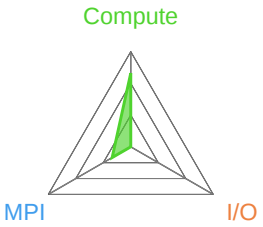
Command:	aprun -n 64 ./wave_cray
Resources:	2 nodes (32 physical, 64 logical cores per node)
Memory:	63 GiB per node
Tasks:	64 processes
Machine:	beskow-login1.pdc.kth.se
Start time:	Thu Sep 8 14:57:38 2016
Total time:	31 seconds
Full path:	/cfs/klemming/scratch/c/cira/students/allinea-reports-6.1



Summary: wave_cray is Compute-bound in this configuration

Compute	77.4%	Time spent running application code. High values are usually good. This is **high**; check the CPU performance section for advice.
MPI	22.6%	Time spent in MPI calls. High values are usually bad. This is **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 little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 77.4% CPU time:

Scalar numeric ops	5.3%
Vector numeric ops	14.7%
Memory accesses	80.0%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

MPI

A breakdown of the 22.6% MPI time:

Time in collective calls	2.8%
Time in point-to-point calls	97.2%
Effective process collective rate	1.13 MB/s
Effective process point-to-point rate	2.64 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	99.3%
System load	100.0%

No measurable time is spent in multithreaded code.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	97.2 MiB
Peak process memory usage	108 MiB
Peak node memory usage	11.0%

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