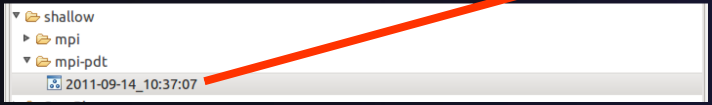


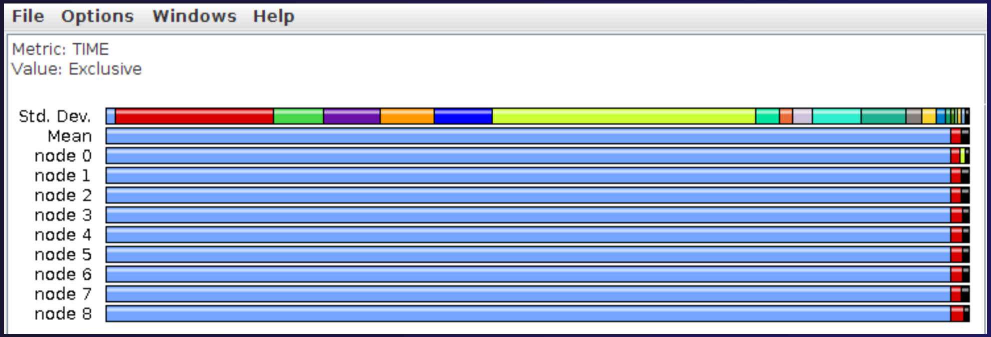
Launch TAU Analysis

- Once your TAU launch is configured select 'Profile'

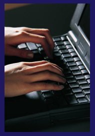


- Notice that the project rebuilds with TAU compiler commands
- The project will execute normally but TAU profiles will be generated
- TAU profiles will be processed as specified in the launch configuration.
- If you have a local profile database the run will show up in the Performance Data Management view
  - Double click the new entry to view in ParaProf
  - Right click on a function bar and select **Show Source Code** for source callback to Eclipse



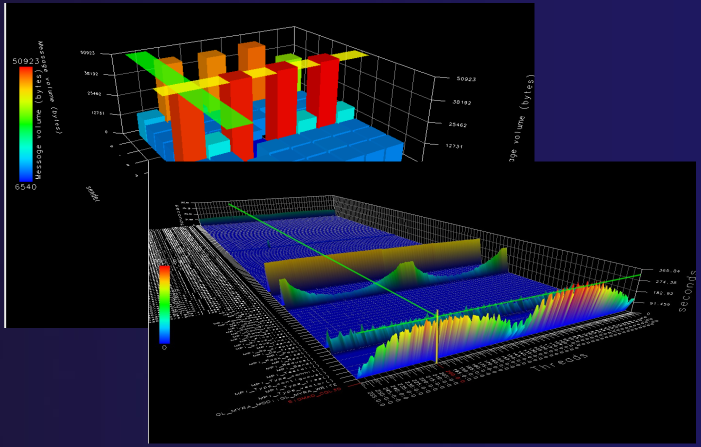


Metric: TIME
Value: Exclusive



Paraprof

- Use ParaProf for profile visualization to identify performance hotspots
  - Inefficient sequential computation
  - Communication overhead
  - IO/Memory bottlenecks
  - Load imbalance
  - Suboptimal cache performance



- Compare multiple trials in PerfExplorer to identify performance regressions and scaling issues
- To use ParaProf, install TAU from tau.uoregon.edu or use Java webstart from tau.uoregon.edu/paraprof

Exercise

- Multi-Trial profile comparison
  1. Edit the shallow Makefile, adding -O3 to CFLAGS and FFLAGS
  2. Rerun the analysis (Run->Profile Configurations. Hit Profile)
  3. A second trial, distinguished by a new timestamp, will be generated
     - It will appear in your Performance Data Manager view if a profile database is available
     - Also present in the Profile subdirectory of your project directory
     - If you do not see a Profile directory right click on your project and go to Synchronization->'Sync All Now'
  4. Load the two trials in paraprof (on the command line: paraprof /path/to/tauprofile.xml)
  5. Open Windows->ParaProf Manager
  6. Expand your database down to reveal all trials
  7. Right click on each trial and click 'Add Mean to Comparison Window' to visualize the two trials side by side