

# ParaTools

---

Drs. Allen Malony, Sameer Shende, John C. Linford, et al.  
{malony, sameer, jlinford}@paratools.com

2 February 2017  
Northrop Grumman  
Woodlawn, MD

# Contact Info

Dr. John Linford: 540-808-9250

Phone: 541-913-8797

Fax: 541-343-6086

[info@paratools.com](mailto:info@paratools.com)



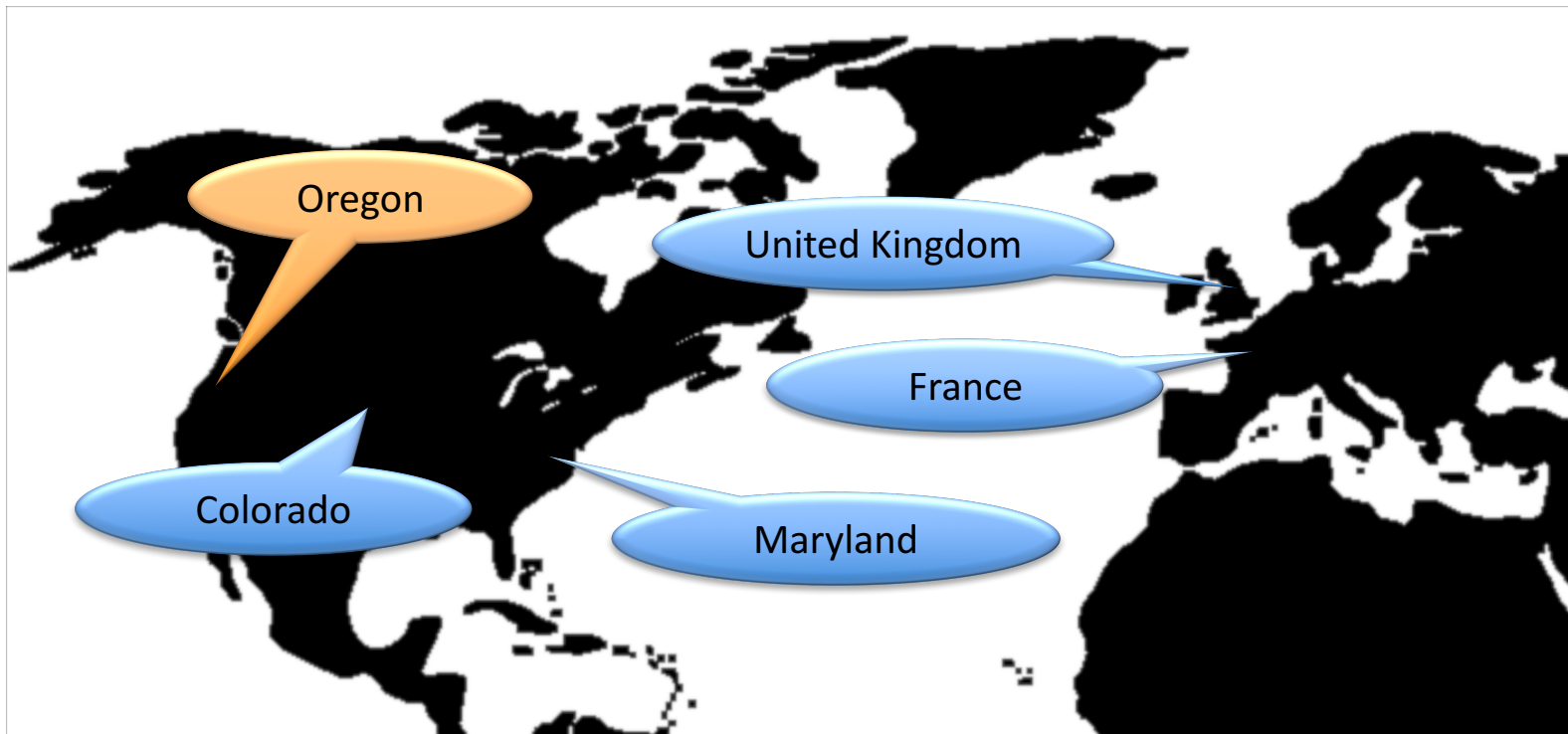
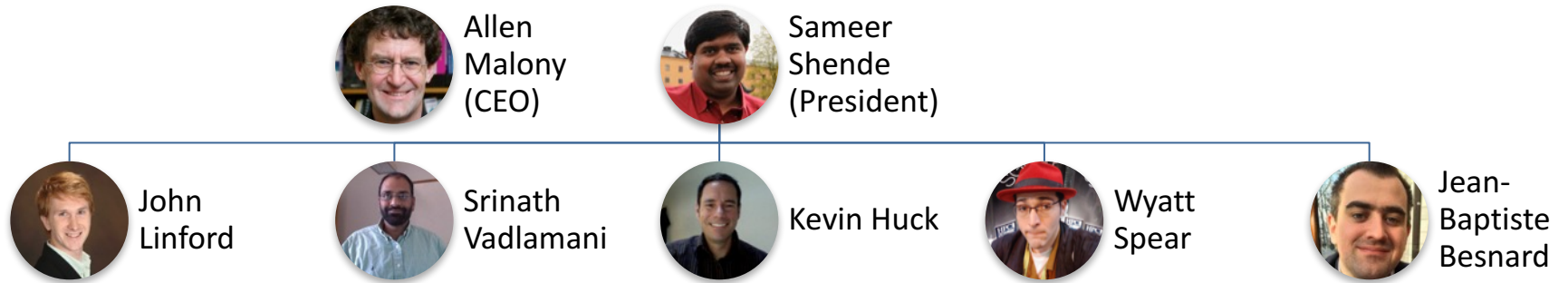
**Today's Goal:  
Make Contact!**

ParaTools, Inc.  
2836 Kincaid St.  
Eugene, OR 97405, USA

# Outline

- Introduction to ParaTools Inc.
- The TAU Performance System<sup>®</sup>
- ParaTools ThreadSpotter
- Success Stories
- Q&A

# ParaTools



# ParaTools Accelerates Software



# In the Press



## Rogue Wave Reveals New Releases and a Presentation Theater at SC14

BOULDER, CO -- (Marketwired) -- 11/14/14 -- [Rogue Wave Software](#), the largest independent provider of cross-platform software development tools and embedded components, will be highlighting their products at [Supercomputer 2014](#) from November 17-20 at Ernest N. Morial Convention Center in New Orleans. Rogue Wave will have new and updated product offerings, as well as several chances to educate, explore, and learn at this year's conference.



With recent releases of both [IMSL Fortran Numerical Library 7.1](#) and [Visualization for C++ 5.8](#), attendees can expect to learn about new performance improvements, code fixes, and updates to supported platforms. In addition to the C++ product release, Rogue Wave will also share the accompaniments made to the [Visualization product set](#). Adding [IViews](#) and [Elixir](#) expands the Visualization toolset for C++, Java, and Adobe Flex.

Also at this conference, Rogue Wave will join [ParaTools](#) in releasing the [source code for ThreadSpotter](#). A memory optimization tool that analyzes cache memory and thread communication in single and multicore systems, ThreadSpotter pinpoints performance issues and provides specific guidance on how to correct them, increasing program performance and improving developer productivity.

- <http://tau.uoregon.edu/success/irmhd.pdf>
- <http://www.defenseindustrydaily.com/Up-to-1471M-to-HPTi-for-DoD-High-Performance-Computing-Work-05688/>
- <http://java.sys-con.com/node/3228954>
- <http://www.bizjournals.com/baltimore/blog/cyberbizblog/2014/02/umbc-tech-park-lures-oregon-software.html>



## UMBC tech park lures Oregon software company

Feb 28, 2014, 1:33pm EST | **UPDATED:** Jun 9, 2014, 9:12am EDT



**Sarah Gantz**  
Reporter - Baltimore Business Journal  
Email | Twitter

A West Coast software company is expanding to Baltimore to tap into the area's growing cyber security industry.

ParaTools, based in Eugene, Ore., has opened a Baltimore office at the [University of Maryland](#), Baltimore County's [bwtech@UMBC](#) research and technology park. The company's software was designed to make complex

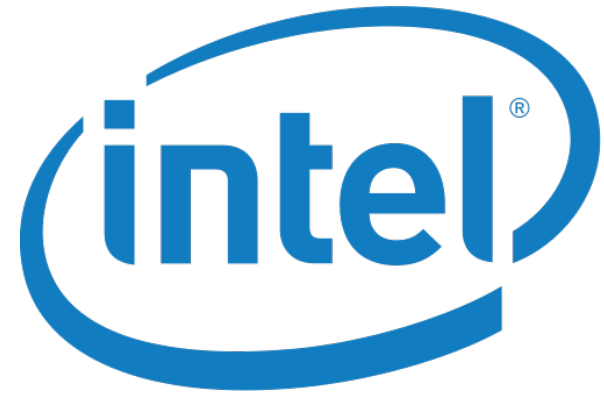
supercomputers run more efficiently. Now, ParaTools wants to use the same efficiency software to improve cyber operations.

"With our skills in optimizing software and computer performance we think there's a lot of opportunity in the cyber," said [John Linford](#), who is heading up the company's Baltimore office.

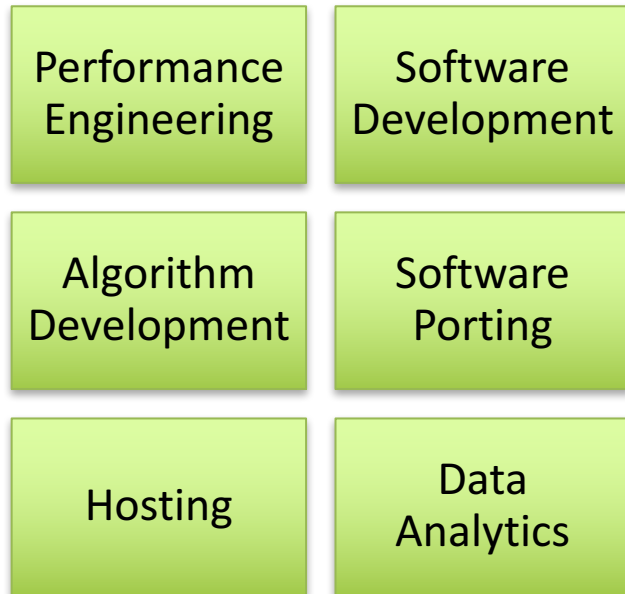
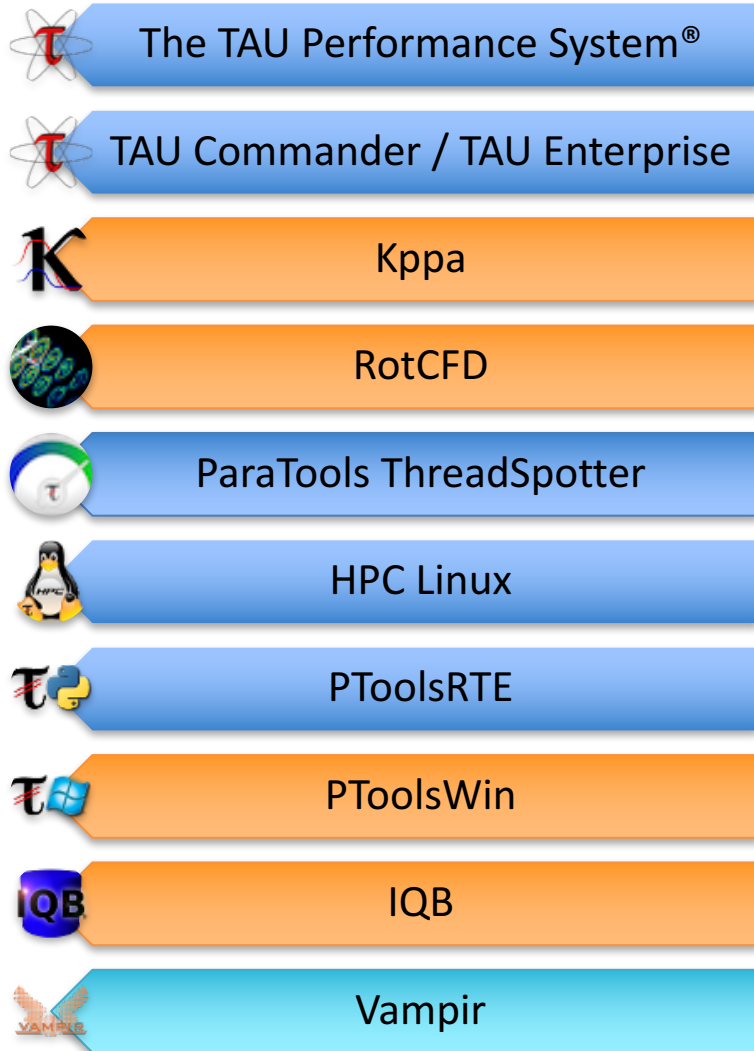
ParaTools' software attaches itself to a system and creates a profile that shows how and where time is being spent. The information can be used to rewrite code in trouble spot and make the system run more smoothly.

A possible cyber use for the company's technology is improving efficiency of drone aircraft that scans for suspicious vehicles. Previously, the drones collected data and brought it home to be analyzed. With ParaTools' technology, could send data real-time, so they can more quickly address any problems identified.

# Partnerships



# Products and Services



*ParaTools*® and *IQB*® are registered trademarks of ParaTools, Inc.

ParaTools is the sole licensee of the *TAU Performance System*® trademark owned by the State of Oregon acting by and through the State Board of Higher Education on behalf of the University of Oregon.



# Products and Services

 The TAU Performance System®


 TAU Commander / TAU Enterprise

 Kppa


 RotCFD

 ParaTools ThreadSpotter

 HPC Linux

 PToolsRTE

 PToolsWin

 IQB

 Vampir

Performance  
Engineering

Software  
Development

Algorithm  
Development

Software  
Porting

Hosting

Data  
Analytics

*ParaTools*® and *IQB*® are registered trademarks of ParaTools, Inc.

ParaTools is the sole licensee of the *TAU Performance System*® trademark owned by the State of Oregon acting by and through the State Board of Higher Education on behalf of the University of Oregon.

