An I/O analysis of HPC workloads on CephFS and Lustre

Alberto Chiusole – eXact lab, dssc.units.it
Stefano Cozzini – CNR-IOM, eXact lab
D. Van Der Ster, M. Lamanna – CERN
G. Giuliani – ICTP

HPC-IODC, Frankfurt, June 20, 2019

Agenda

- Goals of the project
- Introduction
  - Overview of HPC production infrastructures: CERN & C3HPC
  - RegCM application
- Activities performed:
  - RegCM performance analysis
  - IOR tests & MPI-IO hints analysis
  - Lazy I/O on experimental CephFS: CNR-IOM & Pawsey
- Conclusions

Goals of the project

- Assessing the capability of CephFS on HPC workloads:
  - RegCM, climate science HPC application
  - IOR benchmark
- Methodology:
  - Compare CephFS with a common storage solution for HPC, such as Lustre

Storage infrastructures

- Ceph at CERN, Geneva, Switzerland:
  - Version 13.2.5 “Mimic”
  - 402 OSDs on 134 hosts: 3 SSDs on each host
  - Replica 2
  - 10 Gbit Ethernet between storage nodes
  - 4xFDR (64 Gbit) InfiniBand between computing nodes
  - Max 32 client computing nodes used, 20 procs each (max 640 processors)

Storage infrastructures

- Lustre at C3HPC, Trieste, Italy:
  - 2 I/O servers, 4 OSTs each, HDD
  - RAID 6
  - 1xQDR (8 Gbit) between storage and nodes
  - Max 8 client computing nodes used, 24 procs each (but only 20 procs requested, max 160 processors)

RegCM in a nutshell

- HPC benchmark reference application: RegCM (Regional Climate Model) by ICTP
- Largely adopted application in climate science
- Simulates decades of climate evolution:
  - Evolve a 3D grid of initial conditions for a period of time
  - Every "simulation hour" stores simulated data to file
  - Save checkpoint for resuming, at every "simulation day"

RegCM software stack

- Fortran90 + MPI library
- I/O library: NetCDF
- Parallel I/O:
  - HDF5 (default)
  - PnetCDF (recently implemented)

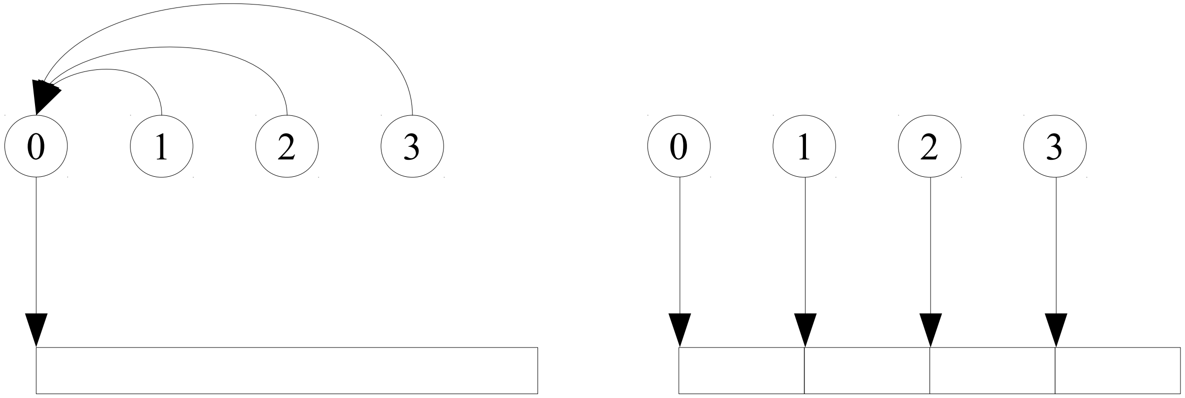
Compiler	Intel 18.0.3	
MPI layer	Intel-MPI 2018.3.222	OpenMPI 3.1.2
Parallel I/O	HDF5 1.10.4	PnetCDF 1.11.0

RegCM software stack

- In total 4 different combinations benchmarked:
  - IH: Intel-MPI + HDF5
  - IP: Intel-MPI + PnetCDF
  - OH: OpenMPI + HDF5
  - OP: OpenMPI + PnetCDF

