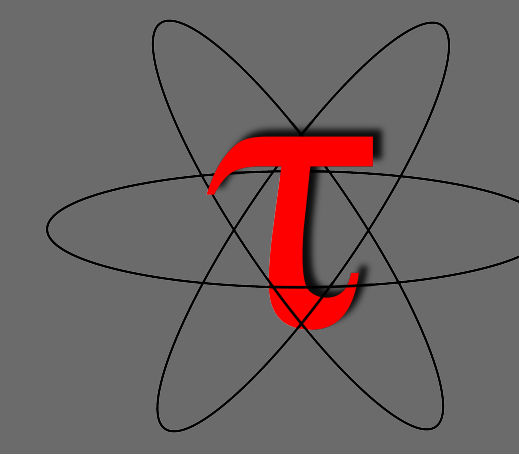


EFFICIENT PARALLEL RUNTIME BOUNDS CHECKING WITH THE TAU PERFORMANCE SYSTEM

John C. Linford, Sameer Shende, Allen D. Malony
ParaTools, Inc., Eugene, OR

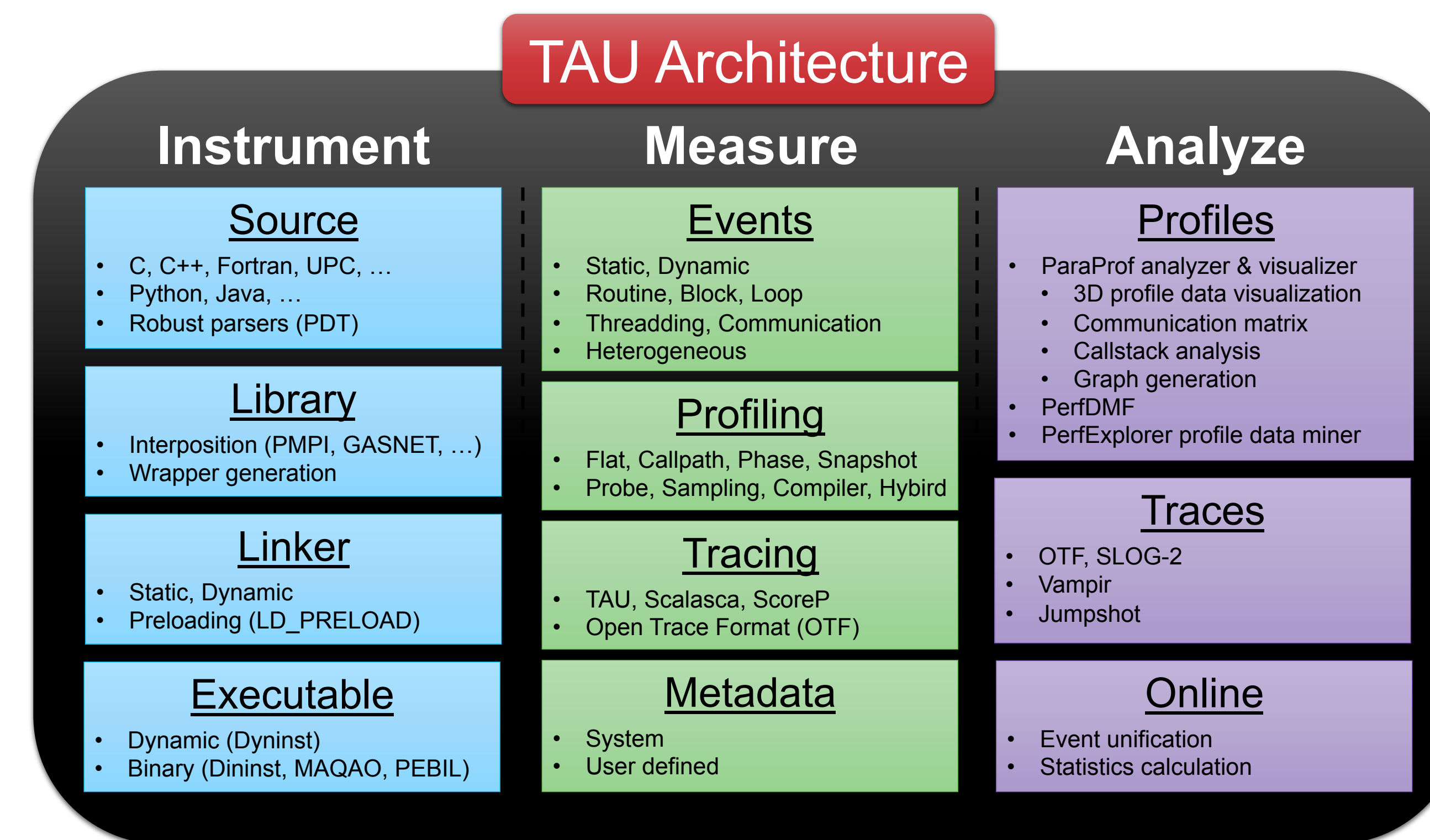


Andrew Wissink
U.S. Army Aviation Development Directorate - AFDD
NASA Ames Research Center, Moffett Field, CA