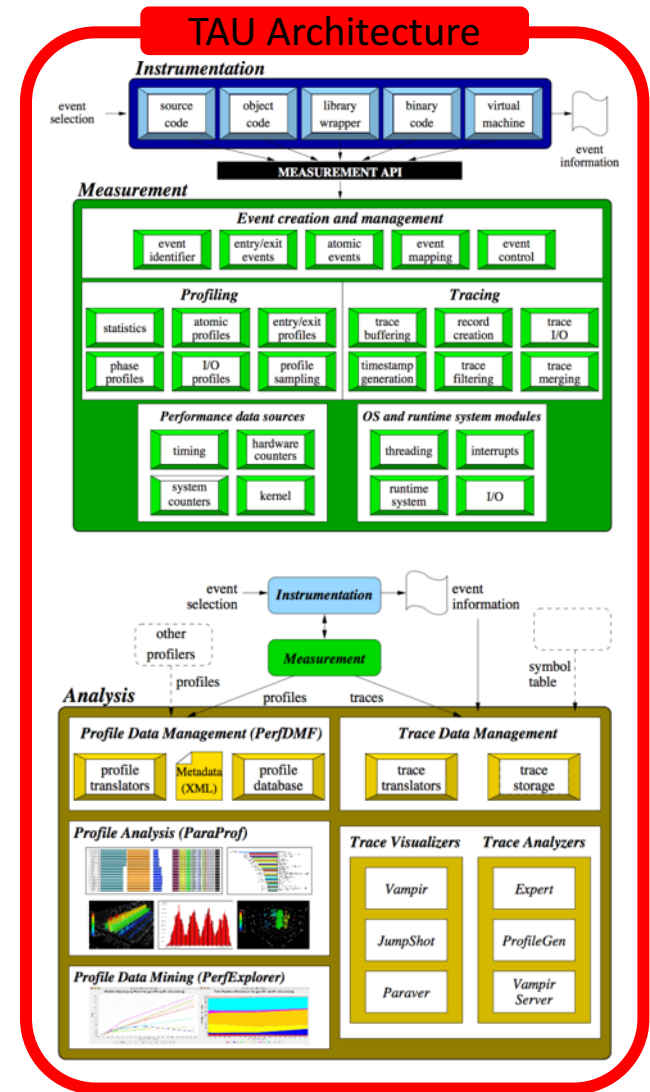
Keeping Up With Technology

---

# THE TAU PERFORMANCE SYSTEM

# The TAU Performance System®

- Toolkit for performance problem solving and software situational awareness
  - Instrumentation, measurement, analysis, visualization
  - Portable profiling and tracing
  - Performance data management and data mining
- Free, open source, BSD license
- <http://tau.uoregon.edu/>



# Questions TAU Can Answer

- **How much time** is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*?
- **How many instructions** are executed in these code regions? Floating point, Level 1 and 2 *data cache misses*, hits, branches taken, *vector instructions*?
- What is the **memory usage** of the code? When and where is memory allocated/de-allocated? Are there any *memory leaks*?
- What are the **I/O characteristics** of the code? What is the peak read and write *bandwidth* of individual calls, total volume?
- What is the **time spent waiting for collectives** (e.g. reduce)?
- How does the application **scale**?

# Use Cases

- Code Modernization and Hardening
  - CREATE-AV HELIOS / KESTREL
    - Army, Navy, Air Force, NASA, Boeing...
  - FraPPE: Framework for Parallel Program Engineering
    - Army, NASA
- Performance Provenance
  - Protection (like ClearCase for performance)
- Performance Optimization
  - Load balancing
  - “Hot spot” identification
  - Improvement quantification
- Performance Regression Testing

# Apples to Apples Across Platforms

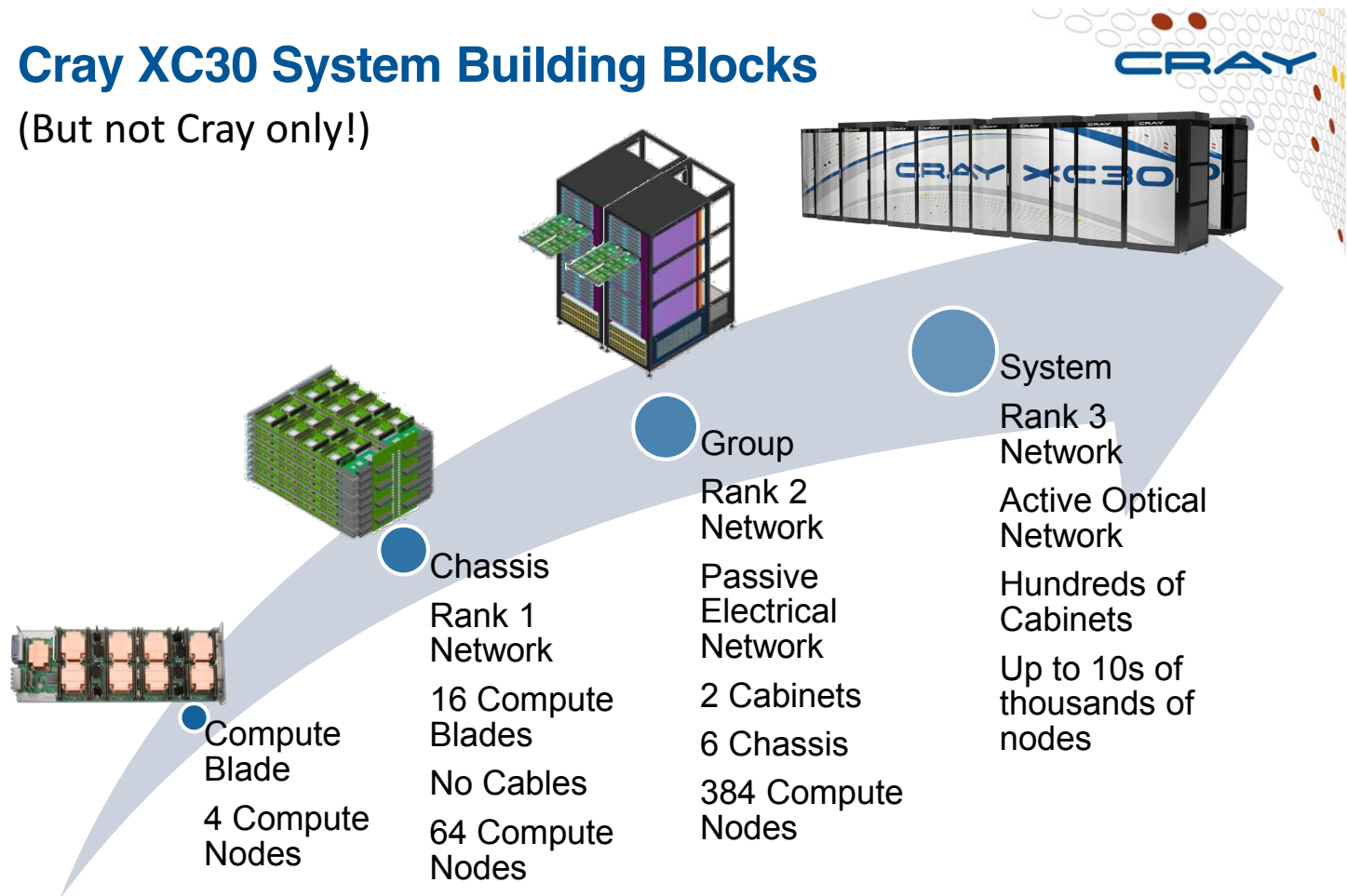
C/C++      CUDA      UPC      Python  
Fortran      OpenACC      GPI      MPI  
pthreads      Intel MIC      Java      OpenMP  
Intel      GNU      Cray      Sun  
MinGW      LLVM      PGI      AIX  
Linux      Windows  
BlueGene      Fujitsu      ARM  
Android      MPC      OS X

Insert  
yours  
here

# Measurement at Every Layer

## Cray XC30 System Building Blocks

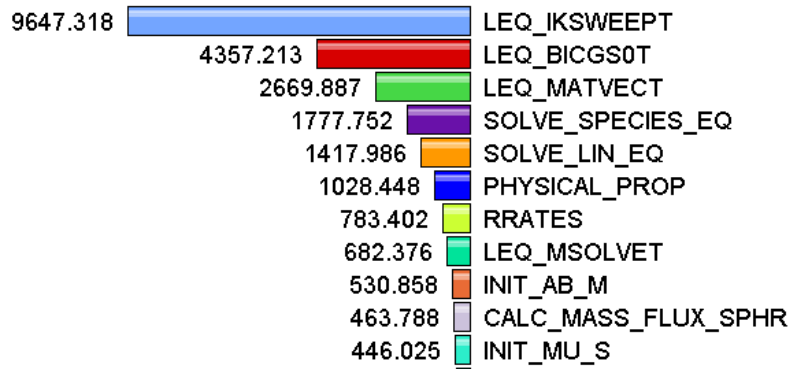
(But not Cray only!)



Copyright © Cray Inc.

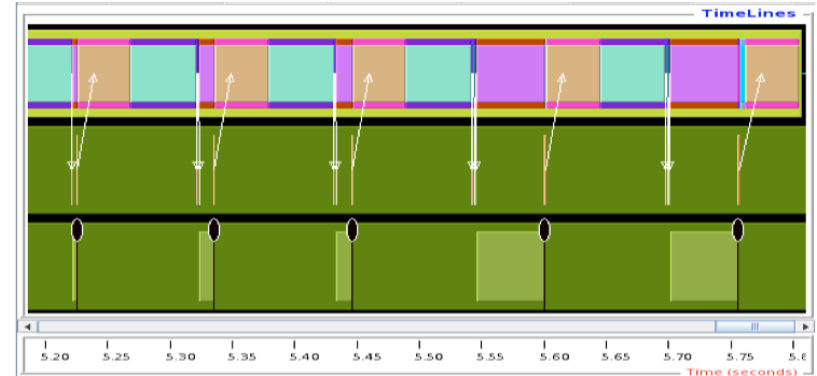
# Measurement Approaches

## Profiling



Shows  
**how much** time  
was spent in each  
routine

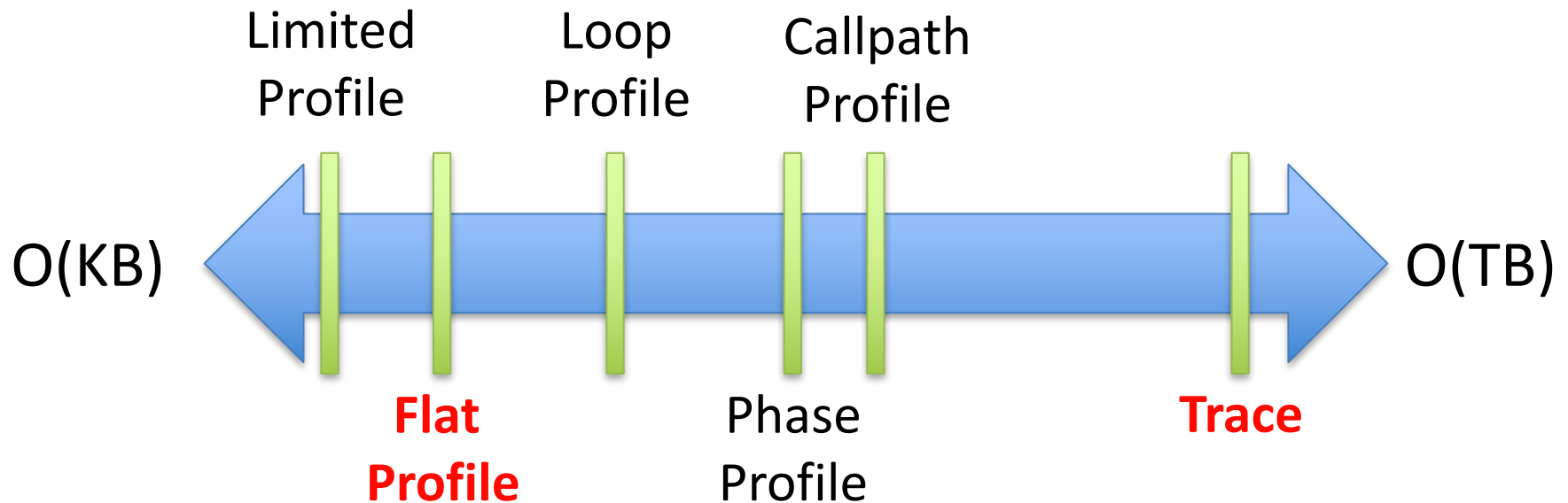
## Tracing



Shows  
**when** events  
take place on a  
timeline



# How Much Data do you Want?



All levels support multiple metrics/counters

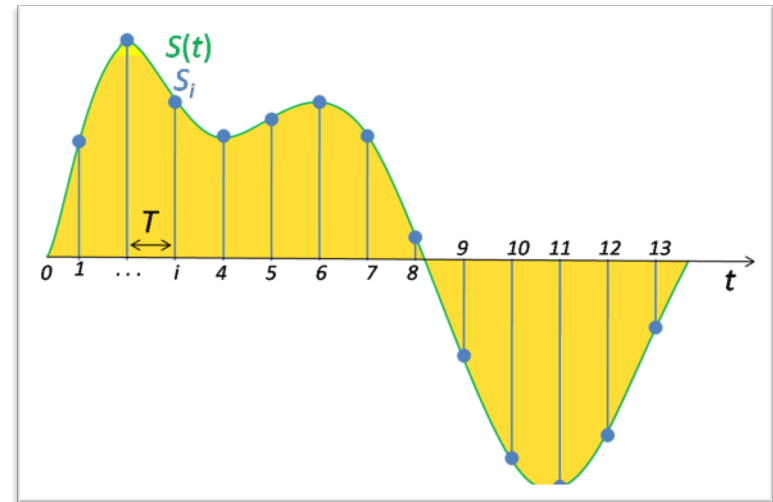
# Performance Data Measurement

## Direct via Probes

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

- Exact measurement
- Fine-grain control
- Code region granularity
- Calls inserted into code

