High performance tools to debug, profile, and analyze your applications

High-productivity development tools for science

Beau Paisley
bpaisley@allinea.com
HPC means being able to work productively on remote machines

- Linux
- OS/X
- Windows
- Multiple hop SSH
- RSA + Cryptocard
- Uses server license
Submit to job queues or run interactively on any system
Open and view files directly on the remote system
We’ve been asked to look at a profile recorded for a wave equation solver.
We go the extra mile to make products that just work

- Small data files
- <5% slowdown
- No instrumentation
- No recompilation
57.2% of the time is spent swapping arrays – this isn’t even in an OpenMP loop
There’s a much quicker way to swap arrays than by copying their data.
We can build this change on the remote system directly
We can build this change on the remote system directly.
And commit it to source control
How much faster will the application be now?
Perhaps we should have tested it before committing that change…
While still connected to the server we switch to the debugger
It’s already configured to reproduce the profiling run
Our tools understand your version control system
Most new bugs are in or around recently changed code
We can visualize multidimensional data across all processes.
And generate statistical summaries of their contents
Let’s run one iteration. Changes are automatically highlighted:
Variables are compared across all threads and processes automatically
These arrays are all pointing to the same area of memory!
Let’s fix that right now
This time we’ll verify our fix before committing it.
A tracepoint shows the arrays pointers are swapping correctly now
We have achieved just under a 2x speedup already
Now the majority of the time is in the `do_math` function where it belongs.
But this function is not using vectorized operations…
Can we remove this conditional from the inner loop?
We can zero the endpoints after the computation
The new code compiles so let’s commit our changes and see if it’s faster.
Apparently it's only 1% faster... what's going on?
The floating point operations are more efficient but the code is memory-bound.
Optimizing a 1-D OpenMP/MPI Wave Equation Solver
Our wave equation solver is put into production...

CPU

A breakdown of the 88.5% CPU time:

Single-core code 100.0%
Scalar numeric ops 22.4%
Vector numeric ops 0.0%
Memory accesses 77.6%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.
No time is spent in vectorized instructions. Check the compiler’s vectorization advice to see why key loops could not be vectorized.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage 49.7 MB
Peak process memory usage 53.6 MB
Peak node memory usage 24.0%

The peak node memory usage is very low. You may be able to reduce the amount of allocation time used by running with fewer MPI processes and more data on each process.