Intuitive Performance Engineering at the Exascale with TAU and TAU Commander

Presented to
ATPESC 2017 Participants

Dr. John C. Linford
ParaTools, Inc.

Q Center, St. Charles, IL (USA)
Date 08/08/2017
Cray-2 (1985)

- 4 vector processors
- 1.9 gigaflops (0.0095 gigaflops/Watt)
Aurora (1985+34)

- >3,200,000 cores
- 180 petaflops (5.62 gigaflops/Watt)
Your Brain is Not Enough

- Supercomputers are incomprehensibly complex.
- Naïve optimization may harm performance.
- Performance engineering **tools are essential** for realizing performance at scale.
NASA, DOD, DOE, Industry

“These days I get excited about 1-2% speedups that I find....quite unusual to find something of this magnitude these days, especially with just a 2-line fix in the code! :)

33% Faster

3x Faster
Software Performance Engineering
Identifying and Resolving Performance Issues

Profile

Identify Hotspots

File I/O
- 50x
- Yes
- Buffers, data formats, in-memory filesystems

Communication
- 10x
- No
- Collectives, blocking, non-blocking, topology, load balance

Memory
- 5x
- No
- Bandwidth/latency, cache utilization

Compute
- 2x
- No
- Vectors, branches, integer, floating point

Focus Optimization

Refine the Profile

Yes

Yes

Yes

Yes

Yes
TAU Commander

- **Universal tool or integrated toolkit**
- **Unbiased, accurate measurements**
  - File I/O: serial and parallel
  - Communication: inter- and intra-node
  - Memory: allocation and access
  - CPU: vectorization, cache utilization, etc.
- **Minimal overhead**
  - Provide multiple measurement methods
  - Focus on one performance aspect at a time
- **Easy to use**
  - Intuitive, systematic, and well documented
  - Easy to understand and configure
TAU Commander’s Approach

• Say where you’re going, not how to get there

• **Experiments** give context to the user’s actions
  – Defines desired metrics and measurement approach
  – Defines operating environment
  – Establishes a baseline for error checking

VS.

Argonne
NATIONAL LABORATORY

Exascale Computing Project
T-A-M Model for Performance Engineering

- **Target**
  - Installed software
  - Available compilers
  - Host architecture/OS

- **Application**
  - MPI, OpenMP, CUDA, OpenACC, etc.

- **Measurement**
  - Profile, trace, or both
  - Sample, source inst...

Experiment = (Target, Application, Measurement)
Which platform is best for my application?

- Many targets:
  - Different MPI implementations
  - Different CPU architectures
  - GPU vs MIC
  - Cray vs SGI
- One measurement
- One application
What are the performance characteristics of my application?

- One target
- Many measurements:
  - File I/O
  - Communication
  - Memory allocation
  - Performance counters
  - Vectorization
- One application
How well does my target perform various tasks?

• One target
• One measurement
• Many applications:
  • Compute bound
    • Dense LA
  • Memory bound
    • Sparse LA
    • Graph
  • Scaling
    • Thread-level
    • Process-level
Getting Started with TAU Commander

1. `tau` initialize
2. `tau` oshf90 *.f90 -o foo
3. `tau` oshrun -np 64 ./foo
4. `tau` show

• This works on any supported system, even if TAU is not installed or has not been configured appropriately.

• TAU and all its dependencies will be downloaded and installed if required.

Just put `tau` in front of everything and see what happens.
TAU Commander Online Help

Positional Arguments:
<subcommand> See subcommand descriptions below.
[options] Options to be passed to <subcommand>.

Optional Arguments:
-<v>, --version Show program's version number and exit.
-<h>, --help Show this help message and exit.
-<v>, --verbose Show debugging messages.

Configuration Subcommands:
application Create and manage application configurations.
experiment Create and manage experiments.
measurement Create and manage measurement configurations.
project Create and manage project configurations.
target Create and manage target configurations.
trial Create and manage experiment trials.

Subcommands:
build Instrument programs during compilation and/or linking.
configure Configure TAU Commander.
dashboard Show all project components.
help Show help for a command or suggest actions for a file.
initialize Initialize TAU Commander.
select Create a new experiment or select an existing experiment.