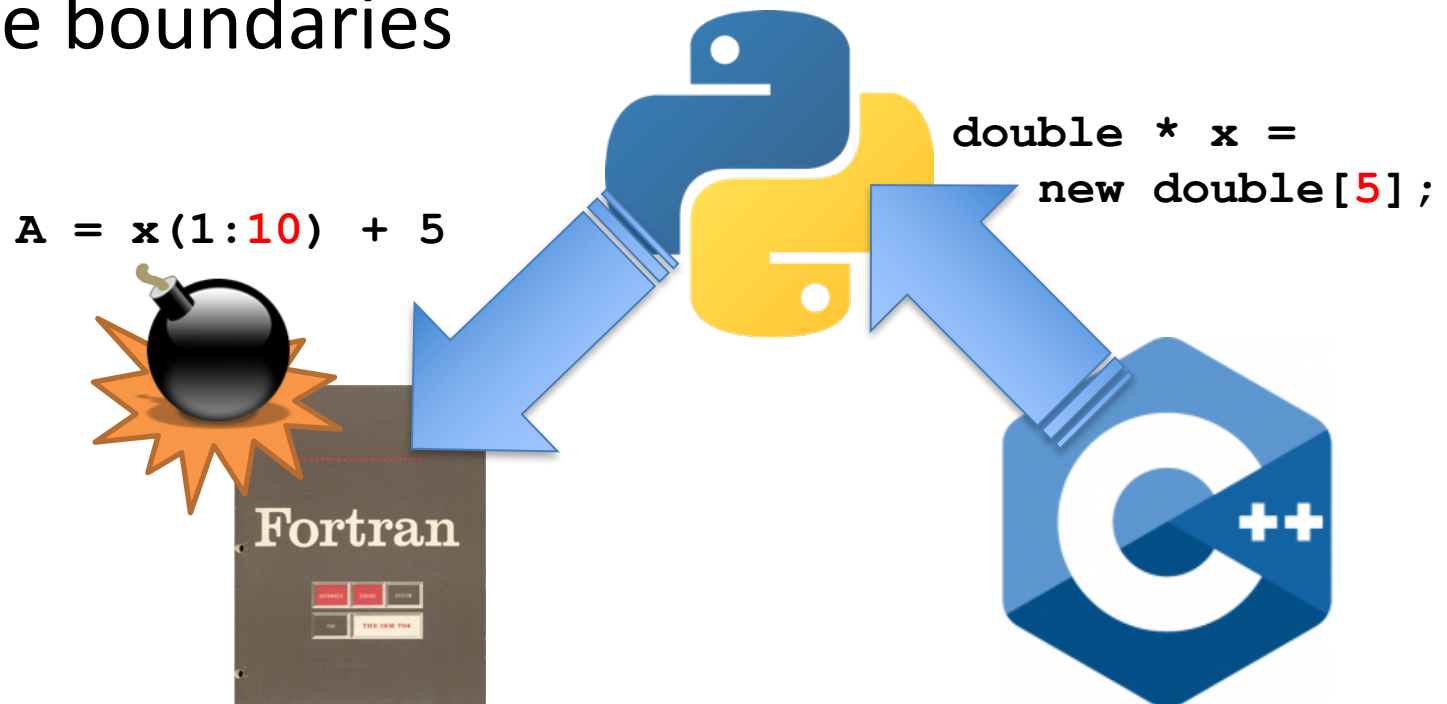
## Indirect via Sampling



- No code modification
- Minimal effort
- Code line granularity
- Uses debug symbols

# Multi-Language Debugging

- Identify the source location of a crash by unwinding the system callstack
- Identify memory errors (off-by-one, etc.) across language boundaries

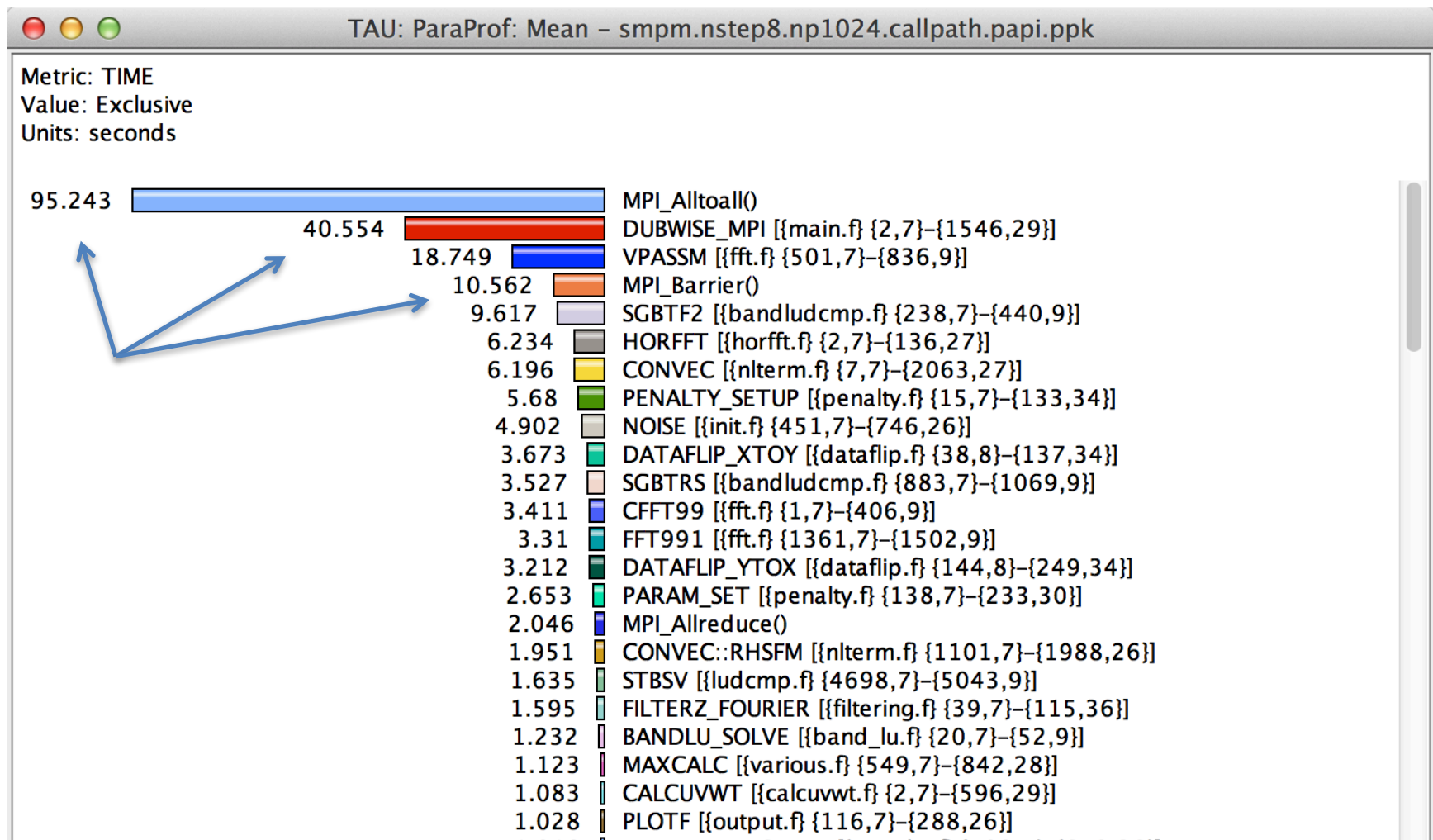


Keeping Up With Technology

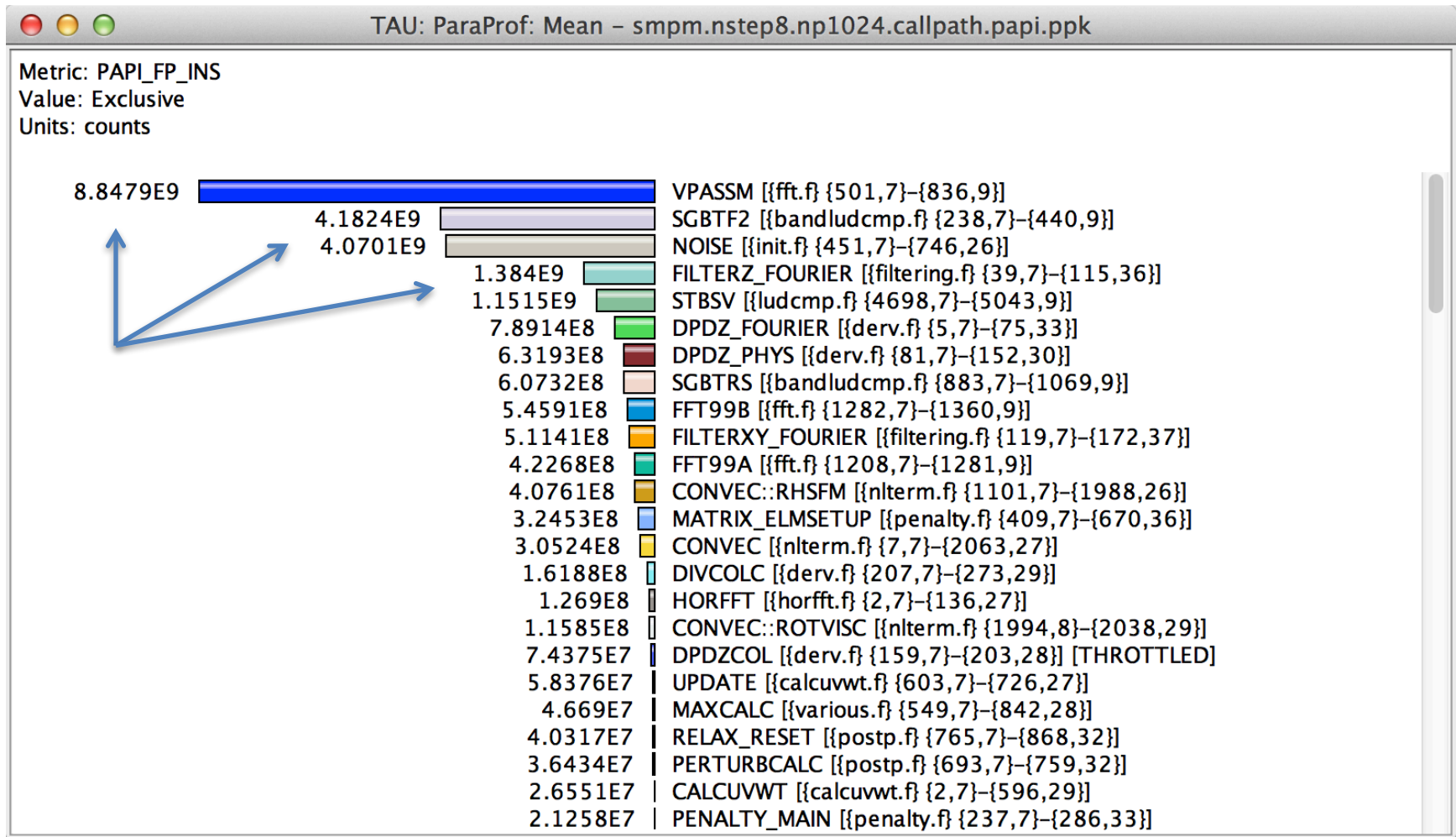
---

# DATA ANALYSIS WITH TAU

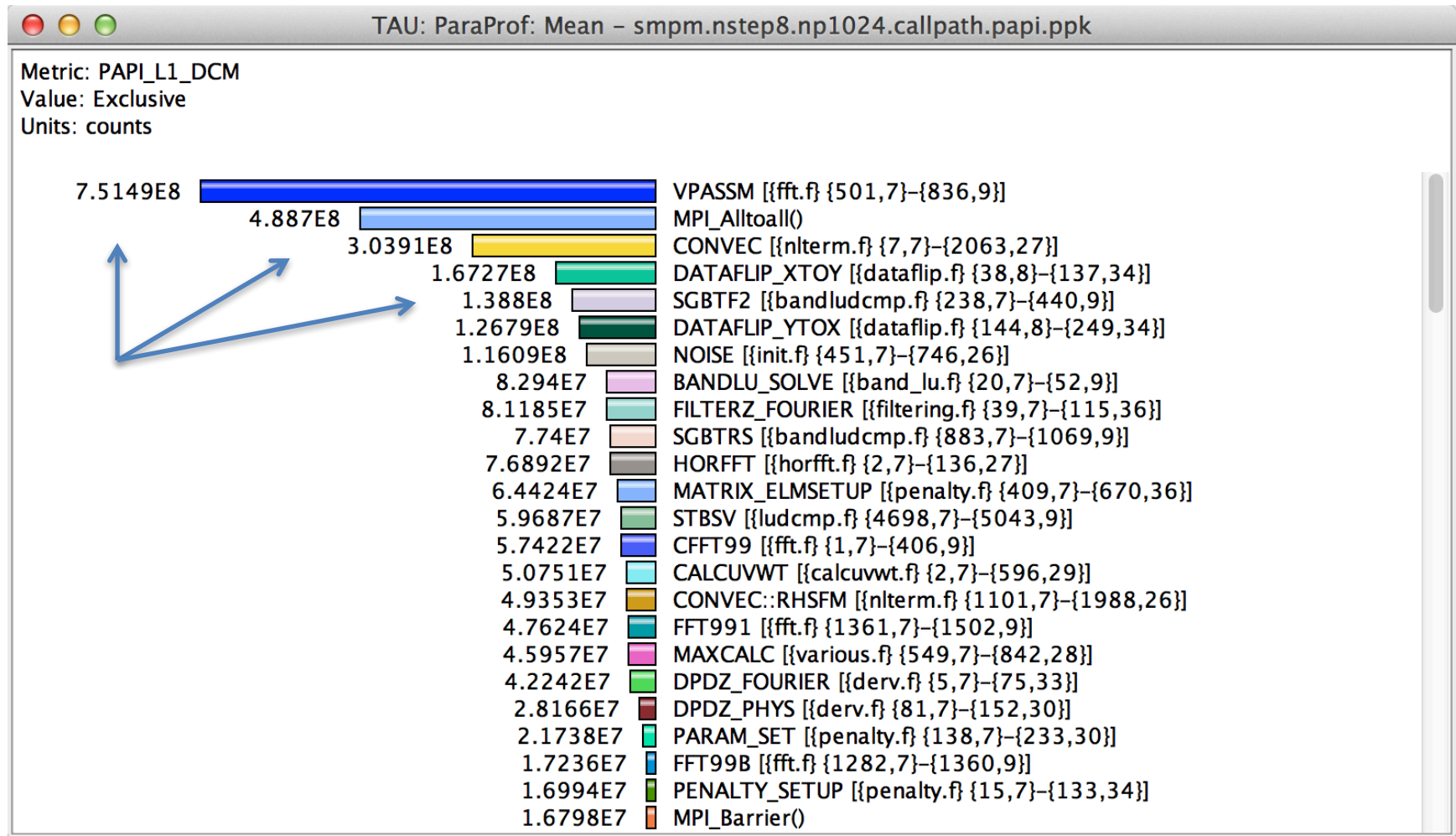
# How Much Time per Code Region?



