Performance and Memory Evaluation using TAU

Sameer Shende, Allen D. Malony, Alan Morris
University of Oregon
{sameer, malony, amorris}@cs.uoregon.edu

Peter H. Beckman
Argonne National Laboratory
beckman@mcs.anl.gov
Outline of Talk

- Overview of TAU
- Instrumentation
- Measurement
- Analysis
- Conclusions
TAU Performance System

- Tuning and Analysis Utilities (14+ year project effort)
- Performance system framework for HPC systems
  - Integrated, scalable, flexible, and parallel
- Targets a general complex system computation model
  - Entities: nodes / contexts / threads
  - Multi-level: system / software / parallelism
  - Measurement and analysis abstraction
- Integrated toolkit for performance problem solving
  - Instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining

http://www.cs.uoregon.edu/research/tau
Definitions – Profiling

- **Profiling**
  - Recording of summary information during execution
    - inclusive, exclusive time, # calls, hardware statistics, …
  - Reflects performance behavior of program entities
    - functions, loops, basic blocks
    - user-defined “semantic” entities
  - Very good for low-cost performance assessment
  - Helps to expose performance bottlenecks and hotspots
  - Implemented through
    - **sampling**: periodic OS interrupts or hardware counter traps
    - **instrumentation**: direct insertion of measurement code
Definitions – Tracing

- **Tracing**
  - Recording of information about significant points (**events**) during program execution
    - entering/exiting code region (function, loop, block, …)
    - thread/process interactions (e.g., send/receive message)
  - Save information in **event record**
    - timestamp
    - CPU identifier, thread identifier
    - Event type and event-specific information
  - **Event trace** is a time-sequenced stream of event records
  - Can be used to reconstruct dynamic program behavior
  - Typically requires code instrumentation
TAU Parallel Performance System Goals

- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid
- Support for performance mapping
- Support for object-oriented and generic programming
- Integration in complex software, systems, applications
Performance and Memory Evaluation using TAU
TAU Performance System Architecture

Profile Data Management
- profile translators
- XML
- profile database

Trace Data Management
- trace translators
- trace storage

Profile Management
- Profile Data Model
- API
- Event System

ParaProf
- display
- analysis

Trace Visualizers
- Vampir
- JumpShot

Trace Analyzers
- Expert
- ProfileGen

Instrumentation
- event information
- source analyzers
- symbol table

Analysis
- profiles
- traces
Program Database Toolkit (PDT)

Application / Library

C / C++ parser

IL

C / C++ IL analyzer

Program Database Files

Fortran parser F77/90/95

IL

IL analyzer

PDBhtml

Program documentation

SILOON

Application component glue

CHASM

C++ / F90/95 interoperability

TAU_instr

Automatic source instrumentation

DUCTAPE

Program documentation

Application component glue

C++ / F90/95 interoperability

Automatic source instrumentation

CUG 2006  Performance and Memory Evaluation using TAU
TAU Instrumentation Approach

- Support for standard program events
  - Routines
  - Classes and templates
  - Statement-level blocks
- Support for user-defined events
  - Begin/End events ("user-defined timers")
  - Atomic events (e.g., size of memory allocated/freed)
  - Selection of event statistics
- Support definition of "semantic" entities for mapping
- Support for event groups
- Instrumentation optimization (eliminate instrumentation in lightweight routines)
Flexible instrumentation mechanisms at multiple levels

- **Source code**
  - manual (TAU API, TAU Component API)
  - automatic
    - C, C++, F77/90/95 (Program Database Toolkit (PDT))
    - OpenMP (*tau_ompcheck*, directive rewriting (*Opapi*), *POMP* spec [KOJAK – UTK, FZJ Germany])

- **Object code**
  - pre-instrumented libraries (e.g., MPI using *PMPI*)
  - *LD_PRELOAD* tau_load.sh library preloading for MPI

- **Executable code**
  - dynamic instrumentation (pre-execution, rewriting) (*DynInstAPI* [U. Maryland, U. Wisconsin, Madison])
  - virtual machine instrumentation (e.g., Java using *JVMPi*)

- **Proxy Components**
Using TAU – A tutorial

- Configuration
- Instrumentation
  - Manual
  - MPI – Wrapper interposition library
  - PDT- Source rewriting for C,C++, F77/90/95
  - OpenMP – Directive rewriting
  - Component based instrumentation – Proxy components
- Binary Instrumentation
  - DyninstAPI – Runtime Instrumentation/Rewriting binary
  - Java – Runtime instrumentation
  - Python – Runtime instrumentation
- Measurement
- Performance Analysis
TAU Measurement System Configuration

configure [OPTIONS]

- \{-c++=<CC>, -cc=<cc}\} Specify C++ and C compilers
- \{-pthread, -sproc\} Use pthread or SGI sproc threads
- -openmp Use OpenMP threads
- -jdk=<dir> Specify Java instrumentation (JDK)
- -opari=<dir> Specify location of Opari OpenMP tool
- -papi=<dir> Specify location of PAPI
- -pdt=<dir> Specify location of PDT
- -dyninst=<dir> Specify location of DynInst Package
- -mpi[inc/lib]=<dir> Specify MPI library instrumentation
- -shmem[inc/lib]=<dir> Specify PSHMEM library instrumentation
- -python[inc/lib]=<dir> Specify Python instrumentation
- -epilog=<dir> Specify location of EPILOG
- -slog2[=<dir>] Specify location of SLOG2/Jumpshot
- -vtf=<dir> Specify location of VTF3 trace package
- -arch=<architecture> Specify architecture explicitly (xt3,bgl…)
TAU Measurement System Configuration

- configure [OPTIONS]
  - -TRACE Generate binary TAU traces
  - -PROFILE (default) Generate profiles (summary)
  - -PROFILECALLPATH Generate call path profiles
  - -PROFILEPHASE Generate phase based profiles
  - -PROFILEMEMORY Track heap memory for each routine
  - -PROFILEHEADROOM Track memory headroom to grow
  - -MULTIPLECOUNTERS Use hardware counters + time
  - -COMPENSATE Compensate timer overhead
  - -CPUTIME Use usertime+system time
  - -PAPIWALLCLOCK Use PAPI’s wallclock time
  - -PAPIVIRTUAL Use PAPI’s process virtual time
  - -SGITIMERS Use fast IRIX timers
  - -LINUXTIMERS Use fast x86 Linux timers
TAU Measurement Configuration – Examples

- ./configure --arch=xt3 --mpi --pdt=/usr/local/pdtoolkit-3.7 --pdt_c++=g++
  - Use TAU with MPI, PDT and use front-end compiler for compiling PDT based tau_instrumentor (that executes on the front-end)
  - Enable TAU profiling (default)

- ./configure
  - Build tools for the front-end (in x86_64/bin directory)