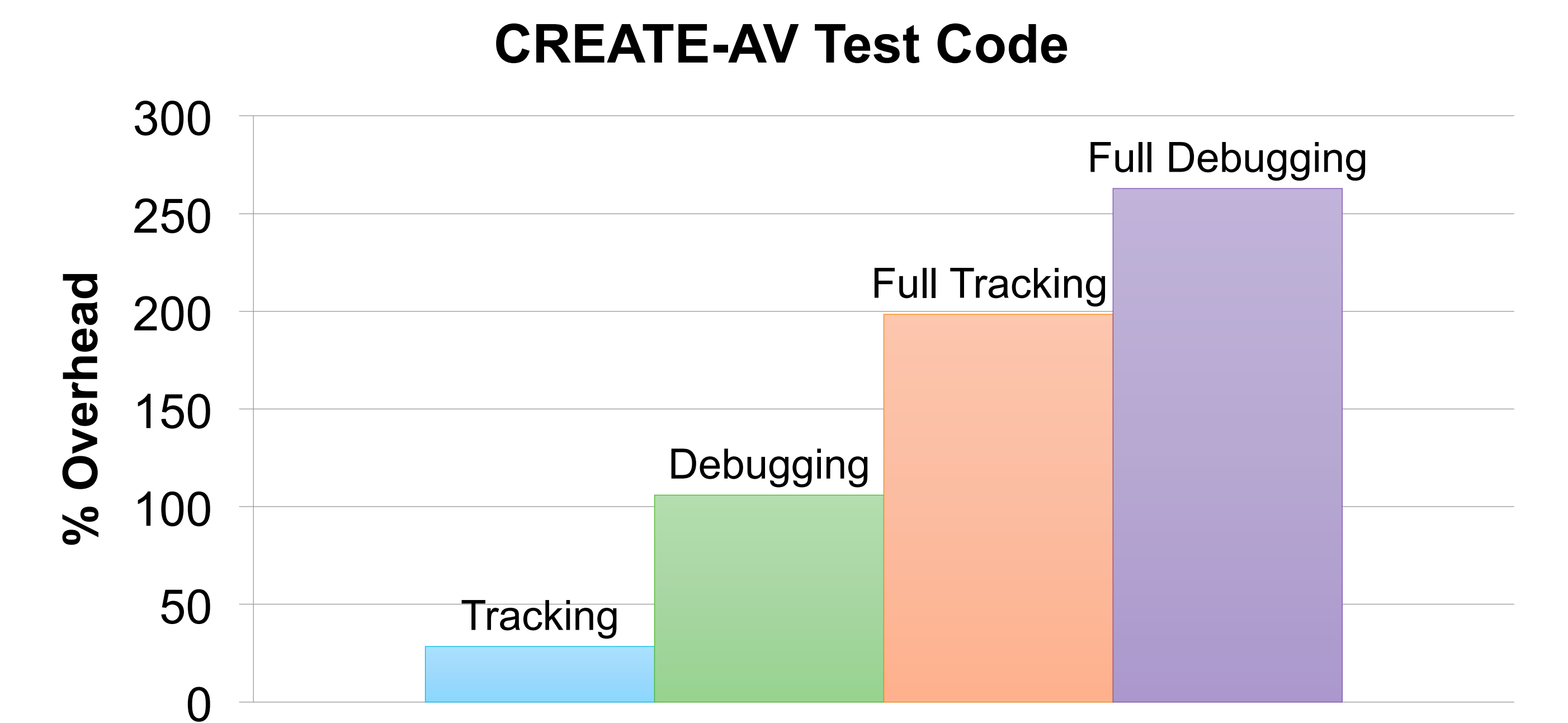
Problems Addressed

- **Debug memory errors in parallel codes written in multiple languages.**
 - When and where do memory errors occur?
 - What is the heap memory usage?
 - Was the error cause by a read or a write?
 - Which processes or threads experienced the error?
 - Were there any memory leaks in the application?
 - What were the performance characteristics prior to fault?
- **Protect sensitive data.**
 - Generate reports free of application data.
 - Open and portable debugging data file format.