# How Many Instructions per Code Region?



# How Many L1/L2/L3 Cache Misses?



# How Much Memory Does the Code Use?

Name $\Delta$	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
▼ TAU application						
free size (bytes)	14,236,992.16	27,169.781	49,152	1	524.001	2,013.103
malloc size (bytes)	13,132,932	23,292	262,144	1	563.839	4,492.057
▶ MPI_Finalize()						
▼ OurMain()						
free size (bytes)	1,298,918.679	1,495.125	461,766.25	4	868.769	16,928.073
malloc size (bytes)	48,150	20	36,032	11	2,407.5	7,911.992
▼ OurMain						
free size (bytes)	3,465	9	769	32	385	260.2
malloc size (bytes)	4,314	12	769	32	359.5	240.981
▼ <module>						
free size (bytes)	293,088	449	32,564	32	652.757	1,526.875
malloc size (bytes)	311,966	493	32,564	32	632.791	1,460.941
▶ staticCFD						
▶ __init__						
▶ <module>						
Memory Utilization (heap, in KB)		849,270.344	192,825.168	0.078	147,832.141	62,621.576
Message size for all-gather	4,096	1	4,096	4,096	4,096	0
Message size for all-reduce	23,340	843	320	4	27.687	64.653
Message size for all-to-all	104	26	4	4	4	0
Message size for broadcast	24,923	206	8,788	4	120.985	860.992
Message size for reduce	8,912	8	8,788	4	1,114	2,900.511
free size (bytes)	27,417,881,391.51	413,600.719	24,025,667	1	66,290.701	199,538.234
malloc size (bytes)	27,468,709,355.914	435,669.625	24,025,667	0	63,049.402	195,561.193

High-water mark





# How Much Memory Does the Code Use?

TAU: ParaProf: Mean Context Events – sphere\_np32\_nsteps5\_mem.ppk

Name $\Delta$	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
▼ TAU application						
free size (bytes)	14,236,992.16	27,169.781	49,152	1	524.001	2,013.103
malloc size (bytes)	13,132,932	23,292	262,144	1	563.839	4,492.057
▶ MPI_Finalize()						
▼ OurMain()						
free size (bytes)	1,298,918.679	1,495.125	461,766.25	4	868.769	16,928.073
malloc size (bytes)	48,150	20	36,032	11	2,407.5	7,911.992
▼ OurMain						
free size (bytes)	3,465	9	769	32	385	260.2
malloc size (bytes)	4,314	12	769	32	359.5	240.981
▼ <module>						
free size (bytes)	293,088	449	32,564	32	652.757	1,526.875
malloc size (bytes)	311,966	493	32,564	32	632.791	1,460.941
▶ staticCFD						
▶ __init__						
▶ <module>						
Memory Utilization (heap, in KB)		849,270.344	192,825.168	0.078	147,832.141	62,621.576
Message size for all-gather	4,096	1	4,096	4,096	4,096	0
Message size for all-reduce	23,340	843	320	4	27.687	64.653
Message size for all-to-all	104	26	4	4	4	0
Message size for broadcast	24,923	206	8,788	4	120.985	860.992
Message size for reduce	8,912	8	8,788	4	1,114	2,900.511
free size (bytes)	27,417,881,391.51	413,600.719	24,025,667	1	66,290.701	199,538.234
malloc size (bytes)	27,468,709,355.914	435,669.625	24,025,667	0	63,049.402	195,561.193

Total allocated/deallocated

# Where is Memory Allocated / Deallocated?

Name $\Delta$	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
▼ TAU application						
free size (bytes)	14,236,992.16	27,169.781	49,152	1	524.001	2,013.103
malloc size (bytes)	13,132,932	23,292	262,144	1	563.839	4,492.057
▶ MPI_Finalize()						
▼ OurMain()						
free size (bytes)	1,298,918.679	1,495.125	461,766.25	4	868.769	16,928.073
malloc size (bytes)	48,150	20	36,032	11	2,407.5	7,911.992
▼ OurMain						
free size (bytes)	3,465	9	769	32	385	260.2
malloc size (bytes)	4,314	12	769	32	359.5	240.981
▼ <module>						
free size (bytes)	293,088	449	32,564	32	652.757	1,526.875
malloc size (bytes)	311,966	493	32,564	32	632.791	1,460.941
▶ staticCFD						
▶ __init__						
▶ <module>						
Memory Utilization (heap, in KB)		849,270.344	192,825.168	0.078	147,832.141	62,621.576
Message size for all-gather	4,096	1	4,096	4,096	4,096	0
Message size for all-reduce	23,340	843	320	4	27.687	64.653
Message size for all-to-all	104	26	4	4	4	0
Message size for broadcast	24,923	206	8,788	4	120.985	860.992
Message size for reduce	8,912	8	8,788	4	1,114	2,900.511
free size (bytes)	27,417,881,391.51	413,600.719	24,025,667	1	66,290.701	199,538.234
malloc size (bytes)	27,468,709,355.914	435,669.625	24,025,667	0	63,049.402	195,561.193

Allocation / Deallocation Events

# What are the I/O Characteristics?

TAU: ParaProf: Context Events for thread: n,c,t, 1,0,0 - samarc\_obe\_4p\_iomem\_cp.ppk

Name ▾	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
▼ .TAU application						
▶ read()						
▶ fopen64()						
▶ fclose()						
▼ OurMain()						
malloc size	25,235	1,097.174	23	11	12,032	2,851.143
free size	22,707	1,746.692	13	11	12,032	3,660.642
▼ OurMain [{{wrapper.py}}{3}]						
▶ read()						
malloc size	3,877	323.083	12	32	981	252.72
free size	1,536	219.429	7	32	464	148.122
▶ fopen64()						
▶ fclose()						
▼ <module> [{{obe.py}}{8}]						
▼ writeRestartData [{{samarcInterface.py}}{145}]						
▼ samarcWriteRestartData						
▼ write()						
WRITE Bandwidth (MB/s) <file="samarc/restore.00002/nodes.00004/proc.00001">		74.565	117	0	2,156.889	246.386
WRITE Bandwidth (MB/s) <file="samarc/restore.00001/nodes.00004/proc.00001">		77.594	117	0	1,941.2	228.366
WRITE Bandwidth (MB/s)		76.08	234	0	2,156.889	237.551
Bytes Written <file="samarc/restore.00002/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written <file="samarc/restore.00001/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written	4,195,104	17,927.795	234	1	1,048,576	133,362.946
▶ open64()						

Write bandwidth per file

Bytes written to each file

# What are the I/O Characteristics?

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

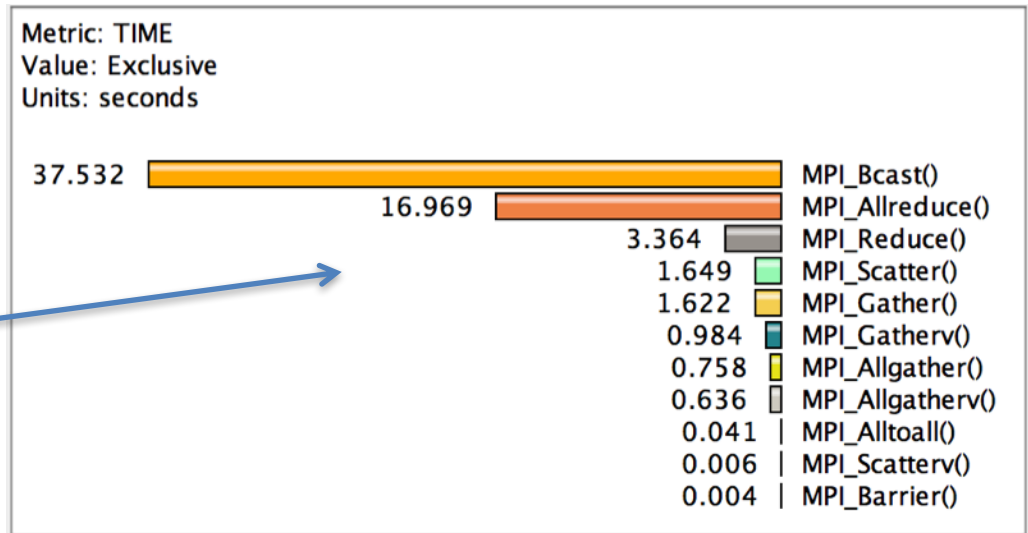
Peak Parallel I/O Write Bandwidth

# How Much Time is Spent in Collectives?

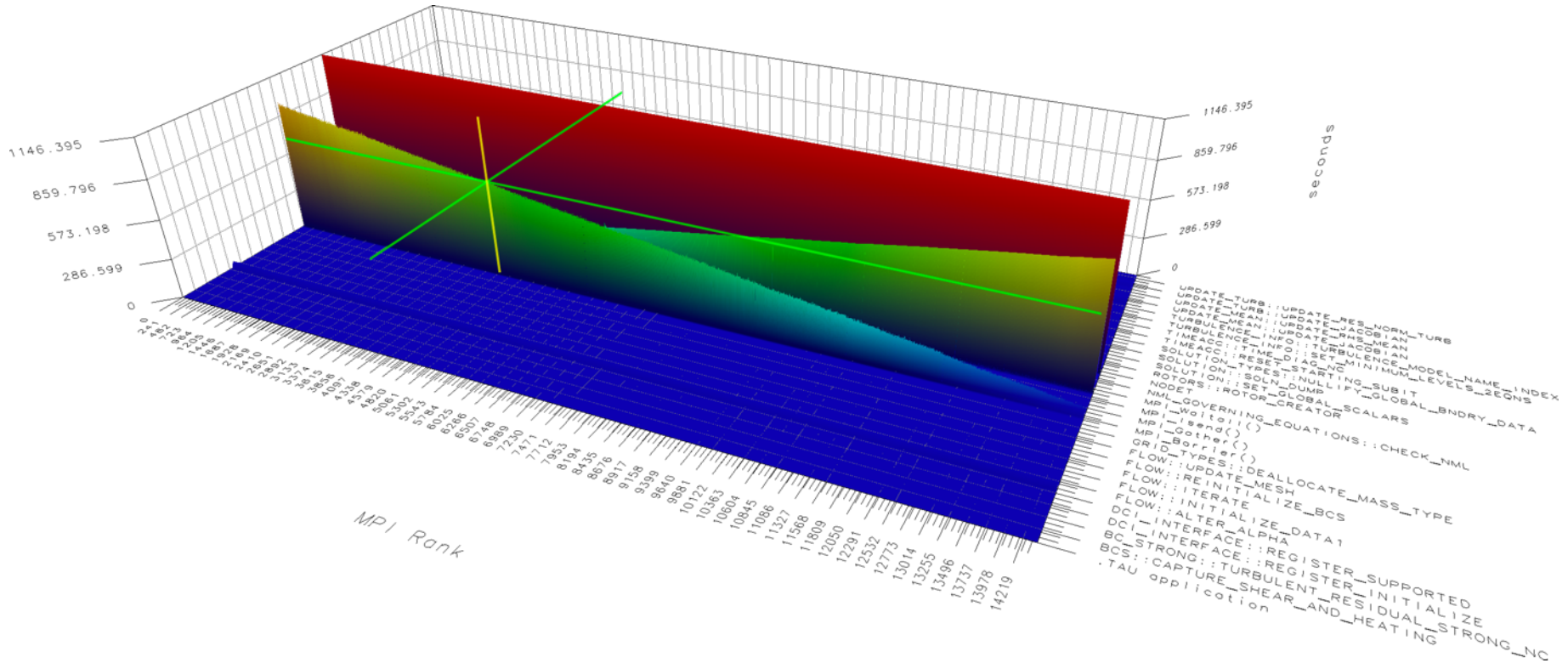
Name <span>△</span>	Total	Num...	MaxValue	MinValue	MeanValue	Std. Dev.
▶ MPI_Wait()						
▶ MPI_Waitall()						
Message size for all-gather	305,753,268	72	172,215,296	4	4,246,573.167	22,551,605.859
Message size for all-reduce	163,308	632	21,908	4	258.399	897.725
Message size for all-to-all	112	14	8	8	8	0
Message size for broadcast	692,208,045.5	3,346	18,117,620	0	206,876.284	1,284,673.036
Message size for gather	6,901,452.378	15.312	1,387,306.625	4	450,707.094	483,216.499
Message size for reduce	66,812	1,520	56	4	43.955	21.598
Message size for scatter	63,147.906	146	62,567.906	4	432.52	5,160.063