Shortcuts:
tau <compiler> Execute a compiler command
- Example: tau gcc -o a.out
  - Alias for 'tau build <compiler>'
tau <program> Gather data from a program
- Example: tau ./a.out
  - Alias for 'tau trial create <program>'
tau metrics Show metrics available in the current experiment
  - Alias for 'tau target metrics'
tau select Select configuration objects to create a new experiment
  - Alias for 'tau experiment create'
tau show Show data from the most recent trial
  - Alias for 'tau trial show'

See 'tau help <subcommand>' for more information on <subcommand>.
Gathering Performance Data
Measurement Approaches

**Profiling** shows how much time

**Tracing** shows the order of events
Types of Performance Profiles

• **Flat** profiles
  – Metric (e.g., time) spent in an event
  – Exclusive/inclusive, # of calls, child calls, …

• **Callpath** profiles
  – Time spent along a calling path (edges in callgraph)
  – “main=> f1 => f2 => MPI_Send”

• **Phase** profiles
  – Flat profiles under a phase (nested phases allowed)
  – Default “main” phase
  – Supports static or dynamic (e.g. per-iteration) phases
How much data do you want?

Limited Profile
Loop Profile
Callpath Profile
Flat Profile
Phase Profile
Trace

All levels support multiple metrics/counters
Performance Data Measurement

Direct via Probes

```c
call TAU_START('potential')
// code
call TAU_STOP('potential')
```

- Exact measurement
- Fine-grain control
- Calls inserted into code

Indirect via Sampling

- No code modification
- Minimal effort
- Relies on debug symbols (-g)
TAU Commander Case Study
Step 1: Initialize TAU Project

$ cd ISx

$ tau initialize --shmем

- Creates a new project configuration using defaults
- Project files exist in a directory named “.tau”
- Like git, all directories below the directory containing the “.tau” directory can access the project
  - E.g. `tau dashboard` works in miniapp1/baseline
Project Initialization (`tau initialize`)

- Compiler detection
- Project initialization
- Download and install PDT
- TAU installation progress
Project Dashboard (`tau dashboard`)

**Project Configuration**
`/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/tau/project.json`

<table>
<thead>
<tr>
<th>Name</th>
<th>Targets</th>
<th>Applications</th>
<th>Measurements</th>
<th># Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISx</td>
<td>cori09</td>
<td>ISx</td>
<td>sample, profile, trace</td>
<td>1</td>
</tr>
</tbody>
</table>

**Targets in project 'ISx'**

<table>
<thead>
<tr>
<th>Name</th>
<th>Most OS</th>
<th>Host Arch</th>
<th>Most Compilers</th>
<th>MPI Compilers</th>
<th>SHMEM Compilers</th>
</tr>
</thead>
<tbody>
<tr>
<td>cori09</td>
<td>CNL</td>
<td>x86_64</td>
<td>Cray</td>
<td>Cray</td>
<td>Cray</td>
</tr>
</tbody>
</table>

**Applications in project 'ISx'**

<table>
<thead>
<tr>
<th>Name</th>
<th>Linkage</th>
<th>OpenMP</th>
<th>Pthreads</th>
<th>TBB</th>
<th>CUDA</th>
<th>OpenCL</th>
<th>SHMEM</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISx</td>
<td>static</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Measurements in project 'ISx'**

<table>
<thead>
<tr>
<th>Name</th>
<th>Profile</th>
<th>Trace</th>
<th>Sample</th>
<th>Source Inst.</th>
<th>Compiler Inst.</th>
<th>OpenMP</th>
<th>CUDA</th>
<th>I/O</th>
<th>MPI</th>
<th>SHMEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample</td>
<td>tau</td>
<td>none</td>
<td>Yes</td>
<td>never</td>
<td>never</td>
<td>ignore</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>profile</td>
<td>tau</td>
<td>none</td>
<td>No</td>
<td>automatic</td>
<td>never</td>
<td>ignore</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>trace</td>
<td>none</td>
<td>slog2</td>
<td>No</td>
<td>automatic</td>
<td>never</td>
<td>ignore</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Experiments in project 'ISx'**