The TAU Performance System



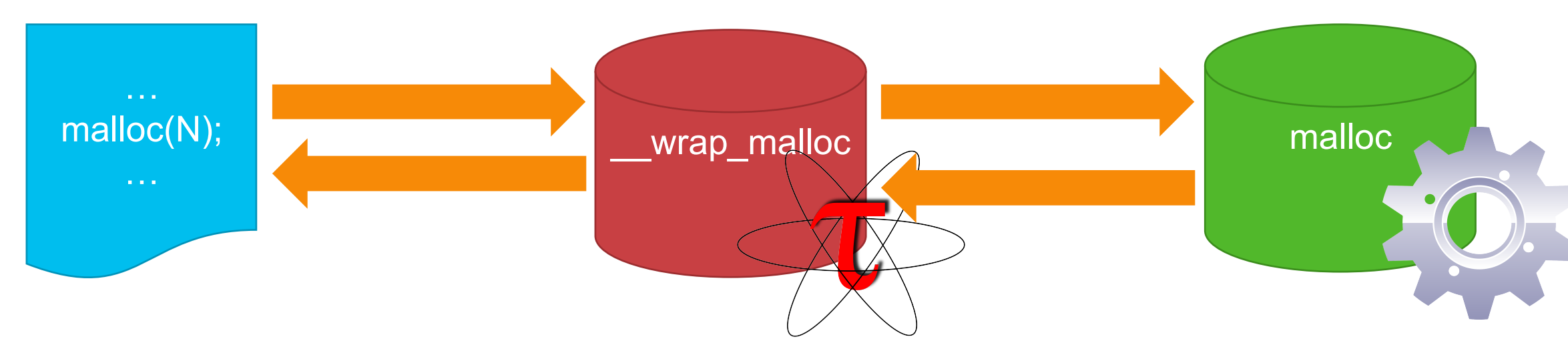
Results



Approach

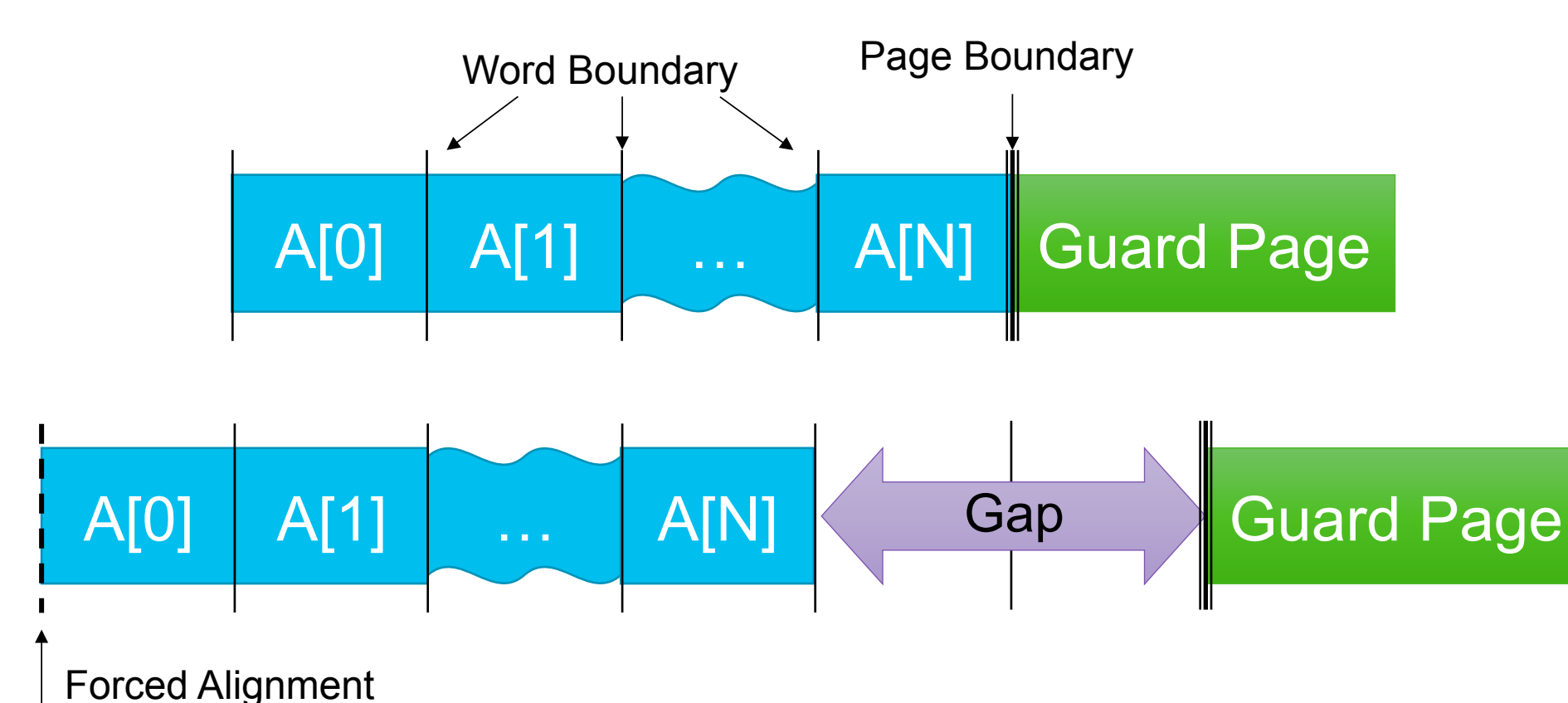
1. Instrument heap memory allocation and deallocation calls.

- Record heap memory use, allocation and deallocation rate.
- Associate every allocation with its use in the source code.



2. Allocate guard pages before and/or after memory allocations.

- Use the MMU hardware to protect regions of memory from access.
- Fast invalid memory access detection but high memory overhead.