Message sizes

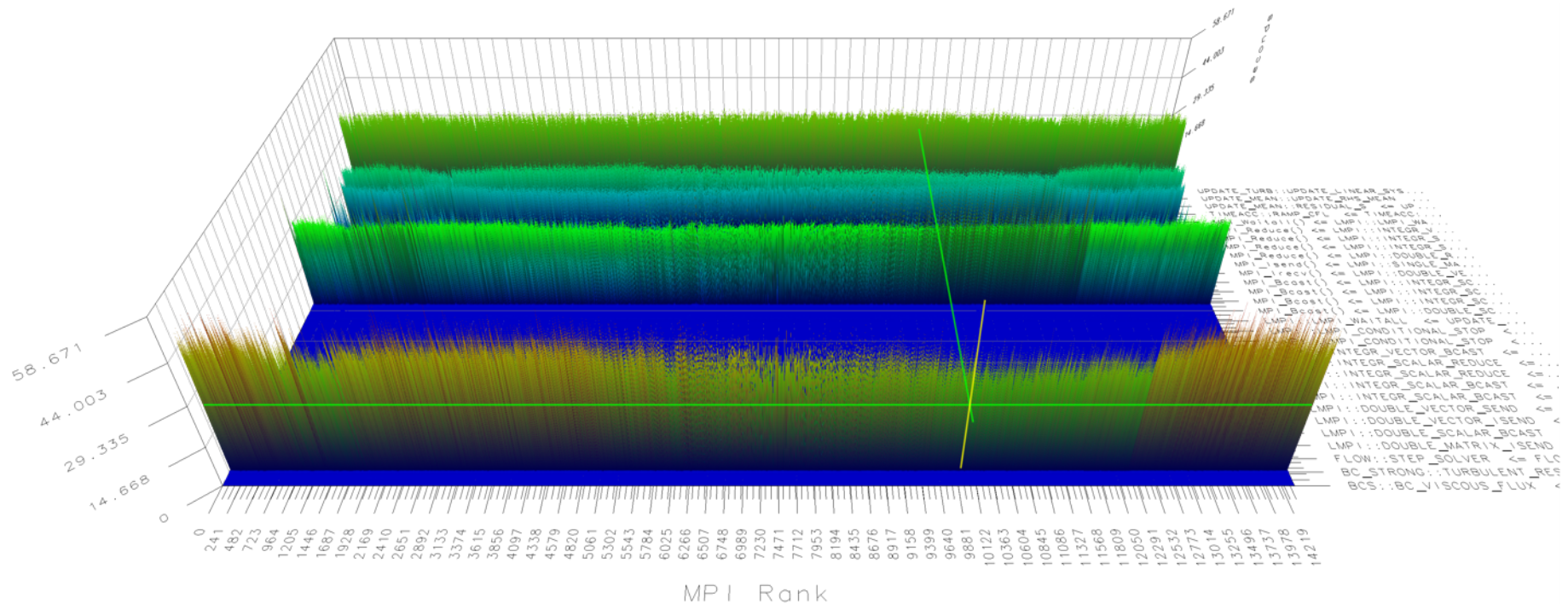
Time spent in collectives



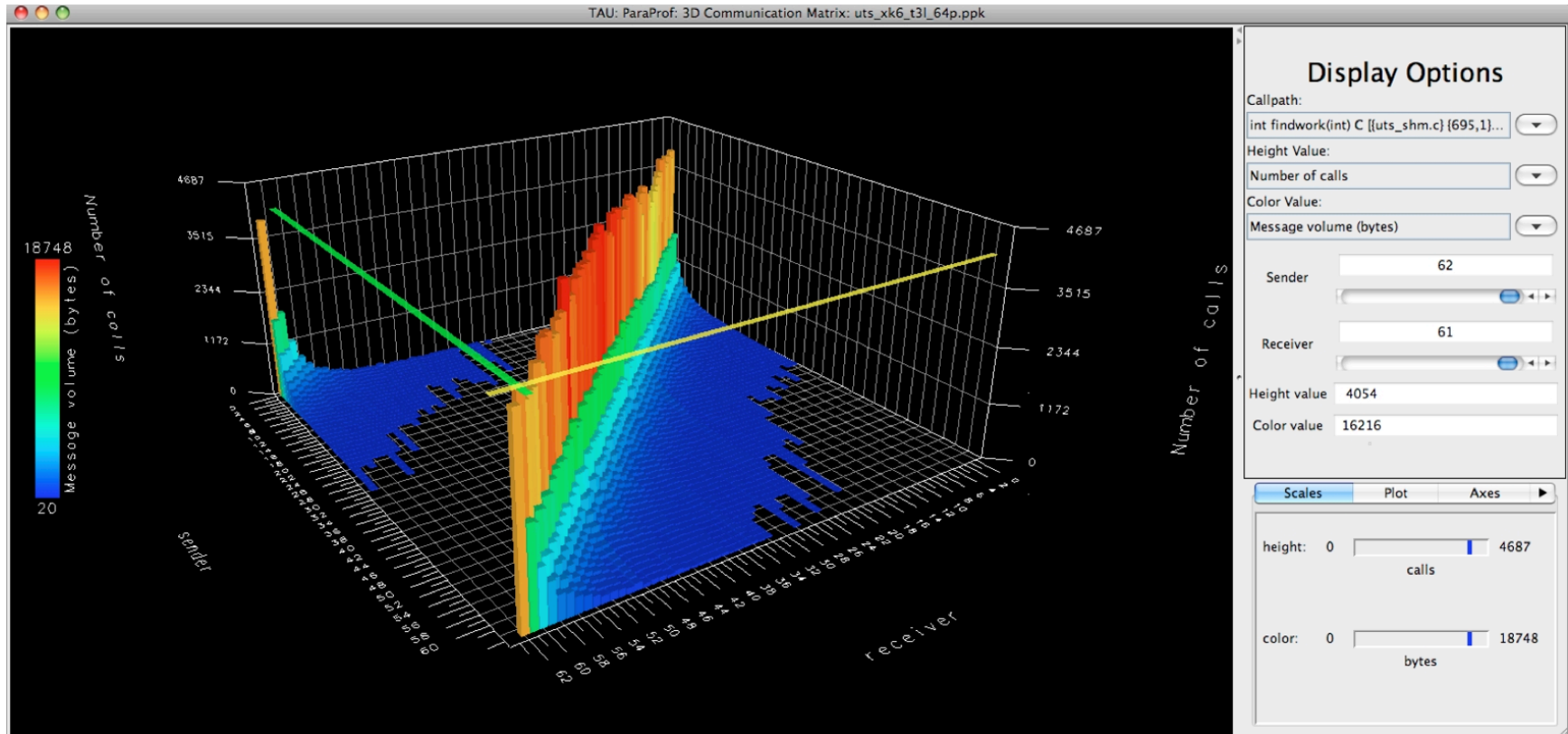
# Are there relationships or patterns?



# Are there load imbalances?

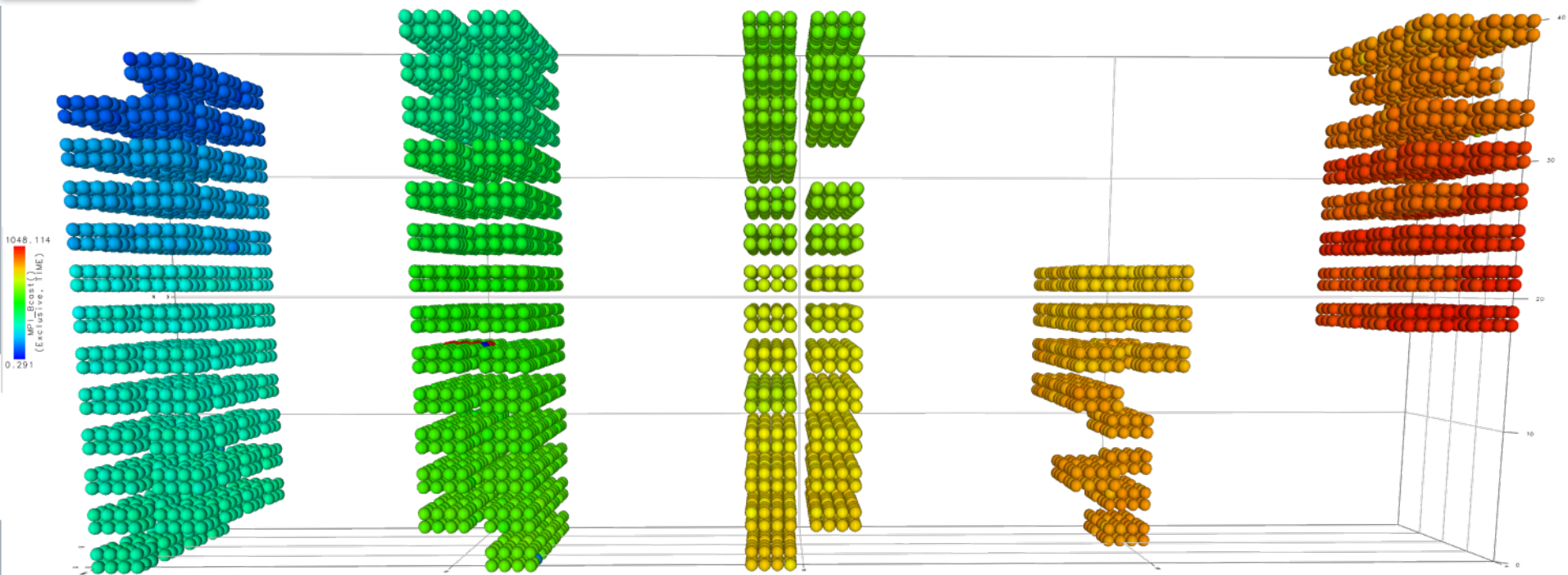


# What is the communication profile?



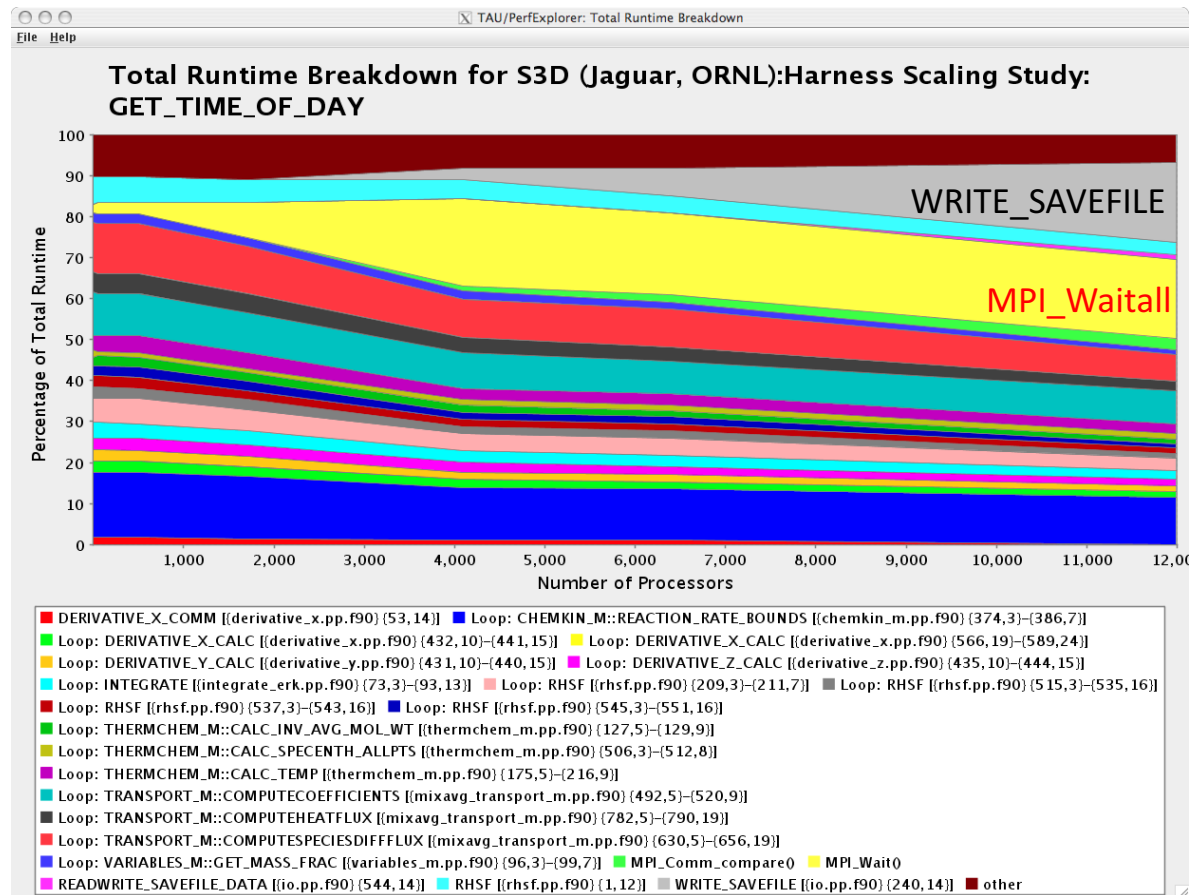


# Where is the computation performed?



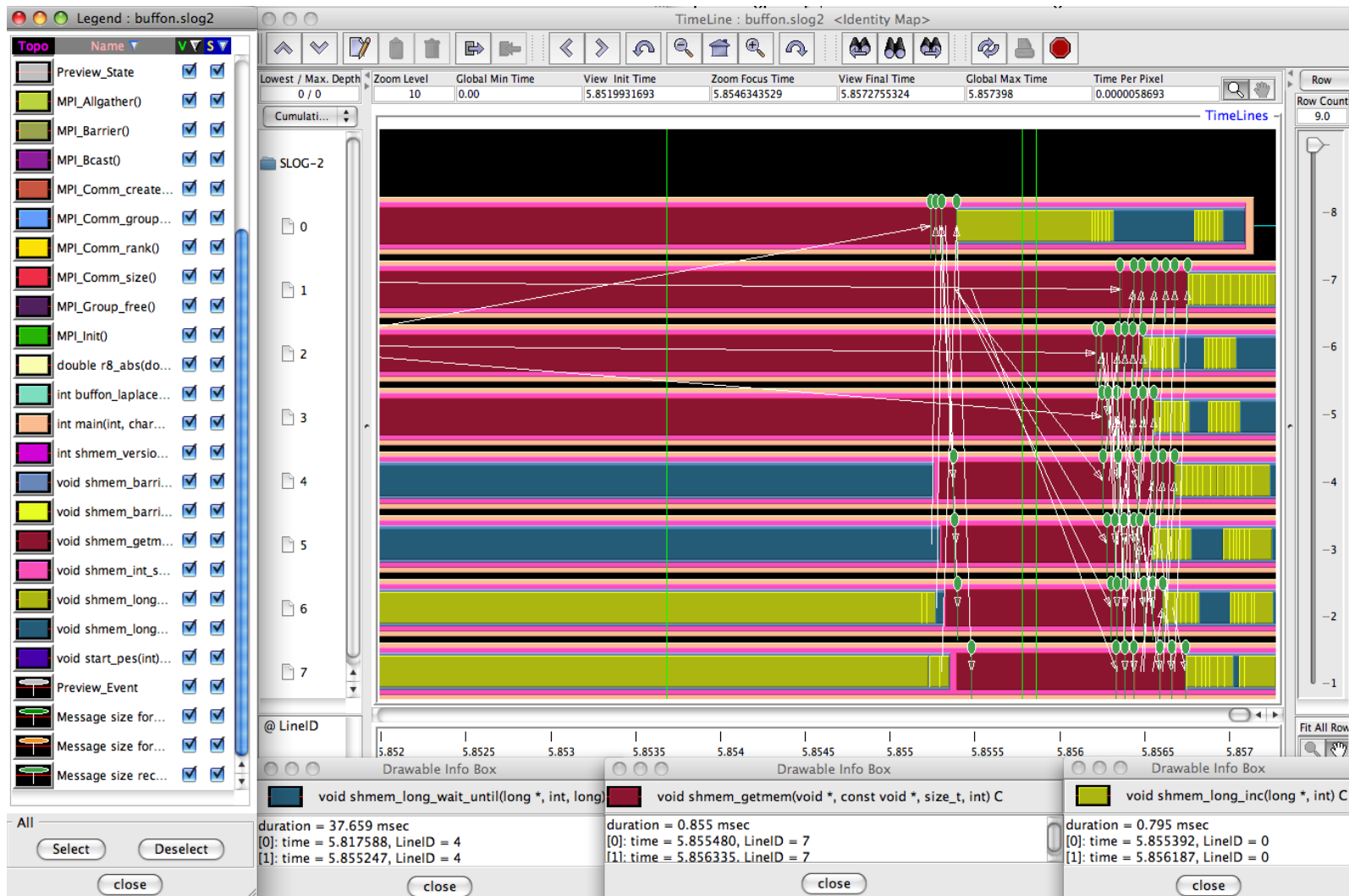
**Note:** Virtual, physical, and user-defined topologies are supported.