- ./configure --arch=xt3
  -papi=/usr/local/packages/papi-3.2.1
  -pdt=/usr/local/pdtoolkit-3.7 --arch=xt3
  -mpi -MULTIPLECOUNTERS
  - Use PAPI, PDT, MPI packages and multiple counters for measurements

- Typically configure multiple measurement libraries

- Each configuration creates a unique <arch>/lib/Makefile.tau-<options>
  stub makefile that corresponds to the configuration options specified. e.g.,
  - /usr/local/tau/tau-2.15.3/xt3/lib/Makefile.tau-mpi-pdt-pgi
  - /usr/local/tau/tau-2.15.3/xt3/lib/Makefile.tau-mpi-pdt-trace-pgi
TAU_SETUP: A GUI for Installing TAU

TAU_SETUP: A GUI for Installing TAU

CUG 2006 Performance and Memory Evaluation using TAU

16
Using TAU

- Install TAU
  \%
  configure ; make clean install

- Typically modify application makefile
  - Change the name of compiler to tau_cxx.sh, tau_f90.sh

- Set environment variables
  - Name of the stub makefile: TAU_MAKEFILE
  - Options passed to tau_compiler.sh: TAU_OPTIONS

- Execute application
  \%
  mpirun –np <procs> a.out;

- Analyze performance data
  - paraprof, vampir, vng, paraver, jumpshot …
Using Program Database Toolkit (PDT)

1. Parse the Program to create foo.pdb:
   % cxxparse foo.cpp -I/usr/local/mydir -DMYFLAGS ...
   or
   % cparsfoo.c -I/usr/local/mydir -DMYFLAGS ...
   or
   % f95parse foo.f90 -I/usr/local/mydir ...
   % f95parse *.f -omerged.pdb -I/usr/local/mydir -R free

2. Instrument the program:
   % tau_instrumentor foo.pdb foo.f90 -o foo.instr.f90
     -f select.tau

3. Compile the instrumented program:
   % ifort foo.instr.f90 -c -I/usr/local/mpi/include -o foo.o
Using TAU

Step 1: Configure and install TAU:

% configure -pdt=<dir> -mpiinc=<dir> -mpilib=<dir> 
  -arch=xt3 -pdt_c++=g++
% make clean; make install

Builds <taudir>/<arch>/lib/Makefile.tau-<options>
% set path=($path <taudir>/<arch>/bin)

Step 2: Choose target stub Makefile

% setenv TAU_MAKEFILE
  /san/cca/tau/tau-2.15.3/xt3/lib/Makefile.tau-mpi-pdt-pgi
% setenv TAU_OPTIONS ‘-optVerbose -optKeepFiles’
  (see tau_compiler.sh for all options)

Step 3: Use tau_f90.sh, tau_cxx.sh and tau_cc.sh as the F90, C++ or C compilers respectively.

% tau_f90.sh -c app.f90
% tau_f90.sh app.o -o app -lm -lblas

Or use these in the application Makefile.
# set TAU_MAKEFILE and TAU_OPTIONS env vars

CXX = tau_cxx.sh
F90 = tau_f90.sh
CFLAGS =
LIBS = -lm
OBJJS = f1.o f2.o f3.o ... fn.o

app: $(OBJJS)
   $(CXX) $(LDFLAGS) $(OBJJS) -o $@ $(LIBS)

.cpp.o:
   $(CC) $(CFLAGS) -c $<
Tau_[cxx,cc,f90].sh – Improves Integration in Makefiles

```bash
# Step 1: Set the compiler names
F90  = tau_f90.sh
CXX  = tau_cxx.sh
CC   = tau_cc.sh

# instead of mpif90, mpicxx, and mpicc

# Step 2: Set environment variables before invoking make:
# 2a) set TAU stub makefile name
# setenv TAU_MAKEFILE /usr/local/tau-2.15.3/xt3/lib/Makefile.tau-mpi-pdt-pgi
#
# 2b) and optional parameters to pass to tau_compiler.sh:
# setenv TAU_OPTIONS -optVerbose -optTauSelectFile=select.tau -optPdtGnuFortranParser
# See tau_compiler.sh for a complete list of options!

# Use the same compilation rules
OBJS = f1.o f2.o c1.o c2.o
app: $(OBJS)
   $(F90) $(OBJS) -o app $(LIBS)
.f90.o:
   $(F90) -c $(INCLUDE) $< -o $@
.cpp.o:
   $(CXX) -c $(INCLUDE) $< -o $@
clean: /bin/rm -f $(OBJS) app
```

"Makefile" 171L, 4335C written 25.6 0%
### TAU_COMPILER Options

- **-optVerbose**  
  Turn on verbose debugging messages

- **-optPreProcess**  
  Invoke cpp for pre-processing Fortran sources prior to instrumentation

- **-optPdtF95Opts=""**  
  Options for Fortran parser in PDT (f95parse)

- **-optPdtCOpts=""**  
  Options for C parser in PDT (cparse). Typically
  
  $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)

- **-optPdtF90Parser=""**  
  Specify a different Fortran parser. For e.g., f90parse instead of f95parse

- **-optPdtGnuFortranParser**  
  Specify the use of the GNU gfortran parser (gfparse instead of f95parse)

- **-optPdtCxxOpts=""**  
  Options for C++ parser in PDT (cxxparse). Typically
  
  $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)

- **-optPDBFile=""**  

- **-optTauInstr=""**  
  Specify location of tau_instrumentor. Typically
  
  $(TAUROOT)/$(CONFIG_ARCH)/bin/tau_instrumentor

- **-optTauSelectFile=""**  
  Specify selective instrumentation file for tau_instrumentor

- **-optTau=""**  
  Specify options for tau_instrumentor

- **-optCompile=""**  
  Options passed to the compiler. Typically
  
  $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)

- **-optLinking=""**  
  Options passed to the linker. Typically
  
  $(TAU_MPI_FLIBS) $(TAU_LIBS) $(TAU_CXXLIBS)

- **-optNoMpi**  
  Removes -l*mpi* libraries during linking (default)

- **-optKeepFiles**  
  Does not remove intermediate .pdb and .inst.* files

**Example:**

% setenv TAU_OPTIONS ‘-optTauSelectFile=select.tau
 -optVerbose -optPdtGnuFortranParser’
% tau_f90.sh matrix.f90 -o matrix -lm
Using Stub Makefile and TAU_COMPILER

```
include /usr/common/acts/TAU/tau-2.15.3/xt3/lib/
        Makefile.tau-mpi-pdt-pgi

MYOPTIONS=  -optVerbose -optKeepFiles -optTauSelectFile=s.tau

F90 = $(TAU_COMPILER) $(MYOPTIONS) mpif90
OBJS = f1.o f2.o f3.o ...
LIBS = -Lappdir -lapplib1 -lapplib2 ...

app:  $(OBJS)
       $(F90) $(OBJS) -o app $(LIBS)

.f90.o:
       $(F90) -c $<
```
Instrumentation Specification

% tau_instrumentor
For selective instrumentation, use -f option
% tau_instrumentor foo.pdb foo.cpp -o foo.inst.cpp -f selective.dat
% cat selective.dat
# Selective instrumentation: Specify an exclude/include list of routines/files.