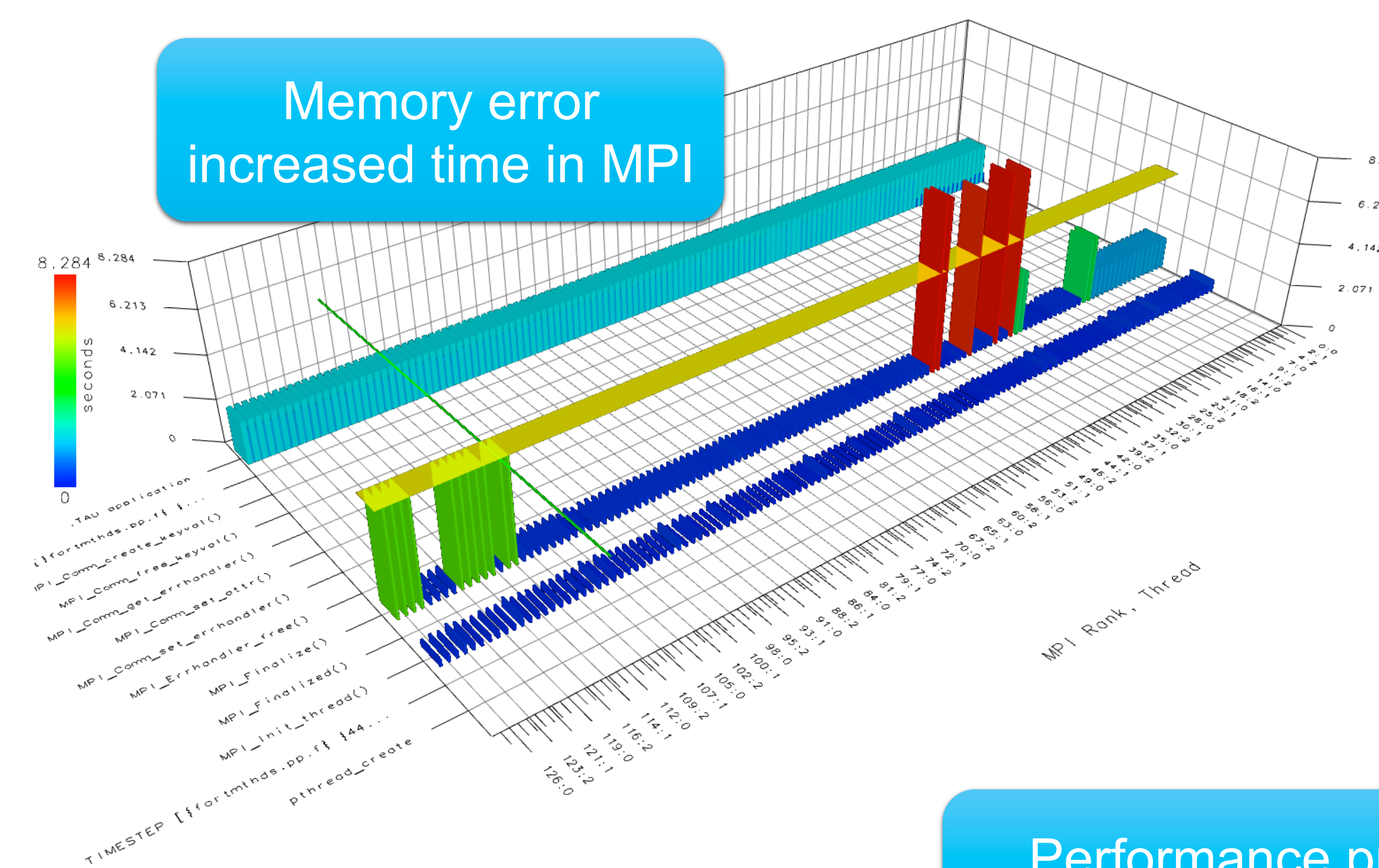
3. Intercept error signal if error occurs.

- Save backtrace in the application profile.
- Write application profile to disk and gracefully shut down.