RegCM I/O approach

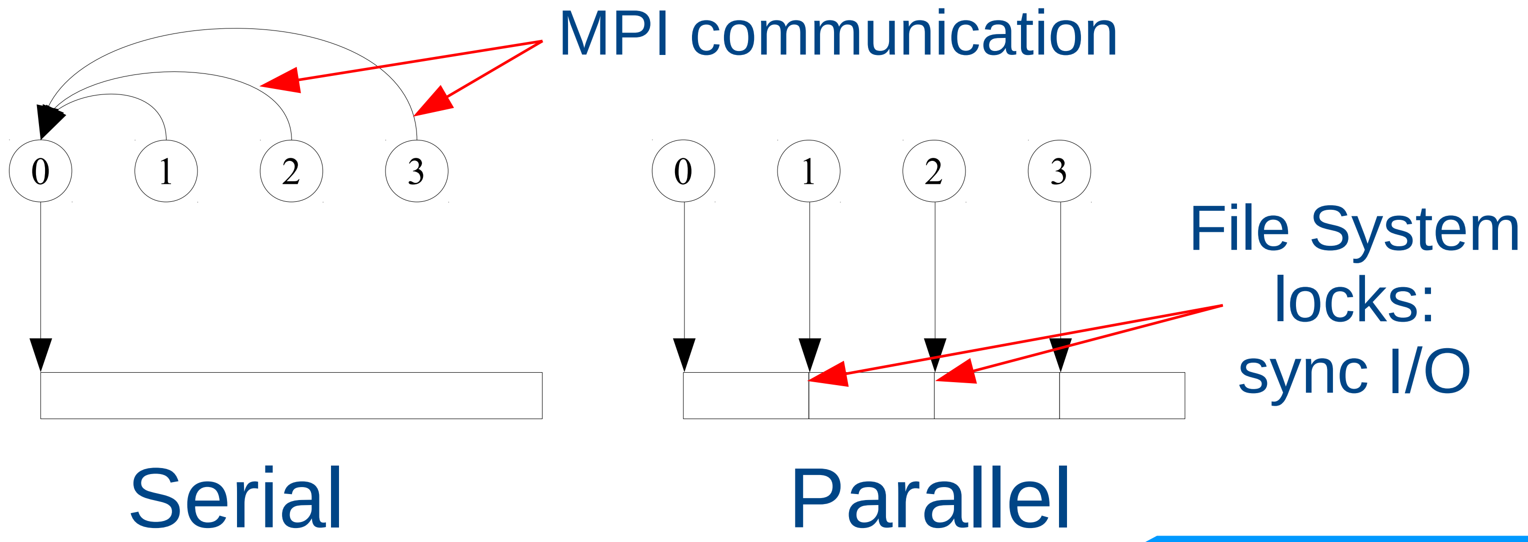
- 2 writing modes implemented in the application to write to a single file:
  - Serial/Spokesperson: all processes send to single writer
  - Parallel: every process writes at a specific offset