BEGIN_EXCLUDE_LIST
void quicksort(int *, int, int)
void sort_5elements(int *)
void interchange(int *, int *)
END_EXCLUDE_LIST

BEGIN_FILE_INCLUDE_LIST
Main.cpp
Foo?.c
*.C
END_FILE_INCLUDE_LIST
# Instruments routines in Main.cpp, Foo?.c and *.C files only
# Use BEGIN_[FILE]_INCLUDE_LIST with END_[FILE]_INCLUDE_LIST
Automatic Outer Loop Level Instrumentation

BEGIN_INSTRUMENT_SECTION
loops file="loop_test.cpp" routine="multiply"
# comment. TAU accepts # as a wildcard in routine name
# and * and ? wildcards in file name.
# You can also specify the full
# name of the routine as is found in profile files.
# loops file="loop_test.cpp" routine="double multiply#"
# loops file="*.f90" routine = "#"
END_INSTRUMENT_SECTION

% pprof
NODE 0;CONTEXT 0;THREAD 0:
-----------------------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs Inclusive Name
msec total msec
-----------------------------------------------------------------------------------------------------
100.0        0.12       25,162           1           1   25162827 int main(int, char **)
100.0        0.175       25,162           1           4   25162707 double multiply()
90.5       22,778       22,778 1           0   22778959 Loop: double multiply()
file = <loop_test.cpp> line,col = <23,3> to <30,3> ]
9.3        2,345        2,345 1           0    2345823 Loop: double multiply()
file = <loop_test.cpp> line,col = <38,3> to <46,7> ]
0.1           33           33 1           0      33964 Loop: double multiply()
file = <loop_test.cpp> line,col = <16.10> to <21.12> ]
Optimization of Program Instrumentation

- Need to eliminate instrumentation in frequently executing lightweight routines

- Throttling of events at runtime:
  
  ```bash
  setenv TAU_THROTTLE 1
  ```

  Turns off instrumentation in routines that execute over 10000 times (TAU_THROTTLE_NUMCALLS) and take less than 10 microseconds of inclusive time per call (TAU_THROTTLE_PERCALL)

- Selective instrumentation file to filter events

  ```bash
  tau_instrumentor [options] -f <file>
  ```

- Compensation of local instrumentation overhead

  ```bash
  configure -COMPENSATE
  ```
TAU_REDUCE

- Reads profile files and rules
- Creates selective instrumentation file
  - Specifies which routines should be excluded from instrumentation

rules → tau_reduce → Selective instrumentation file

profile → tau_reduce
Memory Profiling in TAU

- Configuration option –PROFILEMEMORY
  - Records global heap memory utilization for each function
  - Takes one sample at beginning of each function and associates the sample with the function name

- Configuration option -PROFILEHEADROOM
  - Records headroom (amount of free memory to grow) for each function
  - Takes one sample at beginning of each function and associates the sample with the function name
  - Useful for debugging memory usage on Cray XT3

- On Cray XT3, TAU uses the heap_info( ) call for memory headroom and heap memory utilization information

- On other platforms, TAU uses mallinfo for heap memory utilization, and

- TAU allocates a series of memory blocks (1, 2, 4, 8...512 MB) and when there is no memory left, it starts again (1, 2, 4...) until it can no longer allocate the smallest block. It records memory available and frees the blocks to compute memory headroom

- No need for modifications to the source code for using these options
ParaProf– Flat Profile (MFIX)

Thread: n,c,t 0,0,0
Value Type: Min Value

MFIX   - Memory Headroom Available (MB) 655
MPI_Init() - Memory Headroom Available (MB) 655
DES_INIT_NAMERLIST - Memory Headroom Available (MB) 589
GET_DATA   - Memory Headroom Available (MB) 589
GET_RUN_ID  - Memory Headroom Available (MB) 589
INIT_NAMERLIST   - Memory Headroom Available (MB) 589
MACHINE_CONS   - Memory Headroom Available (MB) 589
READ_NAMERLIST   - Memory Headroom Available (MB) 589
CHECK_DATA_00   - Memory Headroom Available (MB) 589
GRIDMAP::GRIDMAP_INIT   - Memory Headroom Available (MB) 589
GRIDMAP::PARTITION   - Memory Headroom Available (MB) 589
MPI_Comm_rank()   - Memory Headroom Available (MB) 589
MPI_Comm_size()   - Memory Headroom Available (MB) 589
ALLOCATE ARRAYS - Memory Headroom Available (MB) 589
BLANK_LINE   - Memory Headroom Available (MB) 577
LINE_TOO_BIG  - Memory Headroom Available (MB) 577
MAKE_UPPER_CASE - Memory Headroom Available (MB) 577
MPI_Recv_init() - Memory Headroom Available (MB) 577
MPI_Send_init() - Memory Headroom Available (MB) 577
PARSE_LINE   - Memory Headroom Available (MB) 577
REMOVE_COMMENT - Memory Headroom Available (MB) 577
REPLACE_TAB   - Memory Headroom Available (MB) 577
SEEK_COMMENT   - Memory Headroom Available (MB) 577
OPEN_FILES   - Memory Headroom Available (MB) 528
OPEN_FILE   - Memory Headroom Available (MB) 528
CHECK_DATA_01   - Memory Headroom Available (MB) 526
CHECK_DATA_02   - Memory Headroom Available (MB) 526
CALC_CELL   - Memory Headroom Available (MB) 525
CHECK_DATA_03   - Memory Headroom Available (MB) 525
CHECK_DATA_04   - Memory Headroom Available (MB) 525
CHECK_DATA_05   - Memory Headroom Available (MB) 525
CHECK_DATA_06   - Memory Headroom Available (MB) 525
CHECK_DATA_07   - Memory Headroom Available (MB) 525
CHECK_DATA_08   - Memory Headroom Available (MB) 525
CHECK_DATA_09   - Memory Headroom Available (MB) 525
CHECK_ONE_AXIS - Memory Headroom Available (MB) 525
CHECK_PLANE   - Memory Headroom Available (MB) 525
FLOW_TO_VEL  - Memory Headroom Available (MB) 525
GET_BC_AREA   - Memory Headroom Available (MB) 525
GET FLOW BC  - Memory Headroom Available (MB) 525
**pprof – Memory Events on Cray XT3 (MFIX)**

```plaintext
|xterm

<table>
<thead>
<tr>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
<th>Event Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>655</td>
<td>655</td>
<td>655</td>
<td>1.04</td>
<td>MFTX - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>486</td>
<td>579</td>
<td>577</td>
<td>576.9</td>
<td>0.14</td>
<td>BLANK_LTNF - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>577</td>
<td>577</td>
<td>577</td>
<td>0.14</td>
<td>CHECK_DATA_00 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1405</td>
<td>503</td>
<td>502</td>
<td>503</td>
<td>0.14</td>
<td>ACCUMULATION - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>300</td>
<td>503</td>
<td>502</td>
<td>503</td>
<td>0.14</td>
<td>ADJUST_A_II_G - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>300</td>
<td>503</td>
<td>502</td>
<td>503</td>
<td>0.14</td>
<td>ADJUST_A_II_S - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>2400</td>
<td>503</td>
<td>502</td>
<td>503</td>
<td>0.14</td>
<td>ROUND_X - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>174</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0.14</td>
<td>CALC_CELL - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1.40AF-05</td>
<td>525</td>
<td>468</td>
<td>500.1</td>
<td>1.034</td>
<td>MPT_Startall() - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>21</td>
<td>525</td>
<td>491</td>
<td>523.4</td>
<td>7.241</td>
<td>MPT_UTILTTY::BCAST_0C - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>42</td>
<td>525</td>
<td>489</td>
<td>523.3</td>
<td>7.457</td>
<td>MPT_UTILTTY::BCAST_0T - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>491</td>
<td>491</td>
<td>491</td>
<td>0</td>
<td>MPT_UTILTTY::BCAST_0R - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>2</td>
<td>509</td>
<td>496</td>
<td>502.5</td>
<td>6.5</td>
<td>MPT_UTILTTY::GATHER_1C - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>128</td>
<td>491</td>
<td>441</td>
<td>459.3</td>
<td>12.39</td>
<td>MPT_UTILTTY::GATHER_1D - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>11</td>
<td>508</td>
<td>461</td>
<td>479.5</td>
<td>14.33</td>
<td>MPT_UTILTTY::GATHER_1T - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>526</td>
<td>526</td>
<td>526</td>
<td>0</td>
<td>CHECK_DATA_01 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>526</td>
<td>526</td>
<td>526</td>
<td>0</td>
<td>CHECK_DATA_02 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>502</td>
<td>502</td>
<td>502</td>
<td>0</td>
<td>ITERATE 1 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>503</td>
<td>503</td>
<td>503</td>
<td>0</td>
<td>TTRATE 2 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>503</td>
<td>503</td>
<td>503</td>
<td>0</td>
<td>ITERATE 3 - Memory Headroom Available (MR)</td>
</tr>
<tr>
<td>1</td>
<td>503</td>
<td>503</td>
<td>503</td>
<td>0</td>
<td>TTRATE 4 - Memory Headroom Available (MR)</td>
</tr>
</tbody>
</table>
```
TAU’s High Resolution Timers - CRAYTIMERS