# How does the application scale?



% perfexplorer (Charts → Runtime Breakdown)

# When do Events Occur?



# What Caused My Application to Crash?

```
TAU: ParaProf Manager

Options Help

Applications
  Standard Applications
  Default App
  Default Exp
  py-c++-f90-create.ppk
  TIME

TrialField                                     Value
Name                                             py-c++-f90-create.ppk
Application ID                                   0
Experiment ID                                    0
Trial ID                                         0
BACKTRACE 1 [SAMINT::timestep(double, double)] [/mnt/home/jlinfo...
BACKTRACE 2 [samarcStep(double, double)] [/mnt/home/jlinfo...
BACKTRACE 3 [_wrap_samarcStep] [/mnt/home/jlinfo...
BACKTRACE 4 [call_function] [/mnt/home/jlinfo...
BACKTRACE 5 [fast_function] [/mnt/home/jlinfo...
BACKTRACE 6 [PyEval_EvalCodeEx] [/mnt/home/jlinfo...
BACKTRACE 7 [PyEval_EvalCode] [/mnt/home/jlinfo...
BACKTRACE 8 [PyImport_ExecCodeModuleEx] [/mnt/home/jlinfo...
BACKTRACE 9 [load_source_module] [/mnt/home/jlinfo...
BACKTRACE 10 [import_submodule] [/mnt/home/jlinfo...
BACKTRACE 11 [load_next] [/mnt/home/jlinfo...
BACKTRACE 12 [import_module_level] [/mnt/home/jlinfo...
BACKTRACE 13 [builtin___import_] [/mnt/home/jlinfo...
BACKTRACE 14 [PyObject_Call] [/mnt/home/jlinfo...
BACKTRACE 15 [PyEval_CallObjectWithKeywords] [/mnt/home/jlinfo...
BACKTRACE 16 [PyEval_EvalFrameEx] [/mnt/home/jlinfo...
BACKTRACE 17 [fast_function] [/mnt/home/jlinfo...
BACKTRACE 18 [PyEval_EvalCodeEx] [/mnt/home/jlinfo...
BACKTRACE 19 [PyEval_EvalCode] [/mnt/home/jlinfo...
BACKTRACE 20 [run_mod] [/mnt/home/jlinfo...
BACKTRACE 21 [exec_statement] [/mnt/home/jlinfo...
BACKTRACE 22 [PyEval_EvalCodeEx] [/mnt/home/jlinfo...
BACKTRACE 23 [fast_function] [/mnt/home/jlinfo...
BACKTRACE 24 [fast_function] [/mnt/home/jlinfo...
BACKTRACE 25 [PyEval_EvalCodeEx] [/mnt/home/jlinfo...
BACKTRACE 26 [fast_function] [/mnt/home/jlinfo...
BACKTRACE 27 [PyEval_EvalCodeEx] [/mnt/home/jlinfo...
BACKTRACE 28 [PyEval_EvalCode] [/mnt/home/jlinfo...
BACKTRACE 29 [run_mod] [/mnt/home/jlinfo...
BACKTRACE 30 [PyRun_SimpleFileExFlags] [/mnt/home/jlinfo...
BACKTRACE 31 [Py_Main] [/mnt/home/jlinfo...
BACKTRACE 32 [pyMPI_Main_with_communicator] [(unknown):0] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/pyMPI-2.5b0/bin/pyMPI]
```

```
% qsub -env TAU_TRACK_SIGNALS=1 ...
```

```
% paraprof
```

# What Caused My Application to Crash?

Right-click to see source code



Name	Value
BACKTRACE 1	[SAMINT::timestep(double, double)] [/mnt/home/jlinford/py-c++-f90-create/SAMINT::timestep.c:3883] [/mnt/home/jlinford/py-c++-f90-create/_samint.so]
BACKTRACE 2	[samarcStep(double, double)] [/mnt/home/jlinford/py-c++-f90-create/pycintfc.C:3883] [/mnt/home/jlinford/py-c++-f90-create/_samint.so]
BACKTRACE 3	[_wrap_samarcStep] [/mnt/home/jlinford/py-c++-f90-create/samint_wrap.c:3883] [/mnt/home/jlinford/py-c++-f90-create/_samint.so]
BACKTRACE 4	[call_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4013] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 5	[fast_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4099] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 6	[PyEval_EvalCodeEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3253] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 7	[PyEval_EvalCode] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:667] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 8	[PyImport_ExecCodeModuleEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/import.c:681] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 9	[load_source_module] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/import.c:1021] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 10	[import_submodule] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/import.c:2596] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 11	[load_next] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/import.c:2416] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 12	[import_module_level] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/import.c:2137] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 13	[builtin___import__] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/builtinmodule.c:49] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 14	[PyObject_Call] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Objects/abstract.c:2529] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 15	[PyEval_CallObjectWithKeywords] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3882] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 16	[PyEval_EvalFrameEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:2333] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 17	[fast_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4099] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 18	[PyEval_EvalCodeEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3253] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 19	[PyEval_EvalCode] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:667] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 20	[run_mod] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/pythonrun.c:1346] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 21	[exec_statement] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4746] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 22	[PyEval_EvalCodeEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3253] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 23	[fast_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4109] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 24	[fast_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4099] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 25	[PyEval_EvalCodeEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3253] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 26	[fast_function] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:4109] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 27	[PyEval_EvalCodeEx] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:3253] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 28	[PyEval_EvalCode] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/ceval.c:667] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 29	[run_mod] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/pythonrun.c:1346] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 30	[PyRun_SimpleFileExFlags] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Python/pythonrun.c:936] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 31	[Py_Main] [/mnt/home/jlinford/0.55/build/Python-2.7.2/Modules/main.c:599] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/Python-2.7.2/lib/libpython2.7.so.1.0]
BACKTRACE 32	[pyMPI_Main_with_communicator] [(unknown):0] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/pyMPI-2.5b0/bin/pyMPI]
BACKTRACE 33	[main] [(unknown):0] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/pyMPI-2.5b0/bin/pyMPI]
BACKTRACE 34	[_libc_start_main] [(unknown):0] [/lib64/libc-2.5.so]
BACKTRACE 35	[_start] [(unknown):0] [/mnt/cfs/pkgs/PTOOLS/pkgs/ptoolsrte-0.55/packages/pyMPI-2.5b0/bin/pyMPI]

# Performance Provenance

TAU: ParaProf Manager

Applications

- Standard Applications
  - Default (jdbc:h2:/Users/jlinfoord/.ParaProf/perfdmf/perfdmf;AUTO\_SERVER=TRUE)
  - ArmyPhasell (jdbc:postgresql://east01.paratools.com:5432/ArmyPhasell)
  - geos (jdbc:postgresql://east01.paratools.com:5432/geos)
  - GraviT (jdbc:postgresql://east01.paratools.com:5432/GraviT)
  - kppa (jdbc:postgresql://east01.paratools.com:5432/kppa)
  - KY05 (jdbc:postgresql://east01.paratools.com:5432/KY05)
  - KY06 (jdbc:postgresql://east01.paratools.com:5432/KY06)
- All Trials
  - Application-STELLAR-subset
  - Application-SAMCart.lightning
    - Experiment-scaling.O3\_io\_nonblocking\_omp
      - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn1.np1.1
      - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn16.np16.1**
      - TIME
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn2.np2.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np1024.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np128.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np2048.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np256.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np32.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np512.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn24.np64.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn4.np4.1
        - SAMCart.scaling.O3\_io\_nonblocking\_omp.craycnl.ppn8.np8.1
    - Experiment-scaling.O3
    - Experiment-scaling.O3\_nonblocking
    - Experiment-O3\_buffered\_nonblocking\_omp
    - All Trials
  - Application-SAMCart.x86\_64
  - Application-SAMCart.mic
    - Experiment-scaling.O3
    - Experiment-scaling.O3\_io\_nonblocking.sampling
    - Experiment-scaling.O3\_omp
    - Experiment-scaling.O3\_novc.sampling
      - SAMCart.scaling.O3\_novc.sampling.mic.np1
    - Experiment-scaling\_omp\_balanced
    - Experiment-scaling
    - Experiment-scaling\_omp\_compact
    - Experiment-scaling.O3\_io\_nonblocking\_omp
    - Experiment-scaling.baseline
    - Experiment-scaling.O3\_io\_nonblocking
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np1
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np2
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np4
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np8
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np16
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np32
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np64
      - SAMCart.scaling.O3\_io\_nonblocking.mic.np128
    - All Trials
  - Application-SAMCart.craycnl
- perfexplorer\_working (jdbc:h2:/Users/jlinfoord/.ParaProf/perfexplorer\_working;AUTO\_SERVER=TRUE)

TrialField	Value
Name	SAMCart.scaling.O3_io_nonblocking_omp.cr...
Application ID	0
Experiment ID	0
Trial ID	321
data_source	0
node_count	16
contexts_per_node	1
threads_per_context	1
total_threads	16
Application	SAMCart.lightning
CPU Cores	12
CPU MHz	2701.000
CPU Type	Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
CPU Vendor	GenuineIntel
CWD	/p/work1/jlinfoord/CART/craycnl_tau/16.1...
Cache Size	30720 KB
Command Line	./samcart-exec samarc/input.samarc.16proc
Executable	/var/opt/cray/alps/spool/2021621/samca...
Experiment	scaling.O3_io_nonblocking_omp
File Type Index	1
File Type Name	TAU profiles
Hostname	nid00687
Local Time	2015-08-07T21:12:52+00:00
MPI Processor Name	nid00687
Memory Size	66072168 kB
Node Name	nid00687
OS Machine	x86_64
OS Name	Linux
OS Release	3.0.101-0.311.1_0502.8394-cray_ari_c
OS Version	#1 SMP Wed Sep 10 04:09:26 UTC 2014
Starting Timestamp	1438981972561763
TAU Architecture	default
TAU Config	-arch=craycnl -papi=/opt/cray/papi/5.3...
TAU Makefile	/p/home/jlinfoord/workspace/CART/tau2/c...
TAU MetaData Merge Time	4.9E-05 seconds
TAU Version	2.24.1-git
TAU_BFD_LOOKUP	on
TAU_CALLPATH	off
TAU_CALLPATH_DEPTH	2
TAU_CALLSITE_LIMIT	1
TAU_COMM_MATRIX	off
TAU_COMPENSATE	off
TAU_CUPTI_API	runtime
TAU_EBS_KEEP_UNRESOLVED_ADDR	off
TAU_IBM_BG_HWP_COUNTERS	off
TAU_MAX_THREADS	1
TAU_MEASURE_TAU	off
TAU_MEMDBG_PROTECT_ABOVE	off
TAU_MEMDBG_PROTECT_BELOW	off
TAU_MEMDBG_PROTECT_FREE	off
TAU_OPENMP_RUNTIME	on
TAU_OPENMP_RUNTIME_EVENTS	on
TAU_OPENMP_RUNTIME_STATES	off
TAU_PROFILE	on
TAU_PROFILE_FORMAT	nprofile

Keeping Up With Technology

---

# PARATOOLS THREADSPOTTER

# ParaTools ThreadSpotter

- A cache and memory optimization tool integrated with TAU
  - Analyzes **memory bandwidth** and latency, data locality and **thread communication**
  - Identifies specific issues and pinpoints troublesome areas in source code
  - Provides guidance towards a resolution
- Provides **qualitative** measurement:
  - Data is not dependent on the hardware
  - Can predict performance for other memory systems
    - E.g. Intel Xeon Phi “Knights Landing” MCDRAM vs. DDR

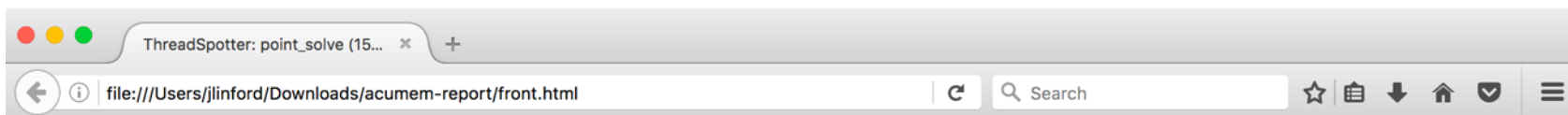


# Questions ThreadSpotter can Answer

- **Miss ratio:** What is the likelihood that a memory access will miss in a cache?
- **Miss rate:** D:0 per time unit, e.g. per-second, per-1000-instructions
- **Fetch ratio/rate\*:** What is the likelihood that a memory access will cause a fetch to the cache [including HW prefetching]
- **Fetch utilization\*:** What fraction of a cacheline was used before it got evicted
- **Writeback utilization\*:** What fraction of a cacheline written back to memory contains dirty data
- **Communication utilization\*:** What fraction of a communicated cacheline is ever used?

\*Terms used in ParaTools ThreadSpotter reports.

# ThreadSpotter Report Front Page



## ThreadSpotter™

ThreadSpotter™ is a tool to quickly analyze an application for a range of performance problems, particularly related to multicore optimization.

[Read more...](#) [Manual](#)

[Open the Report](#)

## Your application

Application: ./point\_solve



### Memory Bandwidth

The memory bus transports data between the main memory and the processor. The capacity of the memory bus is limited. Abuse of this resource limits application scalability.

[Manual: Bandwidth](#)



### Memory Latency

The regularity of the application's memory accesses affects the efficiency of the hardware prefetcher. Irregular accesses causes cache misses, which forces the processor to wait a lot for data to arrive.

[Manual: Cache misses](#) [Manual: Prefetching](#)



### Data Locality

Failure to pay attention to data locality has several negative effects. Caches will be filled with unused data, and the memory bandwidth will waste transporting unused data.

[Manual: Locality](#)



### Thread Communication / Interaction

Several threads contending over ownership of data in their respective caches causes the different processor cores to stall.

[Manual: Multithreading](#)

This means that your application shows opportunities to:

**Avoid major processor stalls due to irregular access patterns**

[Read more...](#)

## ParaTools

### Next Steps

The prepared report is divided into sections.

- Select the tab **Summary** to see global statistics for the entire application.
- Select the tabs **Bandwidth Issues**, **Latency Issues** and **MT Issues** to browse through the detected problems.
- Select the tab **Loops** to browse through statistics and detected problems loop by loop.

The Issue and Source windows contain details and annotated source code for the detected problems.



### Resources

[Manual](#)

[Table of Contents](#)

[Optimization Workflow](#)

[Reading the Report](#)

[ParaTools Web Site](#)

[ParaTools Web Site](#)

[Overview](#)

[Concepts](#)

[Issue Reference](#)

[ThreadSpotter](#)

# ThreadSpotter Report

ThreadSpotter: point\_solve (15... x +)

file:///Users/[inford]/Downloads/acumem-report/main.html

Issues | Loops | Summary | Files | Execution | About/Help

Bandwidth Issues | Latency Issues | Multi-Threading Issues | Pollution Issues

#	Issue type	% of bandwidth	% of fetches	% of write-backs	Fetch utilization	Write-back utilization
6	Fetch hot-spot	66.7%	69.6%	0.0%	99.2%	100.0%
16	Spat/temp blocking	66.7%	69.6%	0.0%	99.2%	100.0%
11	Random access	10.1%	10.5%	0.0%	73.8%	100.0%
15	Spat/temp blocking	10.1%	10.5%	0.0%	73.8%	100.0%
9	Fetch hot-spot	8.2%	8.5%	0.0%	95.3%	100.0%
13	Loop fusion	8.2%	8.5%	0.0%	95.3%	100.0%
18	Spat/temp blocking	8.2%	8.5%	0.0%	95.3%	100.0%
10	Fetch utilization	7.5%	5.0%	64.5%	49.6%	92.8%
4	Fetch hot-spot	2.7%	2.9%	0.0%	100.0%	100.0%
14	Spat/temp blocking	2.7%	2.9%	0.0%	100.0%	100.0%
8	Write-back hot-spot	1.7%	0.9%	21.0%	0.0%	86.5%
7	Fetch hot-spot	1.5%	1.5%	0.0%	99.7%	100.0%
12	Loop fusion	1.5%	1.5%	0.0%	99.7%	100.0%
17	Spat/temp blocking	1.5%	1.5%	0.0%	99.7%	100.0%
3	Write-back hot-spot	1.3%	0.7%	14.5%	42.9%	63.9%

**Issue #11: Random access**

This instruction group also shows symptoms of: Fetch hot-spot.

