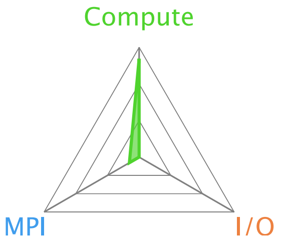
Command:	/nfs/18/hna/tests/allinea/oakley/my.tests/wavee
Resources:	1 node (12 physical, 12 logical cores per node)
Memory:	47 GB per node
Tasks:	12 processes
Machine:	n0096.ten.osc.edu
Start time:	Tue Dec 29 14:47:53 2015
Total time:	31 seconds
Full path:	/nfs/18/hna/tests/allinea/oakley/my.tests
Input file:	
Notes:	



Summary: wavee is Compute-bound in this configuration

Compute	89.8%	Time spent running application code. High values are usually good. This is **high**; check the CPU performance section for advice.
MPI	10.3%	Time spent in MPI calls. High values are usually bad. This is **very low**; this code may benefit from a higher process count.
I/O	0.0%	Time spent in filesystem I/O. High values are usually bad. This is **negligible**; there's no need to investigate I/O performance.

This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

As very little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 89.8% CPU time:

Scalar numeric ops	24.6%
Vector numeric ops	0.0%
Memory accesses	75.4%

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.

MPI

A breakdown of the 10.3% MPI time:

Time in collective calls	5.9%
Time in point-to-point calls	94.1%
Effective process collective rate	80.7 kB/s
Effective process point-to-point rate	1.10 MB/s

Most of the time is spent in point-to-point calls with a very low transfer rate. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

I/O

A breakdown of the 0.0% I/O time:

Time in reads	0.0%
Time in writes	0.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	0.00 bytes/s

No time is spent in I/O operations. There's nothing to optimize here!

Threads

A breakdown of how multiple threads were used:

Computation	0.0%
Synchronization	0.0%
Physical core utilization	99.9%
System load	100.8%

No measurable time is spent in multithreaded code.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	40.7 MB
Peak process memory usage	50.4 MB
Peak node memory usage	4.0%

The peak node memory usage is very low. Running with fewer MPI processes and more data on each process may be more efficient.