<table>
<thead>
<tr>
<th>Name</th>
<th>Trials</th>
<th>Data Size</th>
<th>Target</th>
<th>Application</th>
<th>Measurement</th>
<th>TAU Makefile</th>
</tr>
</thead>
<tbody>
<tr>
<td>cori09-ISx-sample</td>
<td>0</td>
<td>0.8B</td>
<td>cori09</td>
<td>ISx</td>
<td>sample</td>
<td>Makefile.tau-intel-3f4x231a-shmem-pdt</td>
</tr>
</tbody>
</table>

Selected Experiment: cori09-ISx-sample
Step 2: Use `tau` to compile

- TAU Commander constructs a new compilation command line.
  - May replace compiler commands with TAU’s compiler wrapper scripts.
  - May set environment variables, parse configuration files, etc.
  - If no changes are required then nothing is changed.

Prepend `tau` command to compiler command

Compile as normal
Step 3: Use `tau` to run

1. Prepend `tau` command to command line
2. Application executes, possibly with `tau_exec`
3. New data is added to the performance database
Step 4: Use `tau` to view data (`tau show`)
Create a New Experiment

Select a new measurement to create a new experiment

TAU Performance System® automatically reconfigured and recompiled.

User advised that an application rebuild is required to use source-based instrumentation.
Create and Select KNL Target

```
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $ tau target copy cori09 cori.KNL --arch=KNL
[TAU] Added target 'cori.KNL' to project configuration 'ISx'.
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $ tau select cori.KNL ISx profile
[TAU] Created a new experiment 'cori.KNL-ISx-profile'
[TAU] Installing PDT to '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/pdt/04860562'
[TAU] Using PDT source archive '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz'
[TAU] Checking contents of '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz'
[TAU] Completed in 8.297 seconds
[TAU] Extracting '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz' to create '/dev/shm/tmpedFKyo/./pdtoolkit-3.24'
[TAU] Completed in 4.488 seconds
[TAU] Configuring PDT...
[TAU] Completed in 18.205 seconds
[TAU] Compiling PDT...
[TAU] Completed in 5.997 seconds
[TAU] Installing PDT...
[TAU] Completed in 0.116 seconds
[TAU] Checking installed files...
[TAU] Completed in 0.115 seconds
[TAU] Setting file permissions...
[TAU] Completed in 0.179 seconds
[TAU] Verifying PDT installation...
[TAU] Installing TAU Performance System at '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/tau/./tau-2.26.2'
[TAU] Configuring TAU...
[TAU] Completed in 10.261 seconds
[TAU] Compiling and installing TAU...
[TAU] Completed in 30.248 seconds
[TAU] Checking installed files...
[TAU] Completed in 0.215 seconds
[TAU] Setting file permissions...
[TAU] Completed in 0.484 seconds
[TAU] Verifying TAU Performance System installation...
[TAU] Selected experiment 'cori.KNL-ISx-profile'.
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $
```
Performance Data Analysis
How Much Time per Code Region?
How Many Instructions per Code Region?
How Many L1 or L2 Cache Misses?
### How Much Memory Does the Code Use?