Serial Parallel

RegCM I/O approach

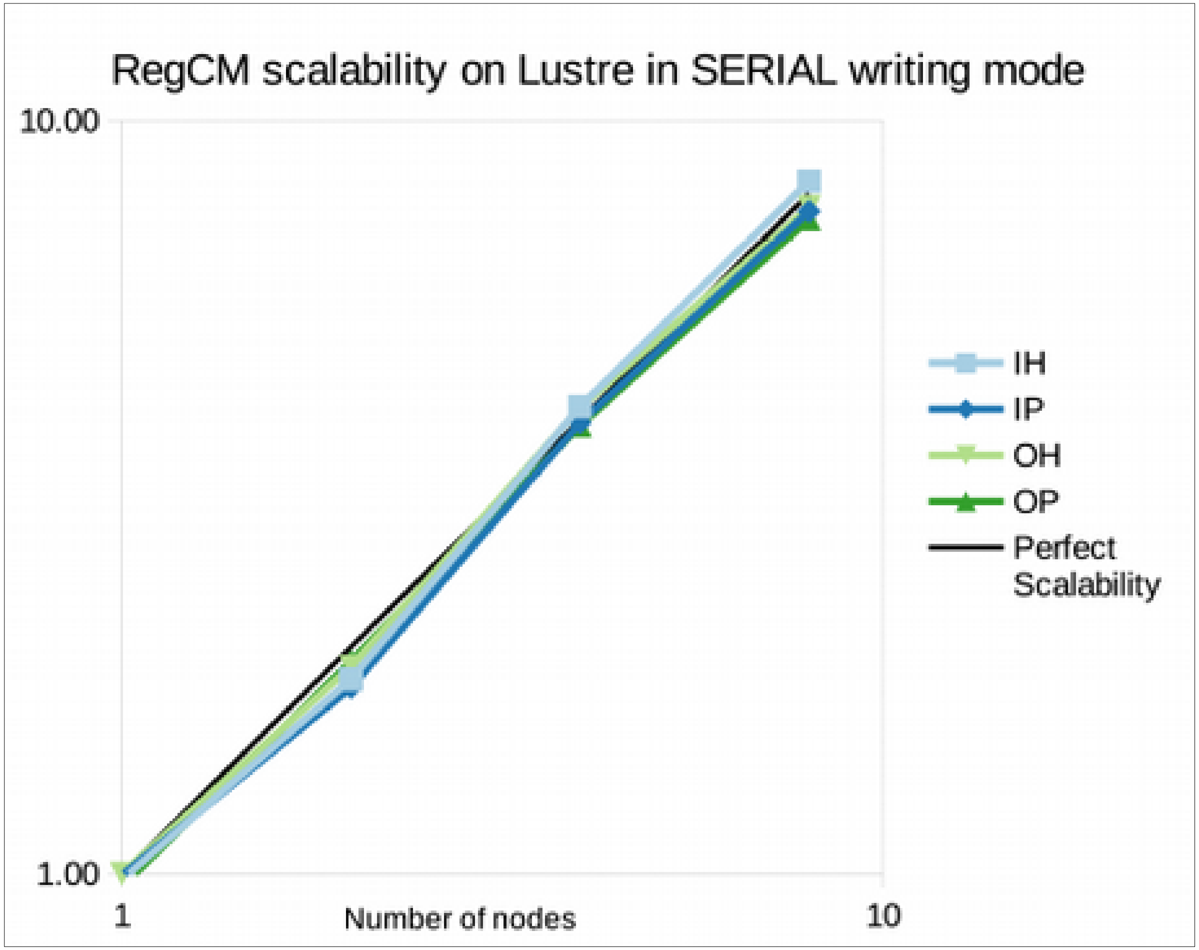
- 2 writing modes implemented in the application to write to a single file:
  - Serial/Spokesperson: all processes send to single writer
  - Parallel: every process writes at a specific offset