- Instead of the default `gettimeofday()`, it uses low-overhead timers
- On Cray XD1 (Linux x86_64), TAU uses TSC (time stamp counter) registers to access wallclock time
- On Cray X1E, `_rtc()` call is used
- On Cray XT3, `dclock()` call is used
  - `setenv TAU_MAKEFILE <dir>/xt3/lib/Makefile.tau-crays timers- mpi-pdt-pgi`
  - Use `tau_cxx.sh`, `tau_f90.sh` and `tau_cc.sh` as compilers
TAU’s MPI Wrapper Interposition Library

- Uses standard MPI Profiling Interface
  - Provides name shifted interface
    - MPI_Send = PMPI_Send
    - Weak bindings
- Interpose TAU’s MPI wrapper library between MPI and TAU
  - -lmpi replaced by –lTauMpi –lpmpi –lmpi
- No change to the source code! Just re-link the application to generate performance data. Similar to SHMEM/PSHMEM
  - setenv TAU_MAKEFILE
    <dir>/<arch>/lib/Makefile.tau-mpi-[options]
  - Use tau_cxx.sh, tau_f90.sh and tau_cc.sh as compilers
Building Bridges to Other Tools: TAU
TAU Performance System Interfaces

- PDT [U. Oregon, LANL, FZJ] for instrumentation of C++, C99, F95 source code
- PAPI [UTK] & PCL[FZJ] for accessing hardware performance counters data
- DyninstAPI [U. Maryland, U. Wisconsin] for runtime instrumentation
- KOJAK [FZJ, UTK]
  - Epilog trace generation library
  - CUBE callgraph visualizer
  - Opari OpenMP directive rewriting tool
- Vampir/Intel® Trace Analyzer [Pallas/Intel]
- VTF3 trace generation library for Vampir [TU Dresden] (available from TAU website)
- Paraver trace visualizer [CEPBA]
- Jumpshot-4 trace visualizer [MPICH, ANL]
- JVMPi from JDK for Java program instrumentation [Sun]
- Paraprof profile browser/PerfDMF database supports:
  - TAU format
  - Gprof [GNU]
  - HPM Toolkit [IBM]
  - MpiP [ORNL, LLNL]
  - Dynaprof [UTK]
  - PSRun [NCSA]
- PerfDMF database can use Oracle, MySQL or PostgreSQL (IBM DB2 support planned)
Profile Measurement – Three Flavors

- Flat profiles
  - Time (or counts) spent in each routine (nodes in callgraph).
  - Exclusive/inclusive time, no. of calls, child calls
  - E.g.: MPI_Send, foo, …

- Callpath Profiles
  - Flat profiles, plus
  - Sequence of actions that led to poor performance
  - Time spent along a calling path (edges in callgraph)
  - E.g., “main=> f1 => f2 => MPI_Send” shows the time spent in MPI_Send when called by f2, when f2 is called by f1, when it is called by main. Depth of this callpath = 4 (TAU_CALLPATH_DEPTH environment variable)

- Phase based profiles
  - Flat profiles, plus
  - Flat profiles under a phase (nested phases are allowed)
  - Default “main” phase has all phases and routines invoked outside phases
  - Supports static or dynamic (per-iteration) phases
  - E.g., “IO => MPI_Send” is time spent in MPI_Send in IO phase
**TAU Timers and Phases**

- **Static timer**
  - Shows time spent in all invocations of a routine (foo)
  - E.g., “foo()” 100 secs, 100 calls

- **Dynamic timer**
  - Shows time spent in each invocation of a routine
  - E.g., “foo(3)” 4.5 secs, “foo 10” 2 secs (invocations 3 and 10 respectively)

- **Static phase**
  - Shows time spent in all routines called (directly/indirectly) by a given routine (foo)
  - E.g., “foo() => MPI_Send()” 100 secs, 10 calls shows that a total of 100 secs were spent in MPI_Send() when it was called by foo.