<table>
<thead>
<tr>
<th>Name</th>
<th>Total</th>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TAU application</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>14,236,992.16</td>
<td>27,169.781</td>
<td>49,152</td>
<td>1</td>
<td>524.001</td>
<td>2,013.103</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>13,132,932</td>
<td>23,292</td>
<td>262,144</td>
<td>1</td>
<td>563.839</td>
<td>4,492.057</td>
</tr>
<tr>
<td><strong>MPI_Finalize()</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>1,298,918.679</td>
<td>1,495.125</td>
<td>461,766.25</td>
<td>4</td>
<td>868.769</td>
<td>16,928.073</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>48,150</td>
<td>20</td>
<td>36,032</td>
<td>11</td>
<td>2,407.5</td>
<td>7,911.992</td>
</tr>
<tr>
<td><strong>OurMain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>3,465</td>
<td>9</td>
<td>769</td>
<td>32</td>
<td>385</td>
<td>260.2</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>4,314</td>
<td>12</td>
<td>769</td>
<td>32</td>
<td>359.5</td>
<td>240.981</td>
</tr>
<tr>
<td><strong>&lt;module&gt;</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>293,088</td>
<td>449</td>
<td>32,564</td>
<td>32</td>
<td>652.757</td>
<td>1,526.875</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>311,966</td>
<td>493</td>
<td>32,564</td>
<td>32</td>
<td>632.791</td>
<td>1,460.941</td>
</tr>
<tr>
<td><strong>staticCFD</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>init</strong></strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>&lt;module&gt;</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Utilization (heap, in KB)</td>
<td>849,270.344</td>
<td>192,825.168</td>
<td>0.078</td>
<td>147,832.141</td>
<td>62,621.576</td>
<td></td>
</tr>
<tr>
<td>Message size for all-gather</td>
<td>4,096</td>
<td>1</td>
<td>4,096</td>
<td>4,096</td>
<td>4,096</td>
<td>0</td>
</tr>
<tr>
<td>Message size for all-reduce</td>
<td>23,340</td>
<td>843</td>
<td>320</td>
<td>4</td>
<td>27,687</td>
<td>64,653</td>
</tr>
<tr>
<td>Message size for all-to-all</td>
<td>104</td>
<td>26</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Message size for broadcast</td>
<td>24,923</td>
<td>206</td>
<td>8,788</td>
<td>4</td>
<td>120,985</td>
<td>860,992</td>
</tr>
<tr>
<td>Message size for reduce</td>
<td>8,912</td>
<td>8</td>
<td>8,788</td>
<td>4</td>
<td>1,114</td>
<td>2,900,511</td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>27,417,881.391</td>
<td>413,600.719</td>
<td>24,025.667</td>
<td>1</td>
<td>66,290.701</td>
<td>199,538.234</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>27,468,709,355.914</td>
<td>435,669.625</td>
<td>24,025.667</td>
<td>0</td>
<td>63,049.402</td>
<td>195,561.193</td>
</tr>
</tbody>
</table>

**High-water mark**
How Much Memory Does the Code Use?

<table>
<thead>
<tr>
<th>Name</th>
<th>Total</th>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>14,236,992.16</td>
<td>27,169.781</td>
<td>49,152</td>
<td>1</td>
<td>524.001</td>
<td>2,013.103</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>13,132,932</td>
<td>23,292</td>
<td>262,144</td>
<td>1</td>
<td>563.839</td>
<td>4,492.057</td>
</tr>
<tr>
<td>MPI_Finalize()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>1,298,918.679</td>
<td>1,495.125</td>
<td>461,766.25</td>
<td>4</td>
<td>868.769</td>
<td>16,928.073</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>48,150</td>
<td>20</td>
<td>36,032</td>
<td>11</td>
<td>2,407.5</td>
<td>7,911.992</td>
</tr>
<tr>
<td>OurMain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>3,465</td>
<td>9</td>
<td>769</td>
<td>32</td>
<td>385</td>
<td>260.2</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>4,314</td>
<td>12</td>
<td>769</td>
<td>32</td>
<td>359.5</td>
<td>240.981</td>
</tr>
<tr>
<td>&lt;module&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>293,088</td>
<td>449</td>
<td>32,564</td>
<td>32</td>
<td>652.757</td>
<td>1,526.875</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>311,966</td>
<td>493</td>
<td>32,564</td>
<td>32</td>
<td>632.791</td>
<td>1,460.941</td>
</tr>
<tr>
<td>staticCFD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>init</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;module&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Utilization (heap, in KB)</td>
<td>849,270.344</td>
<td>192,825.168</td>
<td>0.078</td>
<td>147,832.141</td>
<td>62,621.576</td>
<td></td>
</tr>
<tr>
<td>Message size for all-gather</td>
<td>4,096</td>
<td>1</td>
<td>4,096</td>
<td>4,096</td>
<td>4,096</td>
<td>0</td>
</tr>
<tr>
<td>Message size for all-reduce</td>
<td>23,340</td>
<td>843</td>
<td>320</td>
<td>4</td>
<td>27.687</td>
<td>64.653</td>
</tr>
<tr>
<td>Message size for all-to-all</td>
<td>104</td>
<td>26</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Message size for broadcast</td>
<td>24,923</td>
<td>206</td>
<td>8,788</td>
<td>4</td>
<td>120.985</td>
<td>860.992</td>
</tr>
<tr>
<td>Message size for reduce</td>
<td>8,912</td>
<td>8</td>
<td>8,788</td>
<td>4</td>
<td>1,114</td>
<td>2,900.511</td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>27,417,881,391.51</td>
<td>413,600,719</td>
<td>24,025,567</td>
<td>1</td>
<td>66,290,701</td>
<td>199,538,234</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>27,468,709,355,914</td>
<td>435,669,625</td>
<td>24,025,567</td>
<td>0</td>
<td>63,049,402</td>
<td>195,561,193</td>
</tr>
</tbody>
</table>

Total allocated/deallocated
# Where is Memory Allocated /Deallocated?