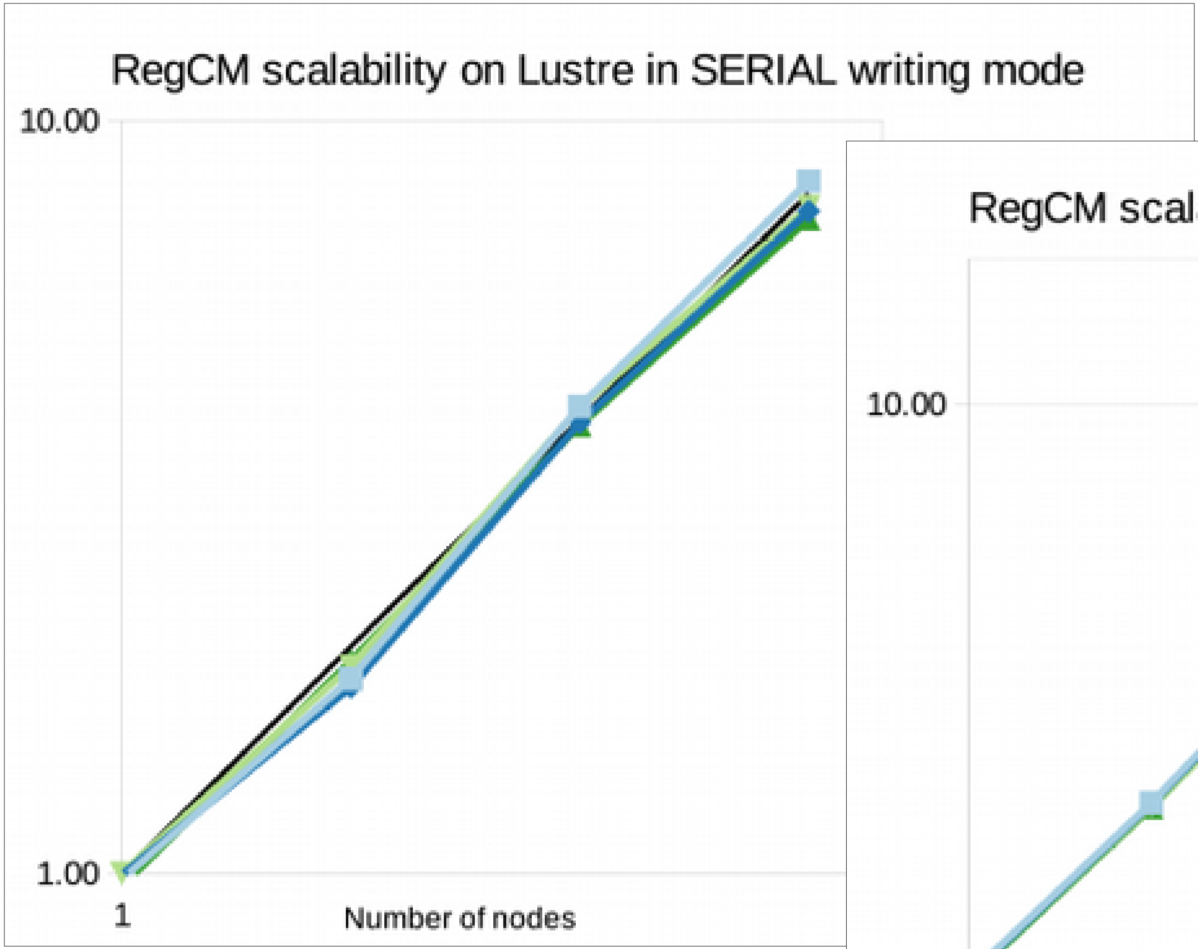
exact

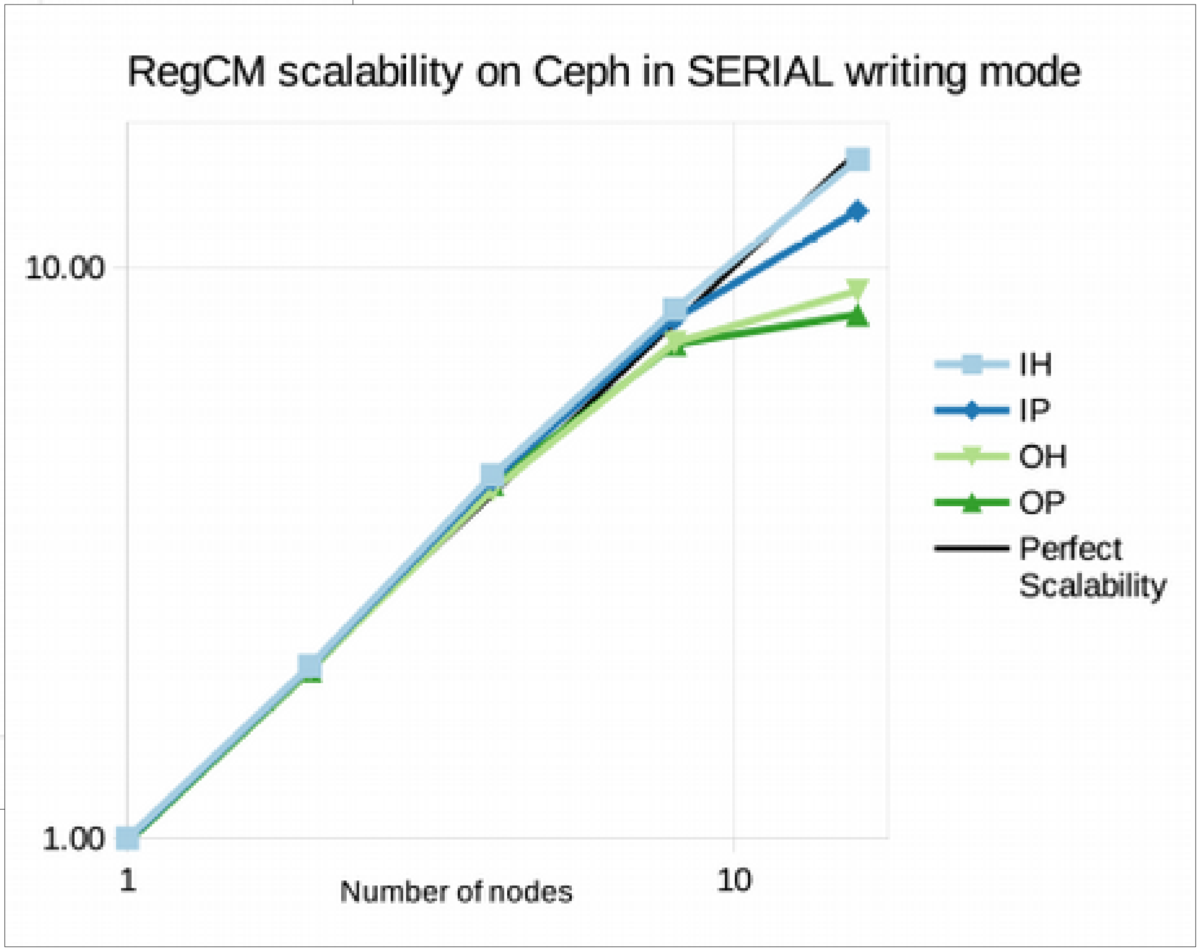
