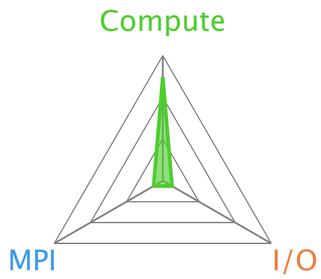
arm PERFORMANCE REPORTS

Command:	aprun -b -j 1 -n 16 python DG_advection.py
Resources:	1 node (64 physical, 256 logical cores per node)
Memory:	252 GiB per node
Tasks:	16 processes
Machine:	xcimom
Start time:	Sat Aug 7 12:00:19 2021
Total time:	141 seconds (about 2 minutes)
Full path:	/lustre/home/ex-dmacreman/firedrake/firedrake/bin



Summary: python is Compute-bound in this configuration

Compute	83.0%	Time spent running application code. High values are usually good. This is **high**; check the CPU performance section for advice
MPI	8.5%	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	8.6%	Time spent in filesystem I/O. High values are usually bad. This is **low**; check the I/O breakdown section for optimization advice

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

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

CPU Metrics

Linux perf event metrics:

Cycles per instruction	0.86
L2D cache miss	36.8%
Stalled backend cycles	25.4%
Stalled frontend cycles	15.1%

Cycles per instruction is low, which is good. Vectorization allows multiple instructions per clock cycle.

MPI

A breakdown of the 8.5% MPI time:

Time in collective calls	51.1%
Time in point-to-point calls	48.9%
Effective process collective rate	525 kB/s
Effective process point-to-point rate	1.51 MB/s

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

The point-to-point transfer rate is very low. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

I/O

A breakdown of the 8.6% I/O time:

Time in reads	92.6%
Time in writes	7.4%
Effective process read rate	1.98 MB/s
Effective process write rate	134 kB/s

Most of the time is spent in read operations with a very low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

Threads

A breakdown of how multiple threads were used:

Computation	0.0%
Synchronization	0.0%
Physical core utilization	23.0%
System load	23.6%

No measurable time is spent in multithreaded code.

Physical core utilization is low. Try increasing the number of processes to improve performance.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	268 MiB
Peak process memory usage	396 MiB
Peak node memory usage	3.0%

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

Energy

A breakdown of how energy was used:

CPU	not supported %
System	not supported %
Mean node power	not supported W
Peak node power	0.00 W

Energy metrics are not available on this system.