- **Dynamic phase**
  - Shows time spent in all routines called by a given invocation of a routine.
  - E.g., “foo(4) => MPI_Send()” 12 secs, shows that 12 secs were spent in MPI_Send when it was called by the 4th invocation of foo.
SUBROUTINE SUM_OF_CUBES
    integer profiler(2)
    save profiler
    INTEGER :: H, T, U

    call TAU_PROFILE_TIMER(profiler, 'SUM_OF_CUBES')
    call TAU_PROFILE_START(profiler)

! This program prints all 3-digit numbers that
! equal the sum of the cubes of their digits.
DO H = 1, 9
    DO T = 0, 9
        DO U = 0, 9
            IF (100*H + 10*T + U == H**3 + T**3 + U**3) THEN
                PRINT '(3I1)', H, T, U
            ENDIF
        END DO
    END DO
END DO
END SUBROUTINE SUM_OF_CUBES
Static Phases and Timers

SUBROUTINE FOO
  integer profiler(2)
  save profiler

  call TAU_PHASE_CREATE_STATIC(profiler, 'foo')
  call TAU_PHASE_START(profiler)
  call bar()
! Here bar calls MPI_Barrier and we evaluate foo=>MPI_Barrier and foo=>bar
  call TAU_PHASE_STOP(profiler)
END SUBROUTINE SUM_OF_CUBES

SUBROUTINE BAR
  integer profiler(2)
  save profiler
  call TAU_PROFILE_TIMER(profiler, 'bar')
  call TAU_PROFILE_START(profiler)
    call MPI_Barrier()
    call TAU_PROFILE_STOP(profiler)
END SUBROUTINE BAR
Dynamic Phases

SUBROUTINE ITERATE(IER, NIT)
    IMPLICIT NONE
    INTEGER IER, NIT
    character(11) taucharary
    integer tauiteration / 0 /
    integer profiler(2) / 0, 0 /
    save profiler, tauiteration

    write (taucharary, '(a8,i3)') 'ITERATE ', tauiteration
    tauiteration = tauiteration + 1

    call TAU_PHASE_CREATE_DYNAMIC(profiler,taucharary)
    call TAU_PHASE_START(profiler)

    IER = 0
    call SOLVE_K_EPSILON_EQ(IER)
    ! Other work

    call TAU_PHASE_STOP(profiler)
### TAU’s ParaProf Profile Browser: Static Timers

<table>
<thead>
<tr>
<th>Metric</th>
<th>Name</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time</td>
<td>Exclusive</td>
<td>Inclusive</td>
</tr>
<tr>
<td>2025.001</td>
<td>2025.001</td>
<td>969969</td>
</tr>
<tr>
<td>6.0</td>
<td>11511</td>
<td>0</td>
</tr>
<tr>
<td>2.1</td>
<td>124265.5</td>
<td>0</td>
</tr>
<tr>
<td>2.0</td>
<td>1201</td>
<td>0.041</td>
</tr>
<tr>
<td>1.3</td>
<td>1200</td>
<td>0.028</td>
</tr>
<tr>
<td>94.6</td>
<td>1200</td>
<td>1.958</td>
</tr>
<tr>
<td>0.6</td>
<td>1401</td>
<td>0.011</td>
</tr>
<tr>
<td>0.6</td>
<td>9884</td>
<td>0.002</td>
</tr>
<tr>
<td>7.6</td>
<td>77600</td>
<td>0.055</td>
</tr>
<tr>
<td>98.1</td>
<td>8000</td>
<td>12.18</td>
</tr>
<tr>
<td>0.4</td>
<td>2400</td>
<td>0.004</td>
</tr>
<tr>
<td>0.3</td>
<td>124253</td>
<td>5.7647E-5</td>
</tr>
<tr>
<td>88.2</td>
<td>995712</td>
<td>1.826</td>
</tr>
<tr>
<td>0.4</td>
<td>30600</td>
<td>0.004</td>
</tr>
<tr>
<td>0.2</td>
<td>1200</td>
<td>0.004</td>
</tr>
<tr>
<td>0.2</td>
<td>7424</td>
<td>0.009</td>
</tr>
<tr>
<td>1.0</td>
<td>34806</td>
<td>8.3319E-4</td>
</tr>
<tr>
<td>1.0</td>
<td>30600</td>
<td>8.3923E-4</td>
</tr>
<tr>
<td>0.1</td>
<td>1200</td>
<td>0.003</td>
</tr>
<tr>
<td>0.01</td>
<td>124265.5</td>
<td>2.7216E-5</td>
</tr>
</tbody>
</table>

Mean Data Statistics: 16pAX200iter/s3d/tau/data/rs/sameer/Users/
## Dynamic Timers

<table>
<thead>
<tr>
<th>Metric Name: Time</th>
<th>Sorted By: Inclusive</th>
<th>Units: hour:minute:seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Total Time</td>
<td>Exclusive</td>
<td>Inclusive</td>
</tr>
<tr>
<td></td>
<td>#Calls</td>
<td>#Child Calls</td>
</tr>
<tr>
<td></td>
<td>Total Time/Call</td>
<td>Name</td>
</tr>
<tr>
<td>100.0</td>
<td>0:0:0.005</td>
<td>0:9:14.379</td>
</tr>
<tr>
<td>99.8</td>
<td>0:0:0.006</td>
<td>0:9:13.509</td>
</tr>
<tr>
<td>97.5</td>
<td>0:0:7.403</td>
<td>0:9:0.34</td>
</tr>
<tr>
<td>97.3</td>
<td>0:0:7.3.977</td>
<td>0:8:3.803</td>
</tr>
<tr>
<td>65.8</td>
<td>0:0:0.104</td>
<td>0:6:4.601</td>
</tr>
<tr>
<td>65.7</td>
<td>0:0:0.01</td>
<td>0:6:4.497</td>
</tr>
<tr>
<td>64.3</td>
<td>0:5:56.271</td>
<td>0:5:56.271</td>
</tr>
<tr>
<td>8.2</td>
<td>0:0:45.564</td>
<td>0:0:45.564</td>
</tr>
<tr>
<td>8.2</td>
<td>0:0:10.008</td>
<td>0:0:45.561</td>
</tr>
<tr>
<td>7.7</td>
<td>0:0:0.012</td>
<td>0:0:42.959</td>
</tr>
<tr>
<td>7.7</td>
<td>0:0:8.289</td>
<td>0:0:42.947</td>
</tr>
<tr>
<td>5.1</td>
<td>0:0:0.017</td>
<td>0:0:28.181</td>
</tr>
<tr>
<td>4.3</td>
<td>0:0:23.902</td>
<td>0:0:23.902</td>
</tr>
<tr>
<td>3.0</td>
<td>0:0:16.848</td>
<td>0:0:16.848</td>
</tr>
<tr>
<td>2.7</td>
<td>0:0:14.499</td>
<td>0:0:14.713</td>
</tr>
<tr>
<td>2.6</td>
<td>0:0:14.499</td>
<td>0:0:14.499</td>
</tr>
<tr>
<td>2.6</td>
<td>0:0:13.624</td>
<td>0:0:14.201</td>
</tr>
<tr>
<td>2.3</td>
<td>0:0:12.725</td>
<td>0:0:12.725</td>
</tr>
<tr>
<td>1.9</td>
<td>0:0:3.436</td>
<td>0:0:10.686</td>
</tr>
<tr>
<td>1.9</td>
<td>0:0:0.128</td>
<td>0:0:10.414</td>
</tr>
<tr>
<td>1.1</td>
<td>0:0:9.494</td>
<td>0:0:9.494</td>
</tr>
<tr>
<td>0.9</td>
<td>0:0:0.01</td>
<td>0:0:4.772</td>
</tr>
<tr>
<td>0.8</td>
<td>0:0:4.628</td>
<td>0:0:4.628</td>
</tr>
<tr>
<td>0.7</td>
<td>0:0:7.675</td>
<td>0:0:3.694</td>
</tr>
</tbody>
</table>