RegCM scalability: Serial I/O

RegCM scalability: Serial I/O

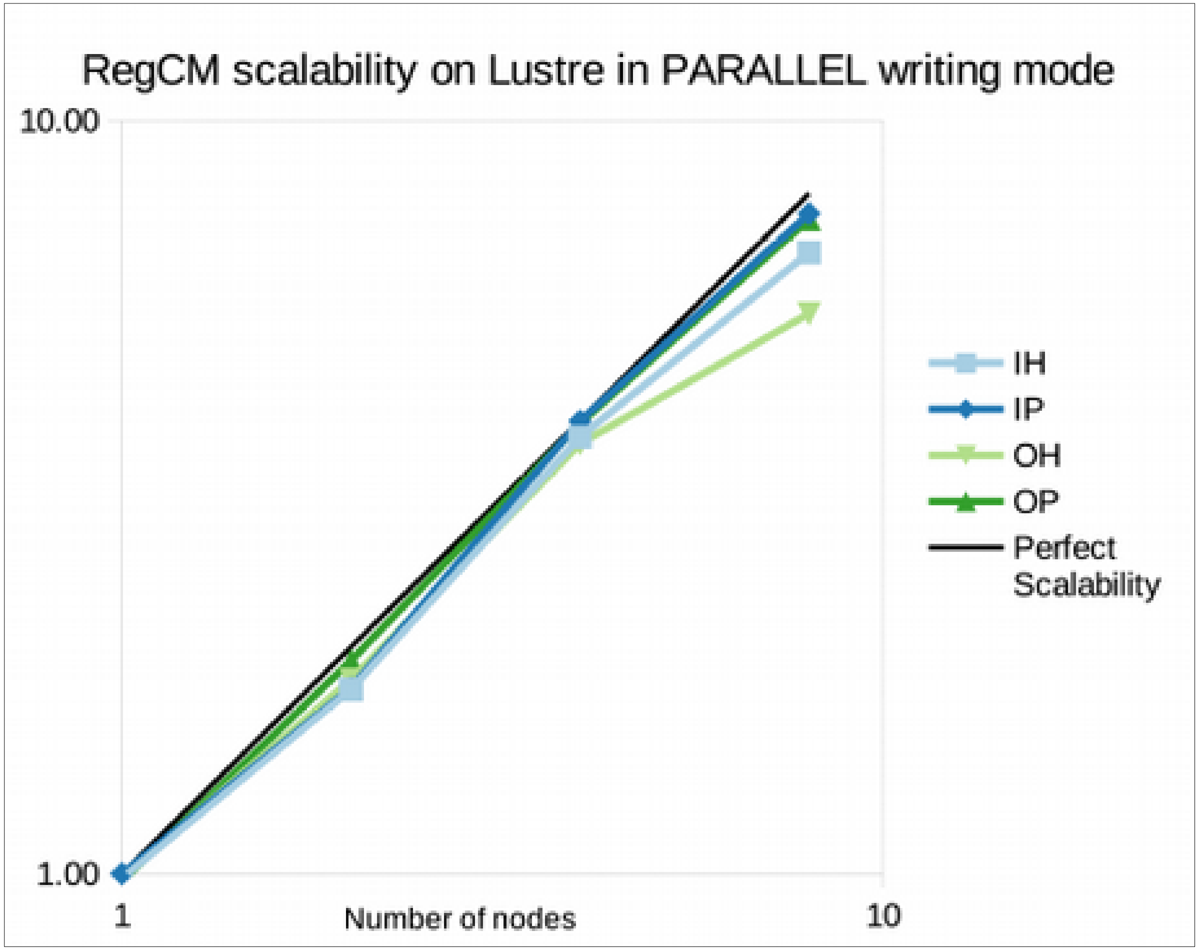