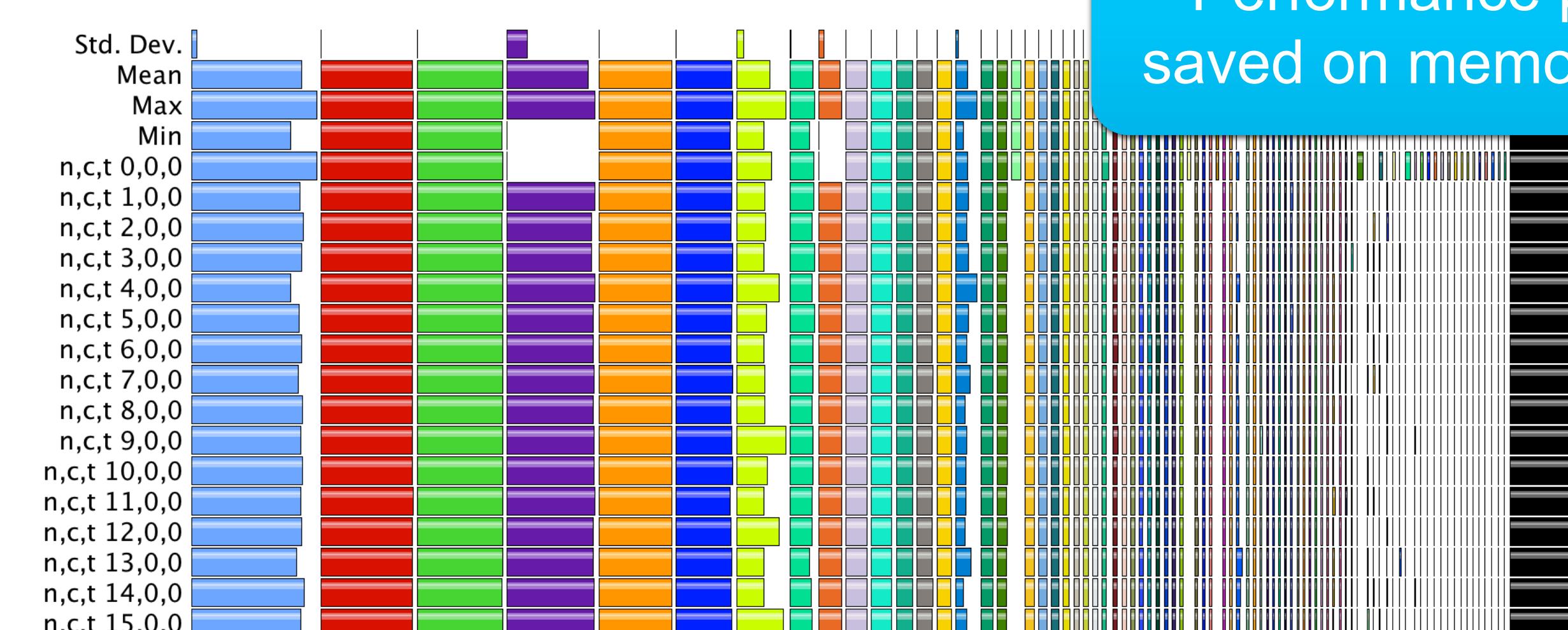
4. Send application profile to developers for analysis.

- Profile contains no application input or memory content information.