<table>
<thead>
<tr>
<th>Name</th>
<th>Total</th>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU_application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>14,236,992.16</td>
<td>27,169.781</td>
<td>49,152</td>
<td>1</td>
<td>524.001</td>
<td>2,013.103</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>13,132,932</td>
<td>23,292</td>
<td>262,144</td>
<td>1</td>
<td>563.839</td>
<td>4,492.057</td>
</tr>
<tr>
<td>MPI_Finalize()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OurMain()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>1,298,918.679</td>
<td>1,495.125</td>
<td>461,766.25</td>
<td>4</td>
<td>868.769</td>
<td>16,928.073</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>48,150</td>
<td>20</td>
<td>36,032</td>
<td>11</td>
<td>2,407.5</td>
<td>7,911.992</td>
</tr>
<tr>
<td>OurMain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>3,465</td>
<td>9</td>
<td>769</td>
<td>32</td>
<td>385</td>
<td>260.2</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>4,314</td>
<td>12</td>
<td>769</td>
<td>32</td>
<td>359.5</td>
<td>240.981</td>
</tr>
<tr>
<td>&lt;module&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free size (bytes)</td>
<td>293,088</td>
<td>449</td>
<td>32,564</td>
<td>32</td>
<td>652.757</td>
<td>1,526.875</td>
</tr>
<tr>
<td>malloc size (bytes)</td>
<td>311,966</td>
<td>493</td>
<td>32,564</td>
<td>32</td>
<td>632.791</td>
<td>1,460.941</td>
</tr>
<tr>
<td>staticCFD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>init</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;module&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Allocation / Deallocation Events**
What are the I/O Characteristics?

- **Write bandwidth per file**
- **Bytes written to each file**
What are the I/O Characteristics?

<table>
<thead>
<tr>
<th>Name</th>
<th>Total</th>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoadBodyEuler</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoadMesh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI-IO Bytes Written</td>
<td>4,328,712</td>
<td>144</td>
<td>893,152</td>
<td>0</td>
<td>30,060.5</td>
<td>128,042.696</td>
</tr>
<tr>
<td>MPI-IO Write Bandwidth (MB/s)</td>
<td>144</td>
<td>196.86</td>
<td>0</td>
<td>3.421</td>
<td>16.87</td>
<td></td>
</tr>
<tr>
<td>MPI_Allgatherv()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Bcast()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Comm_create()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_File_close()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_File_open()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_File_write_all()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_File_write_at()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Finalize()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Gather()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Gatherv()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Peak MPI-IO Write Bandwidth
# How Much Time is spent in Collectives?

<table>
<thead>
<tr>
<th>Name</th>
<th>Total</th>
<th>Num...</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MPI_Wait()</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>MPI_Waitall()</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message size for all-gather</td>
<td>305,753,268</td>
<td>72</td>
<td>172,215,296</td>
<td>4</td>
<td>4,246,573.167</td>
<td>22,551,605.859</td>
</tr>
<tr>
<td>Message size for all-reduce</td>
<td>163,308</td>
<td>632</td>
<td>21,908</td>
<td>4</td>
<td>258.399</td>
<td>897.725</td>
</tr>
<tr>
<td>Message size for all-to-all</td>
<td>112</td>
<td>14</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Message size for broadcast</td>
<td>692,208,045.5</td>
<td>3,346</td>
<td>18,117,620</td>
<td>0</td>
<td>206,876.284</td>
<td>1,284,673.036</td>
</tr>
<tr>
<td>Message size for gather</td>
<td>6,901,452,378</td>
<td>15,312</td>
<td>1,387,306,625</td>
<td>4</td>
<td>450,707.094</td>
<td>483,216.499</td>
</tr>
<tr>
<td>Message size for reduce</td>
<td>66,812</td>
<td>1,520</td>
<td>56</td>
<td>4</td>
<td>43.955</td>
<td>21.598</td>
</tr>
<tr>
<td>Message size for scatter</td>
<td>63,147,906</td>
<td>146</td>
<td>62,567,906</td>
<td>4</td>
<td>432.52</td>
<td>5,160.063</td>
</tr>
</tbody>
</table>