Statistics for instructions of this issue

Accesses	1.45e+09
% of misses	12.7%
% of bandwidth	10.1%
% of fetches	10.5%
% of write-backs	0.0%
% of upgrades	---
Miss ratio	4.2%
Fetch ratio	4.7%
Write-back ratio	0.0%
Upgrade ratio	0.0%
Communication ratio	0.0%
Fetch utilization	73.8%
Write-back utilization	100.0%
Communication utilization	100.0%
False sharing ratio	0.0%

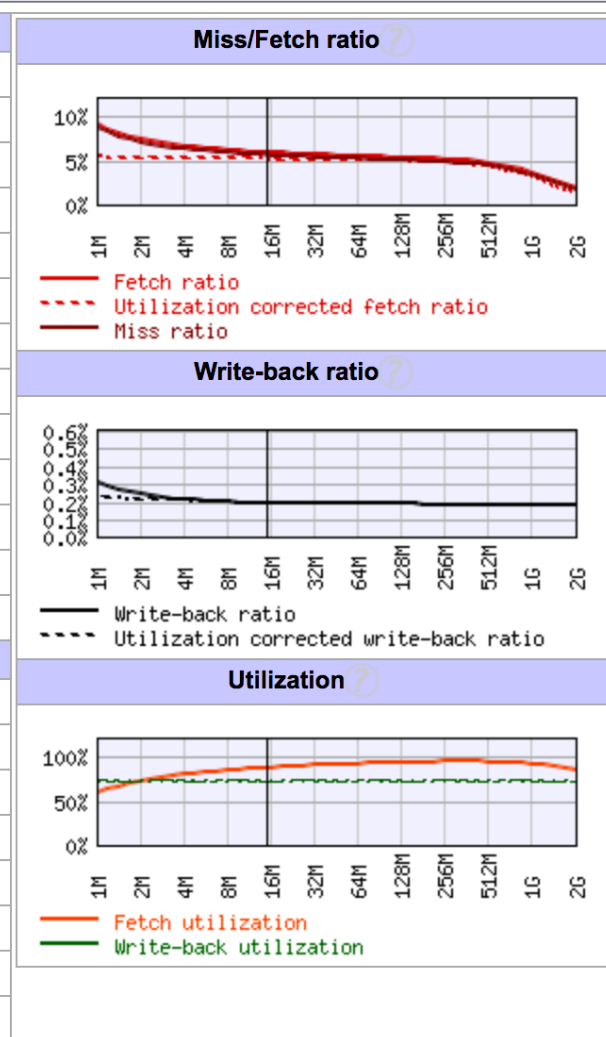
Thread Id	Fetch utilization
12835	99.2%

216	
217	0.3% f1 = f1 - a_off(1,1,j)*dq(1,icol)
218	f2 = f2 - a_off(2,1,j)*dq(1,icol)
219	f3 = f3 - a_off(3,1,j)*dq(1,icol)
220	f4 = f4 - a_off(4,1,j)*dq(1,icol)
221	f5 = f5 - a_off(5,1,j)*dq(1,icol)
222	
223	0.3% f1 = f1 - a_off(1,2,j)*dq(2,icol)
224	f2 = f2 - a_off(2,2,j)*dq(2,icol)
225	f3 = f3 - a_off(3,2,j)*dq(2,icol)
226	f4 = f4 - a_off(4,2,j)*dq(2,icol)
227	f5 = f5 - a_off(5,2,j)*dq(2,icol)
228	
229	0.4% f1 = f1 - a_off(1,3,j)*dq(3,icol)
230	f2 = f2 - a_off(2,3,j)*dq(3,icol)
231	f3 = f3 - a_off(3,3,j)*dq(3,icol)
232	f4 = f4 - a_off(4,3,j)*dq(3,icol)
233	f5 = f5 - a_off(5,3,j)*dq(3,icol)
234	
235	0.5% f1 = f1 - a_off(1,4,j)*dq(4,icol)
236	f2 = f2 - a_off(2,4,j)*dq(4,icol)
237	f3 = f3 - a_off(3,4,j)*dq(4,icol)
238	f4 = f4 - a_off(4,4,j)*dq(4,icol)
239	f5 = f5 - a_off(5,4,j)*dq(4,icol)
240	
241	67.4% f1 = f1 - a_off(1,5,j)*dq(5,icol)
	% of fetches Miss ratio Fetch ratio WB ratio Fetch Util WB Util PC Type Issues
	8.9% 18.1% 20.0% 0.0% 73.8% 100.0% 0x421840 R
	44.8% 83.9% 100.0% 0.0% 99.2% 100.0% 0x421846 R
	13.5% 28.3% 31.3% 0.0% 99.2% 100.0% 0x42186f R
	0.0% 0.0% 0.1% 0.0% 99.2% 100.0% 0x4218a4 R
	0.1% 0.1% 0.2% 0.0% 99.2% 100.0% 0x4218c7 R
	0.0% 0.0% 0.1% 0.0% 99.2% 100.0% 0x4218ea R
242	2.6% f2 = f2 - a_off(2,5,j)*dq(5,icol)
243	2.9% f3 = f3 - a_off(3,5,j)*dq(5,icol)
244	2.8% f4 = f4 - a_off(4,5,j)*dq(5,icol)
245	2.8% f5 = f5 - a_off(5,5,j)*dq(5,icol)
246	
247	end do
248	
249	! Forward...sequential access to a_diag_lu.
250	
251	0.8% f2 = f2 - a_diag_lu(2,1,n)*f1
252	0.4% f3 = f3 - a_diag_lu(3,1,n)*f1
253	0.3% f4 = f4 - a_diag_lu(4,1,n)*f1
254	0.4% f5 = f5 - a_diag_lu(5,1,n)*f1

# Application Performance Summary

Issues | Loops | Summary | Files | Execution | About/Help

Global statistics	
Accesses ?	1.16e+10
Misses ?	6.55e+08
Fetches ?	6.92e+08
Write-backs ?	2.26e+07
Upgrades ?	0.00e+00
Miss ratio ?	5.7%
Fetch ratio ?	6.0%
Write-back ratio ?	0.2%
Upgrade ratio ?	0.0%
Communication ratio ?	0.0%
Fetch utilization ?	88.1%
Write-back utilization ?	100.0%
Communication utilization ?	6.5%
Analysis parameters	
Processor model ?	Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz (auto)
Number of CPUs ?	2
Number of caches ?	2
Cache level ?	3
Cache size ?	15M
Line size ?	64
Replacement policy ?	random
Software prefetches active ?	Yes



# Performance Issues

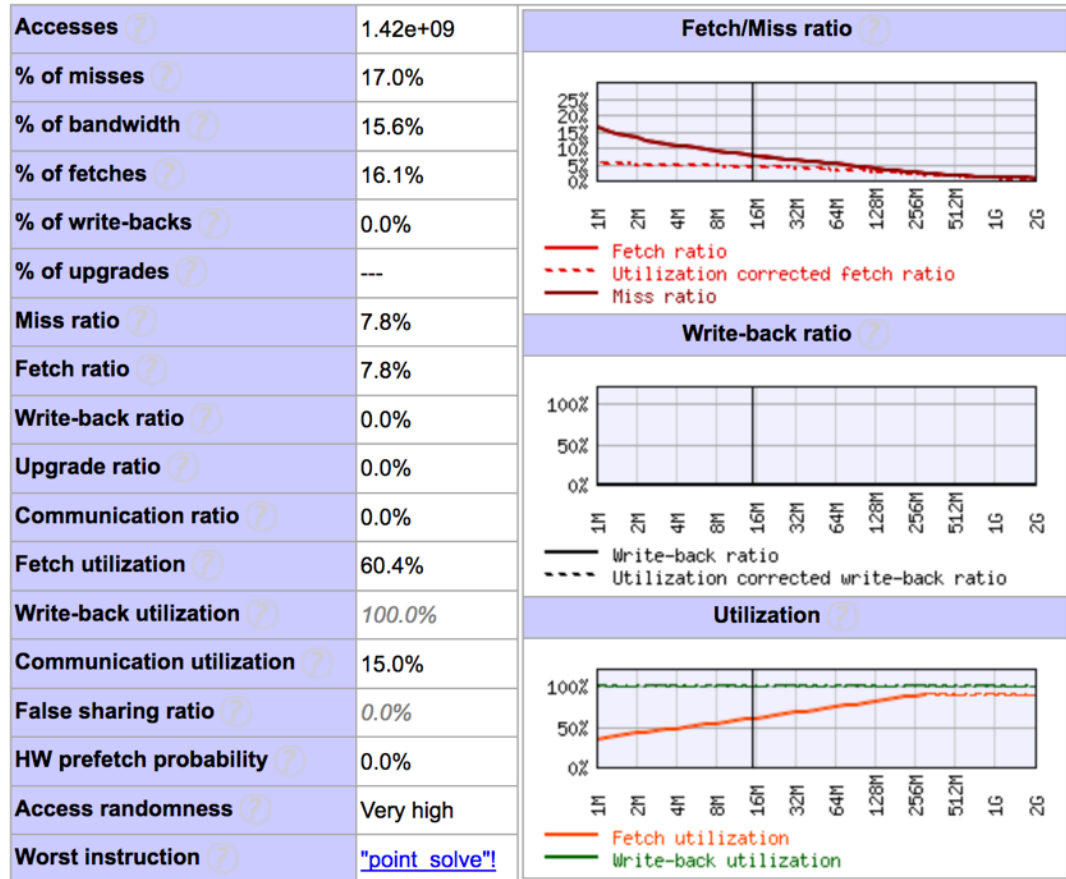
Bandwidth Issues		Latency Issues	Multi-Threading Issues	Pollution Issues			
#		Issue type Filter: All	% of bandwidth▲	% of fetches	% of write-backs	Fetch utilization	Write-back utilization
<a href="#">13</a>		<a href="#">Inefficient loop nesting</a>	63.7%	65.8%	0.0%	96.2%	100.0%
<a href="#">23</a>		<a href="#">Spat/temp blocking</a>	63.7%	65.8%	0.0%	96.2%	100.0%
<a href="#">18</a>		<a href="#">Random access</a>	15.6%	16.1%	0.0%	60.4%	100.0%
<a href="#">22</a>		<a href="#">Spat/temp blocking</a>	15.6%	16.1%	0.0%	60.4%	100.0%
<a href="#">16</a>		<a href="#">Inefficient loop nesting</a>	6.6%	6.9%	0.0%	100.0%	100.0%
<a href="#">20</a>		<a href="#">Loop fusion</a>	6.6%	6.9%	0.0%	100.0%	100.0%
<a href="#">25</a>		<a href="#">Spat/temp blocking</a>	6.6%	6.9%	0.0%	100.0%	100.0%
<a href="#">10</a>		<a href="#">Fetch utilization</a>	6.3%	4.5%	62.1%	51.5%	100.0%
<a href="#">12</a>		<a href="#">Inefficient loop nesting</a>	2.6%	2.7%	0.0%	88.3%	100.0%
<a href="#">21</a>		<a href="#">Spat/temp blocking</a>	2.6%	2.7%	0.0%	88.3%	100.0%
<a href="#">14</a>		<a href="#">Inefficient loop nesting</a>	1.6%	1.7%	0.0%	86.8%	100.0%
<a href="#">19</a>		<a href="#">Loop fusion</a>	1.6%	1.7%	0.0%	86.8%	100.0%
<a href="#">24</a>		<a href="#">Spat/temp blocking</a>	1.6%	1.7%	0.0%	86.8%	100.0%
<a href="#">15</a>		<a href="#">Inefficient loop nesting</a>	1.2%	0.6%	19.6%	0.0%	100.0%
<a href="#">3</a>		<a href="#">Write-back hot-spot</a>	1.1%	0.7%	12.8%	43.5%	86.1%
<a href="#">17</a>		<a href="#">Inefficient loop nesting</a>	0.3%	0.3%	2.1%	57.4%	30.5%
<a href="#">26</a>		<a href="#">Spat/temp blocking</a>	0.3%	0.3%	2.1%	57.4%	30.5%

# Random Access Issue Detail

## Issue #18: Random access

This instruction group also shows symptoms of:   Fetch utilization,   Fetch hot-spot.



### Statistics for instructions of this issue



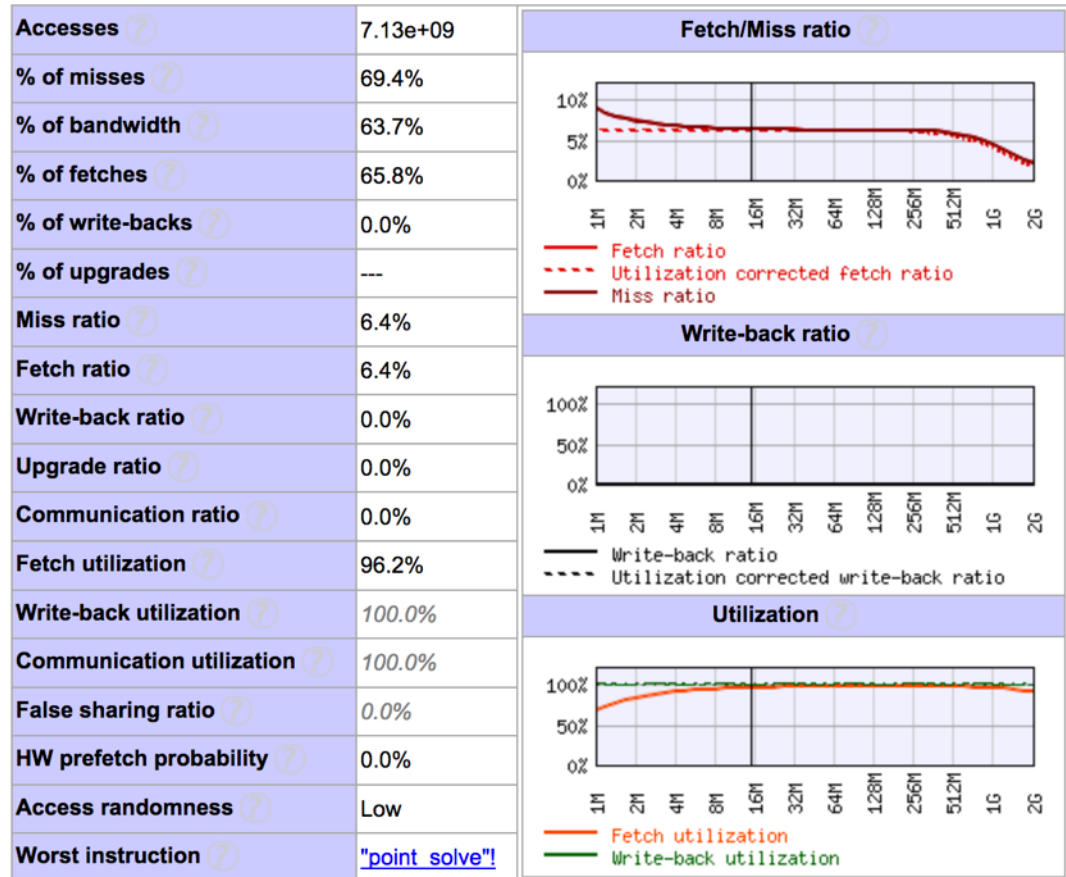
If the program was changed as to reach 100% fetch utilization, fetches in this instruction group would be reduced with 42.4%, and total number of fetches would be reduced with 6.8%.