---

**CUG 2006**

**Performance and Memory Evaluation using TAU** 41
### Static Phases

**Performance and Memory Evaluation using TAU**

---

**Figure: TAU Profiler Output**

- **Metric Name:** Time
- **Units:** seconds

<table>
<thead>
<tr>
<th>Exclusive</th>
<th>Inclusive</th>
<th>Calls/Tot.Calls</th>
<th>Name[id]</th>
</tr>
</thead>
<tbody>
<tr>
<td>349.74</td>
<td>349.74</td>
<td>172368.0/172368.0</td>
<td>DTM PHASE[217]</td>
</tr>
<tr>
<td>0.032</td>
<td>0.032</td>
<td>1.0/1701.0</td>
<td>S3D[0]</td>
</tr>
<tr>
<td>44.888</td>
<td>44.588</td>
<td>1200.0/1201.0</td>
<td>REACTION PHASE[218]</td>
</tr>
<tr>
<td>44.621</td>
<td>44.621</td>
<td>1201.0</td>
<td>CHEM_KIN__REACTION_RATE[148]</td>
</tr>
<tr>
<td>25.58</td>
<td>25.58</td>
<td>1200.0/1200.0</td>
<td>SOOT PHASE[214]</td>
</tr>
<tr>
<td>21.781</td>
<td>468.5</td>
<td>1200.0/1200.0</td>
<td>S3D[0]</td>
</tr>
<tr>
<td>16.301</td>
<td>16.301</td>
<td>1401.0/1401.0</td>
<td>S3D[0]</td>
</tr>
<tr>
<td>10.52</td>
<td>10.52</td>
<td>67823.0/108081.5</td>
<td>S3D[0]</td>
</tr>
<tr>
<td>1.565</td>
<td>1.565</td>
<td>9.0/108081.5</td>
<td>TO PHASE[59]</td>
</tr>
<tr>
<td>1.287</td>
<td>1.287</td>
<td>25200.0/108081.5</td>
<td>SOOT PHASE[214]</td>
</tr>
<tr>
<td>0.433</td>
<td>0.433</td>
<td>3349.5/108081.5</td>
<td>DTM PHASE[217]</td>
</tr>
<tr>
<td>0.298</td>
<td>0.298</td>
<td>11700.0/108081.5</td>
<td>BOUNDARY CONDITION PHASE[220]</td>
</tr>
<tr>
<td>14.103</td>
<td>14.103</td>
<td>108081.5</td>
<td>MPI_Recv[174]</td>
</tr>
<tr>
<td>5.05</td>
<td>5.05</td>
<td>1616.0/4514.0</td>
<td>S3D[0]</td>
</tr>
<tr>
<td>0.341</td>
<td>0.341</td>
<td>202.0/4514.0</td>
<td>TO PHASE[59]</td>
</tr>
<tr>
<td>1.74</td>
<td>1.74</td>
<td>1200.0/4514.0</td>
<td>SOOT PHASE[214]</td>
</tr>
<tr>
<td>4.858</td>
<td>4.858</td>
<td>1496.0/4514.0</td>
<td>DTM PHASE[217]</td>
</tr>
<tr>
<td>13.488</td>
<td>13.488</td>
<td>4514.0</td>
<td>MPI_Barrier[49]</td>
</tr>
</tbody>
</table>

---

**Note:**

- **MPI_Barrier took 4.85 secs out of 13.48 secs in the DTM Phase**

---

_CUG 2006_
### Dynamic Phases

The first iteration was expensive for INT_RTE. It took 27.89 secs. Other iterations took less time – 14.2, 10.5, 10.3, 10.5 seconds.
Dynamic Phases

Time spent in MPI_Barrier, MPI_Recv, ... in DTM ITERATION 1

Breakdown of time spent in MPI_Isend based on its static and dynamic parent phases
ParaProf – Manager Window

performance database

derived performance metrics
Performance Database: Storage of MetaData

ParaProf Manager

File Options Help

Applications
- Standard Applications
  - Default App
    - Default Exp
      - 16pAX200iter/s3d/taudata/rs/sameer/Users/
        - Time
  - Runtime Applications
  - DB Applications
    - AORSA2D
      - Basic run-time profiling for Socorro
      - Heap memory management for Socorro
    - hydroshock
    - MFIX
    - S3D
      - AIX
        - 16pAX10iter/s3d/taudata/rs/sameer/Users/
        - Time
        - 16pAX200iter/s3d/taudata/rs/sameer/Users/
          - Time
        - 16pAXcall200iter/s3d/taudata/rs/sameer/Users/
          - Time

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>16pAXcall200iter/s3d/taudata/rs/sameer/Users/</td>
</tr>
<tr>
<td>Application ID</td>
<td>8</td>
</tr>
<tr>
<td>Experiment ID</td>
<td>16</td>
</tr>
<tr>
<td>Trial ID</td>
<td>34</td>
</tr>
<tr>
<td>time</td>
<td></td>
</tr>
<tr>
<td>problem_definition</td>
<td>nx_g=400, ny_g=400, npx=1, npy=4, npz=1</td>
</tr>
<tr>
<td>node_count</td>
<td>16</td>
</tr>
<tr>
<td>contexts_per_node</td>
<td>1</td>
</tr>
<tr>
<td>threads_per_context</td>
<td>1</td>
</tr>
<tr>
<td>userdata</td>
<td>i_time_end=200, i_time_save=200,TAU_CALLPATH_DEPTH=2</td>
</tr>
</tbody>
</table>

Load Trial

Trial Type: Tau profiles

Select Directory: /Users/sameer/rs/taudata/s3d

Cancel Ok
ParaProf – Full Profile (MFIX, NETL)
ParaProf – Mean Profile (MFIX)
### ParaProf – Memory Profiling (MFIX)