**Message sizes**

**Time spent in collectives**
3D Profile Visualization
3D Communication Visualization
3D Topology Visualization
How Does Each Routine Scale?
How Does Each Routine Scale?
Which Events Correlate with Runtime?
When do events occur?
Different Nodes, Different Timelines
Zoom In to View Individual Communication API Calls
Where do Events Occur?

- Callsites differentiate calls to common functions
- Separate time spent in API routines that are called throughout the code.
What Caused My Application to Crash?

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>py-c++-f90-create.ppkn</td>
</tr>
<tr>
<td>Application ID</td>
<td>0</td>
</tr>
<tr>
<td>Experiment ID</td>
<td>0</td>
</tr>
<tr>
<td>Time ID</td>
<td>0</td>
</tr>
</tbody>
</table>

Backtrace:
1. [SAMINT::timemaps::double, double] [mnt/home/jliford/py-c++-f90-create/SAMINT C:77] [mnt/home/jliford/py-c++-f90-create/samint.so]
2. [SAMINT::timemaps::double, double] [mnt/home/jliford/py-c++-f90-create/samint.so]
3. [wrap_samintStep] [mnt/home/jliford/py-c++-f90-create/samint_wrap.c:1383] [mnt/home/jliford/py-c++-f90-create/samint.so]
4. [call function] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4031] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
5. [fast_function] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4099] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
6. [PyEval_EvalCodeEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:692] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
7. [PyEval_EvalCode] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:667] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
8. [PyImport_GenerateModuleEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:671] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
9. [filesource::module] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:2137] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
10. [import_submodule] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:2584] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
11. [filesource::execute] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:2446] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
12. [import_module_level] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:2333] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
13. [builtin::import] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:489] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
14. [PyObject_Call] [mnt/home/jliford/0.55/build/Python-2.7.2/Objects/abstract.c:2529] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
15. [PyEval_EvalCodeWithKeywords] [mnt/home/jliford/0.55/build/Python-2.7.2/Objects/cwkw.c:3882] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
16. [PyEval_EvalFrameEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:2333] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
17. [fast_function] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4099] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
18. [PyEval_EvalCodeEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4100] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
19. [PyEval_EvalCode] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4100] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
20. [run_module] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:1386] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
21. [exec::statement] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4746] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
22. [PyEval_EvalCodeEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4100] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
23. [fast_function] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4099] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
24. [PyEval_EvalCodeEx] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4100] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]
25. [fast_function] [mnt/home/jliford/0.55/build/Python-2.7.2/Python/cvack-c:4099] [mnt/cvs/pycks/TOOL/0.65/pkgs/python-0.55/packages/Python-2.7.2/lib/libpython2.7.so]

Argonne National Laboratory

50 ATPESC 2017, July 30 – August 11, 2017
What Caused My Application to Crash?

Right-click to see source code
What Caused My Application to Crash?

```c
/*
 * Take a timestep - advance solution from "time" to "time + dt"
 */

void SAMINT::timestep(const double time,
                       const double dt)
{
    cout << "SAMINT::timestep()" << endl;
    timestep_(time, dt);
    int x = 4 / (4-4);
    cout << "x = " << x << endl;
}

/*
 * Write data to outflow
 * (visit, fieldview, or overgrid - set in samarc input file)
 */

void SAMINT::writePlotData(const double time,
                           const int step)
{
    cout << "SAMINT::writePlotData()" << endl;
}
```
Conclusion
Downloads

www.taucommander.com

www.github.com/ParaToolsInc/taucmdr

Free, open source, BSD license
Acknowledgements

• Engility
• HPCMP DoD PETTT Program
• Department of Energy
  – Office of Science
  – Argonne National Laboratory
  – Oak Ridge National Laboratory
  – NNSA/ASC Trilabs (SNL, LLNL, LANL)
• National Science Foundation
• University of Tennessee
• University of New Hampshire
• University of Oregon
• TU Dresden
• Research Centre Jülich