# Inefficient Loop Nesting Issue Detail

## Issue #13: Inefficient loop nesting

This instruction group also shows symptoms of:   Fetch hot-spot.

### Statistics for instructions of this issue



If the program was changed as to reach 100% fetch utilization, fetches in this instruction group would be reduced with 3.8%, and total number of fetches would be reduced with 2.5%.

# Source Code / Issue Mapping

Address	Percentage	Code	Performance Metrics								Issue Types			
	% of fetches	Miss ratio	Fetch ratio	WB ratio	Fetch Util	WB Util	PC	Type	Issues					
244	64.4%	f1 = f1 - a_off(1,5,j)*dq(5,icol)												
	13.0%	31.5%	31.5%	0.0%	60.4%	100.0%	0x421ede	R			ST			
	37.9%	100.0%	100.0%	0.0%	96.2%	100.0%	0x421ee5	R			ST		Pf NT	
	12.8%	31.0%	31.0%	0.0%	96.2%	100.0%	0x421f0f	R			ST		Pf NT	
	0.2%	0.6%	0.6%	0.0%	96.2%	100.0%	0x421f47	R			ST		Pf NT	
	0.2%	0.5%	0.5%	0.0%	96.2%	100.0%	0x421f6a	R			ST		Pf NT	
	0.4%	1.0%	1.0%	0.0%	96.2%	100.0%	0x421f8d	R			ST		Pf NT	
245	4.0%	f2 = f2 - a_off(2,5,j)*dq(5,icol)												
											ST		Pf NT	
246	3.5%	f3 = f3 - a_off(3,5,j)*dq(5,icol)												
											ST		Pf NT	
247	3.3%	f4 = f4 - a_off(4,5,j)*dq(5,icol)												
											ST		Pf NT	
248	3.6%	f5 = f5 - a_off(5,5,j)*dq(5,icol)												
											ST		Pf NT	



# ThreadSpotter Online Help

ThreadSpotter: point\_solve (15... x) 7.1. Statistics x +

file:///Users/jlinford/Downloads/acumem-report/manual\_html/report\_statistics.html

["Access Randomness"](#).

Worst instruction

Points out the instruction that causes the largest number of cache line fetches in this part of the program, and the source code line that generated it.

## 7.1.2. Reading the Diagrams

The report contains diagrams describing several cache size dependent application characteristics. The summary tab in the summary frame shows application global values, while the individual issues and loops show values related to their respective instruction groups.

The diagrams plot their values for different cache sizes, from an 8 kilobyte cache to a 16 megabyte cache in the following example. The cache size that the report focuses on is marked with a vertical black line, in this case at 64 kilobytes.

### 7.1.2.1. Fetch/Miss Ratio Diagram

**Figure 7.6. Fetch/Miss Ratio Diagram**

- *Bright red line*  
Fetch ratio, the ratio of memory operations in the program, loop, issue or instruction group that, directly or indirectly through hardware prefetching, cause a data transfer between memory and cache. See [Section 3.8, "Fetch Ratio"](#).
- *Red dotted line*  
Utilization corrected fetch ratio. Fetch ratio if the fetch utilization was raised to 100%. See [Section 4.8, "Utilization Corrected Fetch Ratio"](#).
- *Dark red line*  
Miss ratio, the ratio of memory operations in the program that stall due to cache misses. The difference between the fetch ratio and the miss ratio is caused by software and hardware prefetching. See [Section 3.4, "Cache Misses"](#).

### 7.1.2.2. Write-Back Ratio Diagram

Keeping Up With Technology

---



# SUCCESS STORIES

# NASA FUN3D

*“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! :)”*



**One of the largest and most accurate wing-body-nacelle simulations of 2016.**

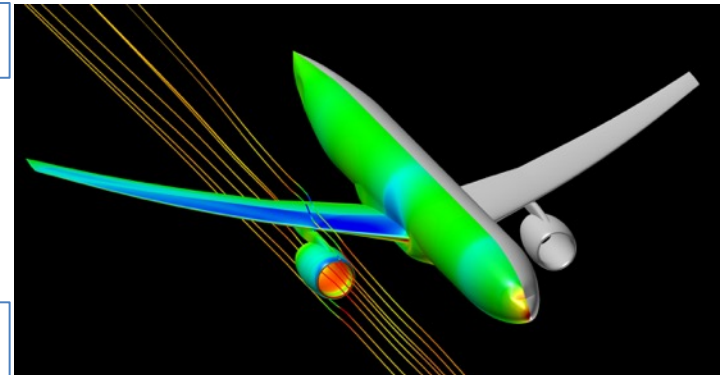
- 1,651,089,924 grid points
- 5,902,801,476 tetrahedral elements
- 1,310,290,264 prismatic elements
- 14,400 Ivy Bridge cores

291.529		FLOW::STEP_SOLVER [{{flow.F90}} {1382,3}–{1551,28}]
291.317		RELAX_STEADY::RELAX [{{relax_steady.f90}} {29,3}–{236,22}]



33% runtime improvement

198.656		FLOW::STEP_SOLVER [{{flow.F90}} {1382,3}–{1551,28}]
198.531		RELAX_STEADY::RELAX [{{relax_steady.f90}} {29,3}–{236,22}]



# DOD CREATE-AV



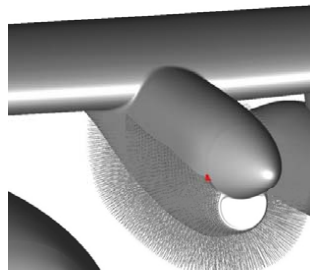
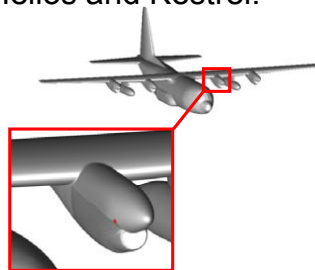
## Strand Technology

### Technology Drivers

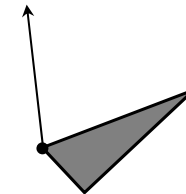
- Timeliness (automation of mesh generation)
- Timeliness (automation and scalability of domain connectivity)
- Timeliness/Physical accuracy (computational efficiency and scalability of aerodynamic solvers)
- Processor architecture (small memory footprint maps well to hierarchical memory architectures, e.g., multi-core, GPU)

### CREATE-AV Example

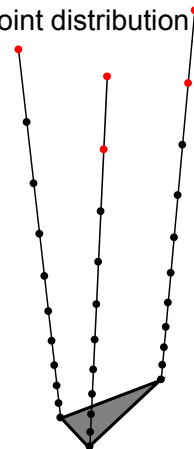
This is a new meshing paradigm introduced in 2007 by current members of the CREATE-AV technical staff. The technology is being matured in the Helios product and will be deployed through both Helios and Kestrel.



Strand  
pointing vector



Strand  
point distribution

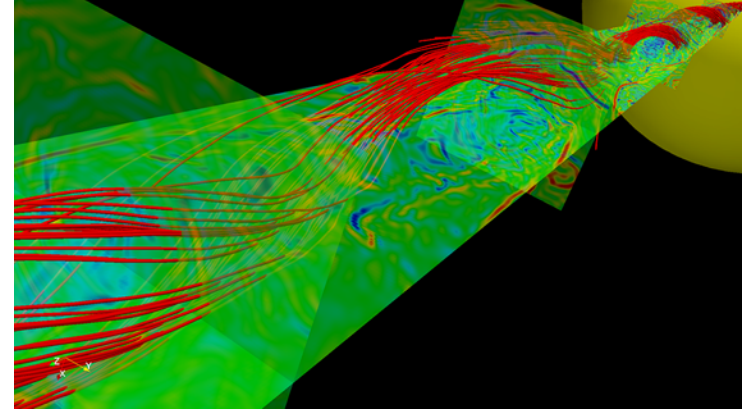






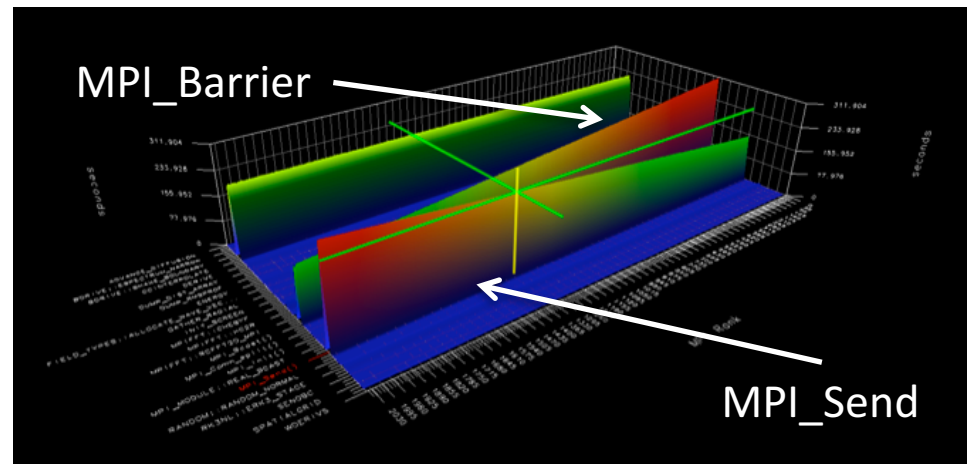
# DOE IRMHD

- INCITE magnetohydrodynamic simulation to understand solar winds and coronal heating
  - First direct numerical simulations of Alfvén wave (AW) turbulence in extended solar atmosphere accounting for inhomogeneities
  - Team
    - University of New Hampshire (Jean Perez and Benjamin Chandran)
    - ALCF (Tim Williams)
    - University of Oregon (Sameer Shende)
- IRMHD (Inhomogeneous Reduced Magnetohydrodynamics)
  - Fortran 90 and MPI
  - Excellent weak and strong scaling properties
  - Tested and benchmarked on Intrepid and Mira
- HPC Source article and ALCF news  
<https://www.alcf.anl.gov/articles/furthering-understanding-coronal-heating-and-solar-wind-origin>



# IRMHD Communication Analysis

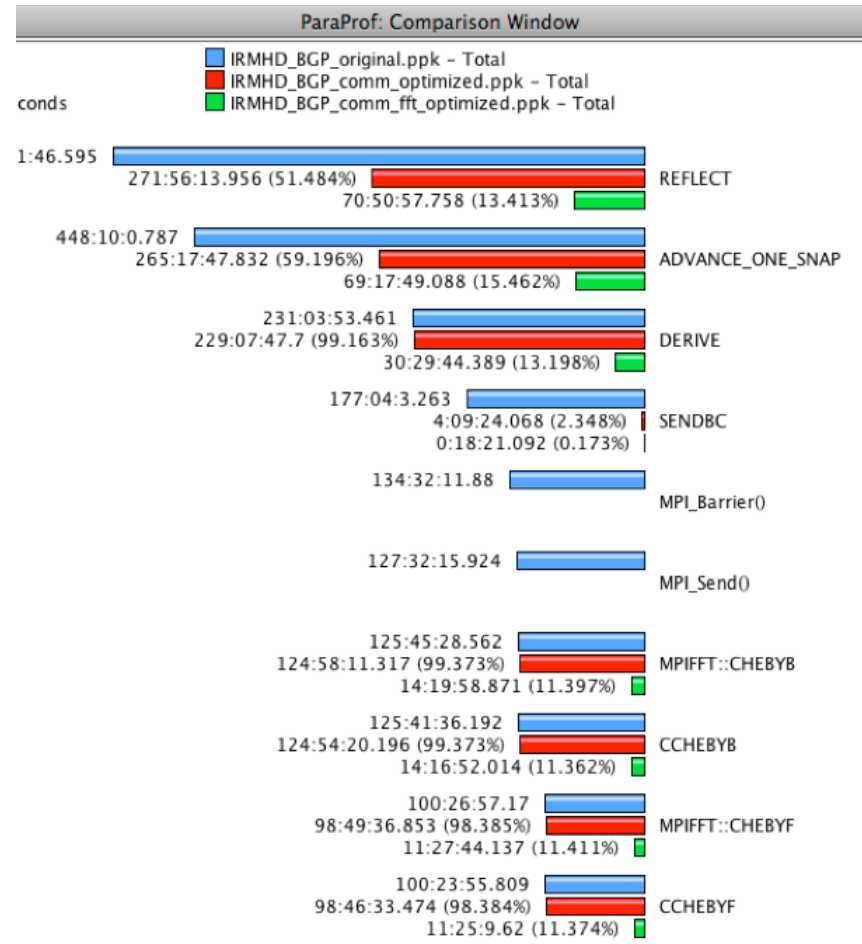
- Source-based (direct) instrumentation
- MPI instrumentation and volume measurement
- IRMHD exhibited significant synchronous communication bottlenecks
- On 2,408 cores:
  - **MPI\_Send** and **MPI\_Bcast** take significant time
  - Opportunities for communication/computation overlap
  - Identified possible targets for computation improvements





# IRMHD Optimization

- On 2,408 cores, overall execution time reduced from 528.18 core hours to 70.8 core hours (>7x improvement)
- Non-blocking communication substrate
- More efficient implementation of underlying FFT

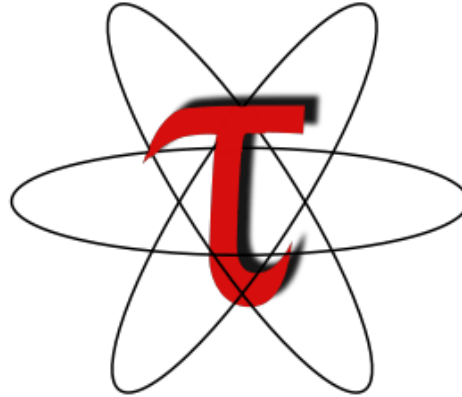


Python Performance Evaluation

---

# CONCLUSION

# Try it out!



<http://www.paratools.com/TAU>

<http://www.paratools.com/ThreadSpotter>

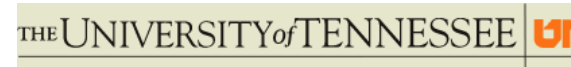
**Free download, open source, BSD license**

[jlinford@paratools.com](mailto:jlinford@paratools.com)

**(540) 808-9250**

# Acknowledgements

- Department of Energy
  - Office of Science
  - Argonne National Laboratory
  - Oak Ridge National Laboratory
  - NNSA/ASC Trilabs (SNL, LLNL, LANL)
- HPCMP DoD PETTT Program
- National Science Foundation
  - Glassbox, SI-2
- University of Tennessee
- University of New Hampshire
  - Jean Perez, Benjamin Chandran
- University of Oregon
  - Allen D. Malony, Sameer Shende
  - Kevin Huck, Wyatt Spear
- TU Dresden
  - Holger Brunst, Andreas Knupfer
  - Wolfgang Nagel
- Research Centre Jülich
  - Bernd Mohr
  - Felix Wolf



UNIVERSITY OF OREGON