- **Sorted By: Mean Value**

<table>
<thead>
<tr>
<th>Num Samples</th>
<th>Max</th>
<th>Min</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>220320</td>
<td>110160</td>
<td>201960</td>
<td>41054</td>
<td>Message size for scatter</td>
</tr>
<tr>
<td>14063</td>
<td>21248</td>
<td>4</td>
<td>3594.7</td>
<td>4495.4</td>
<td>Message size received from all nodes</td>
</tr>
<tr>
<td>14062</td>
<td>21248</td>
<td>4</td>
<td>3594.7</td>
<td>4495.4</td>
<td>Message size sent to all nodes</td>
</tr>
<tr>
<td>1</td>
<td>655</td>
<td>655</td>
<td>655</td>
<td>0</td>
<td>MPI_Init() – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>655</td>
<td>655</td>
<td>655</td>
<td>0</td>
<td>MPI_Init(): – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>BSD.Painting: – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>GET_DATA – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>GET_RUN_ID – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>INIT_NAMELIST – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>MACHINE_CONS – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>READ_NAMELIST – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>USR_INIT_NAMELIST – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>MPI_Comm_rank() – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>MPI_Comm_size() – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>589</td>
<td>589</td>
<td>589</td>
<td>0</td>
<td>CHECK_DATA_NO – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>577</td>
<td>577</td>
<td>577</td>
<td>0</td>
<td>GRIDMAP:GRIDMAP_INIT – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>577</td>
<td>577</td>
<td>577</td>
<td>0</td>
<td>GRIDMAP:GRIDMAP_INIT – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>579</td>
<td>575</td>
<td>576.93</td>
<td>1.02</td>
<td>SET_MAX2 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>470</td>
<td>579</td>
<td>575</td>
<td>576.93</td>
<td>1.02</td>
<td>LIN pharmacies – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>750</td>
<td>579</td>
<td>575</td>
<td>576.93</td>
<td>1.02</td>
<td>PARSE_LINE – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>586</td>
<td>579</td>
<td>575</td>
<td>576.93</td>
<td>1.04</td>
<td>BLANK_LINE – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>577</td>
<td>575</td>
<td>575.29</td>
<td>0.7</td>
<td>REMOVE_COMMENT – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>577</td>
<td>575</td>
<td>575.29</td>
<td>0.7</td>
<td>SDBM COMMENT – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>577</td>
<td>575</td>
<td>575.29</td>
<td>0.7</td>
<td>MPI_Send_init() – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>577</td>
<td>575</td>
<td>575.29</td>
<td>0.7</td>
<td>ALLOCATE ARRAYS – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>14</td>
<td>530</td>
<td>528</td>
<td>528.43</td>
<td>0.821</td>
<td>OPEN FILES – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>527</td>
<td>527</td>
<td>527</td>
<td>0</td>
<td>WRITE_HEADER – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>526</td>
<td>526</td>
<td>526</td>
<td>0</td>
<td>CHECK_DATA_01 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_02 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_CELL – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_03 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_04 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_05 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_06 – Memory Headroom Available (MB)</td>
</tr>
<tr>
<td>1</td>
<td>525</td>
<td>525</td>
<td>525</td>
<td>0</td>
<td>CHECK_DATA_07 – Memory Headroom Available (MB)</td>
</tr>
</tbody>
</table>
ParaProf – 3D Scatter Plot (MFIX)
### Gprof Style Callpath View in Paraprof (SAGE)

<table>
<thead>
<tr>
<th>Exclusive</th>
<th>Inclusive</th>
<th>Calls/Tot.Calls</th>
<th>Name[id]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8584</td>
<td>1.8584</td>
<td>1196/13188</td>
<td>TOKEN_MODULE::TOKEN_GS_I [521]</td>
</tr>
<tr>
<td>0.584</td>
<td>0.584</td>
<td>234/13188</td>
<td>TOKEN_MODULE::TOKEN_GS_L [544]</td>
</tr>
<tr>
<td>25.0819</td>
<td>25.0819</td>
<td>11758/13188</td>
<td>TOKEN_MODULE::TOKEN_GS_R8 [734]</td>
</tr>
<tr>
<td>--&gt; 27.5242</td>
<td>27.5242</td>
<td>13188</td>
<td>MPI_Waitall() [525]</td>
</tr>
</tbody>
</table>

| 17.9579 | 39.1657 | 156/156       | DERIVATIVE_MODULE::DERIVATIVES_NOFACE [841]           |
| --> 17.9579 | 39.1657 | 156           | DERIVATIVE_MODULE::DERIVATIVES_FACE [843]             |
| 0.0156   | 0.0195   | 312/312       | TIMER_MODULE::TIMERSET [77]                           |
| 0.1133   | 9.1269   | 2340/2340     | MESSAGE_MODULE::CLONE_GET_R8 [808]                    |
| 0.1602   | 11.4608  | 4056/4056     | MESSAGE_MODULE::CLONE_PUT_R8 [850]                    |
| 0.0059   | 0.6006   | 117/117       | MESSAGE_MODULE::CLONE_PUT_I [856]                     |

| 14.1151  | 21.6209  | 5/5           | MATRIX_MODULE::MCGDS [1443]                           |
| --> 14.1151 | 21.6209 | 5             | MATRIX_MODULE::CSR_cg_solver [1470]                  |
| 0.0654   | 1.2617   | 1005/1005     | TOKEN_MODULE::TOKEN_GET_R8 [769]                      |
| 0.0557   | 5.2714   | 1005/1005     | TOKEN_MODULE::TOKEN_REDUCTION_R8_S [1475]             |
| 0.0703   | 0.9726   | 1000/1000     | TOKEN_MODULE::TOKEN_REDUCTION_R8_V [208]              |
In 51\textsuperscript{st} iteration, time spent in MPI\_Waitall was 85.81 secs