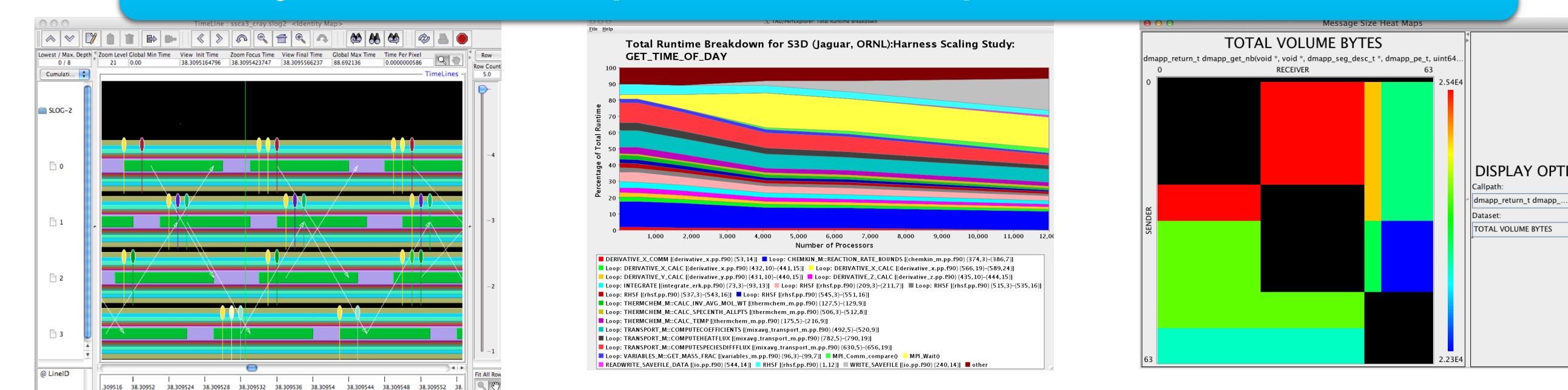
Memory error increased time in MPI



Performance profile saved on memory error



Analyze with Jumpshot, PerfExplorer, and ParaProf



Backtrace across several languages

runStep	Heap Allocate	Heap Memory Used (KB) at Entry	Heap Memory Used (KB) at Exit
runStep [[samarc.py][104]]	88	1	88
Heap Memory Used (KB) at Entry	54,053.779	1	54,053.779
Heap Memory Used (KB) at Exit	54,062.459	1	54,062.459
samarcStep			
Heap Memory Used (KB) at Entry	54,053.865	1	54,053.865
Heap Memory Used (KB) at Exit	54,062.459	1	54,062.459
wrap_samarcStep			
Heap Memory Used (KB) at Entry	54,053.865	1	54,053.865
Heap Memory Used (KB) at Exit	54,062.459	1	54,062.459
int samarcStep(double, double)			
Heap Memory Used (KB) at Entry	54,053.865	1	54,053.865
Heap Memory Used (KB) at Exit	54,062.459	1	54,062.459
void SAMINT::timestep(double, double)			
Heap Allocate	8,800	2	8,000
Heap Memory Used (KB) at Entry	54,053.865	1	54,053.865
Heap Memory Used (KB) at Exit	54,062.459	1	54,062.459
MEMORY LEAK! Heap Allocate	17,256,763	9,327	1,835,520
INVALID memory access	1	1	1

View memory events and statistics

- 236835 Heap Memory Used (KB)
- 131283 Heap Allocate
- 93961 Heap Free
- 53148 Heap Memory Used (KB) at Entry
- 53148 Heap Memory Used (KB) at Exit
- 37322 MEMORY LEAK! Heap Allocate : .TAU a...
- 19368 Heap Allocate : .TAU application => OurMain()
- 17712 Heap Allocate : .TAU application => OurMain()
- 15576 Heap Free : .TAU application => OurMain() =>
- 6112 Heap Allocate : .TAU application => OurMain()

Related Work

- H. Aygün, D.U.M.A – Detect Unintended Memory Access, <http://duma.sourceforge.net/>, May 2013.
- Apple Corp., "Guard Malloc Manual Page," <https://developer.apple.com/>, March 2009.
- B. Perens, "efence – Electric Fence Malloc Debugger," <http://linux.die.net/man/3/efence>, 1999.
- J. Seward and N. Nethercote, "Using Valgrind to detect unintended value errors with bit-precision," in *USENIX ATC*, Anaheim, CA, USA, April 2005.
- K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "Address-sanitizer: A fast address sanity checker," in *USENIX ATC*, 2012.
- Microsoft Corp., "GFlags and PageHeap," <http://msdn.microsoft.com/>, July 2013.
- S. Shende, A. D. Malony, J. Linford, A. Wissink, and S. Adamec, "Isolating Runtime Faults with Callstack Debugging using TAU," in *Proceedings of the IEEE HPEC 2012 Conference*. IEEE Computer Society, 2012.