Total time spent in MPI\_Waitall was 4137.9 secs across all 92 iterations

<table>
<thead>
<tr>
<th>Metric Name: Time Sorted By X</th>
<th>X</th>
<th>Metric Name: Time Sorted By X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorted By X</td>
<td></td>
<td>Sorted By X</td>
<td></td>
</tr>
<tr>
<td>Units: seconds</td>
<td></td>
<td>Units: seconds</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>217</td>
<td>65</td>
<td>217</td>
</tr>
<tr>
<td>55</td>
<td>321</td>
<td>55</td>
<td>321</td>
</tr>
<tr>
<td>51</td>
<td>351</td>
<td>51</td>
<td>351</td>
</tr>
<tr>
<td>86</td>
<td>81</td>
<td>86</td>
<td>81</td>
</tr>
<tr>
<td>75</td>
<td>668</td>
<td>75</td>
<td>668</td>
</tr>
<tr>
<td>78</td>
<td>908</td>
<td>78</td>
<td>908</td>
</tr>
<tr>
<td>69</td>
<td>668</td>
<td>69</td>
<td>668</td>
</tr>
<tr>
<td>58</td>
<td>461</td>
<td>58</td>
<td>461</td>
</tr>
<tr>
<td>85</td>
<td>117</td>
<td>85</td>
<td>117</td>
</tr>
<tr>
<td>47</td>
<td>88</td>
<td>47</td>
<td>88</td>
</tr>
<tr>
<td>46</td>
<td>436</td>
<td>46</td>
<td>436</td>
</tr>
<tr>
<td>46</td>
<td>287</td>
<td>46</td>
<td>287</td>
</tr>
<tr>
<td>45</td>
<td>728</td>
<td>45</td>
<td>728</td>
</tr>
<tr>
<td>45</td>
<td>244</td>
<td>45</td>
<td>244</td>
</tr>
<tr>
<td>45</td>
<td>285</td>
<td>45</td>
<td>285</td>
</tr>
<tr>
<td>46</td>
<td>168</td>
<td>46</td>
<td>168</td>
</tr>
<tr>
<td>46</td>
<td>907</td>
<td>46</td>
<td>907</td>
</tr>
<tr>
<td>51</td>
<td>488</td>
<td>51</td>
<td>488</td>
</tr>
<tr>
<td>44</td>
<td>446</td>
<td>44</td>
<td>446</td>
</tr>
<tr>
<td>44</td>
<td>474</td>
<td>44</td>
<td>474</td>
</tr>
<tr>
<td>41</td>
<td>176</td>
<td>41</td>
<td>176</td>
</tr>
<tr>
<td>52</td>
<td>452</td>
<td>52</td>
<td>452</td>
</tr>
<tr>
<td>44</td>
<td>281</td>
<td>44</td>
<td>281</td>
</tr>
<tr>
<td>44</td>
<td>288</td>
<td>44</td>
<td>288</td>
</tr>
<tr>
<td>43</td>
<td>714</td>
<td>43</td>
<td>714</td>
</tr>
<tr>
<td>42</td>
<td>170</td>
<td>42</td>
<td>170</td>
</tr>
<tr>
<td>41</td>
<td>175</td>
<td>41</td>
<td>175</td>
</tr>
<tr>
<td>45</td>
<td>348</td>
<td>45</td>
<td>348</td>
</tr>
<tr>
<td>45</td>
<td>728</td>
<td>45</td>
<td>728</td>
</tr>
<tr>
<td>38</td>
<td>728</td>
<td>38</td>
<td>728</td>
</tr>
<tr>
<td>46</td>
<td>46</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>52</td>
<td>453</td>
<td>52</td>
<td>453</td>
</tr>
<tr>
<td>44</td>
<td>281</td>
<td>44</td>
<td>281</td>
</tr>
<tr>
<td>44</td>
<td>288</td>
<td>44</td>
<td>288</td>
</tr>
<tr>
<td>38</td>
<td>728</td>
<td>38</td>
<td>728</td>
</tr>
<tr>
<td>38</td>
<td>728</td>
<td>38</td>
<td>728</td>
</tr>
</tbody>
</table>

Dynamic phases one per iteration
### ParaProf - Statistics Table (Uintah)

![ParaProf Statistics Table](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>P_WALL_CLOCK_TIME</th>
<th>Calls</th>
<th>Child Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>main() void (int, char **)</td>
<td>0.015</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Uintah::ProcessorGroup *Uintah::Parallel::getRootProcessor()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uintah::SimpleSimulationController &amp;Uintah::SimpleSimulation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uintah::SimulationController &amp;Uintah::SimulationController::Si</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool Uintah::Parallel::usingMPI()</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>int Uintah::Parallel::getMPIRank()</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>void Uintah::OnDemandDataWarehouse::~OnDemandDataWarehouse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void Uintah::Parallel::determineRunningUnderMPI(int, char **, const</td>
<td>0.002</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>void Uintah::Parallel::finalizeManager(Uintah::Parallel::Circuit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void Uintah::Parallel::initializeManager(int &amp;, char **, const</td>
<td>0.001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MPI_Comm_rank()</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MPI_Comm_size()</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MPI_Init_thread()</td>
<td>6.327</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>void Uintah::Parallel::noThreading()</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>void Uintah::SimpleSimulationController::run() Uintah::Simple</td>
<td>0.074</td>
<td>1</td>
<td>154</td>
</tr>
<tr>
<td>MPI_Scheduler::actuallyCompile()</td>
<td>0.109</td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>MPI_Scheduler::execute()</td>
<td>27.68</td>
<td>11</td>
<td>3,460</td>
</tr>
<tr>
<td>MPI_Reduce()</td>
<td>0.001</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>Uintah::DataWarehouse::ScrubMode Uintah::OnDemandDataWarehouse</td>
<td>0.001</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Uintah::OnDemandDataWarehouse &amp;Uintah::OnDemandDataWarehouse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool Uintah::OnDemandDataWarehouse::timestepAbort</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>bool Uintah::OnDemandDataWarehouse::timestepRestart</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>bool Uintah::SimpleSimulationController::needRecompiler</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>void Uintah::OnDemandDataWarehouse::get(Uintah::Real</td>
<td>0.001</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>void Uintah::OnDemandDataWarehouse::override(const Uintah::Real</td>
<td>0.001</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>
ParaProf – Histogram View (Miranda)

- Scalable 2D displays

![Histograms for 8k processors and 16k processors](image)
ParaProf – Callgraph View (MFIX)
ParaProf – 3D Full Profile (Miranda)

16k processors
ParaProf Bar Plot (Zoom in/out +/-)
ParaProf – 3D Scatterplot (Miranda)

- Each point is a “thread” of execution (32k cpus)
- A total of four metrics shown in relation
- ParaVis 3D profile visualization library
  - JOGL
TAU Performance System Status

- Computing platforms (selected)
  - Cray XT3/X1E/XD1/T3E/SV1, IBM SP/pSeries/BGL, SGI Altix/Origin, HP (Compaq) SC (Tru64), Sun, Linux clusters (IA-32/64, Alpha, PPC, PA-RISC, Power, Opteron), Apple (G4/5, OS X), Hitachi SR8000, NEC SX-5/6, FreeBSD, Windows …

- Programming languages
  - C, C++, Fortran 77/90/95, HPF, Java, Python

- Thread libraries (selected)
  - pthreads, OpenMP, SGI sproc, Java, Windows, Charm++

- Compilers (selected)
  - Cray, Pathscale, Intel, PGI, GNU, Fujitsu, Sun, IBM, HP, NEC, Absoft, Lahey, Nagware
Concluding Discussion

- Performance tools must be used effectively
- More intelligent performance systems for productive use
  - Evolve to application-specific performance technology
  - Deal with scale by “full range” performance exploration
  - Autonomic and integrated tools
  - Knowledge-based and knowledge-driven process
- Performance observation methods do not necessarily need to change in a fundamental sense
  - More automatically controlled and efficiently use
- Develop next-generation tools and deliver to community
- Open source with support by ParaTools, Inc.
Support Acknowledgements

- Department of Energy (DOE) contracts
  - Office of Science
  - University of Utah ASC
  - LLNL ASC/NNSA Level 3
  - Los Alamos National Laboratory
  - Argonne National Laboratory (ZeptoOS)
- DoD PET HPCMO
- Oak Ridge National Laboratory
- Pittsburgh Supercomputing Center
- NETL/Aeolus Research
  - Aytekin Gel
- http://www.cs.uoregon.edu/research/tau