RegCM scalability: Serial I/O

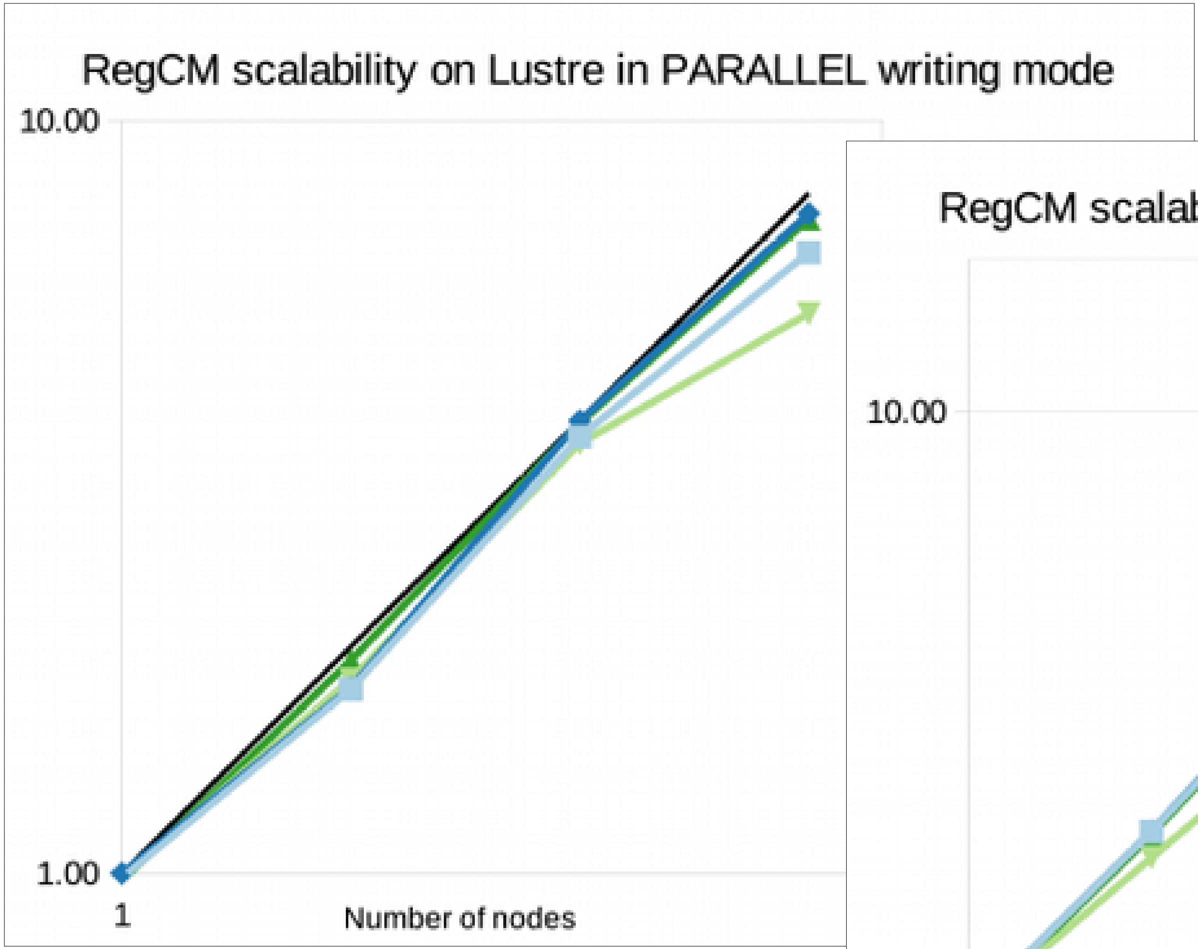


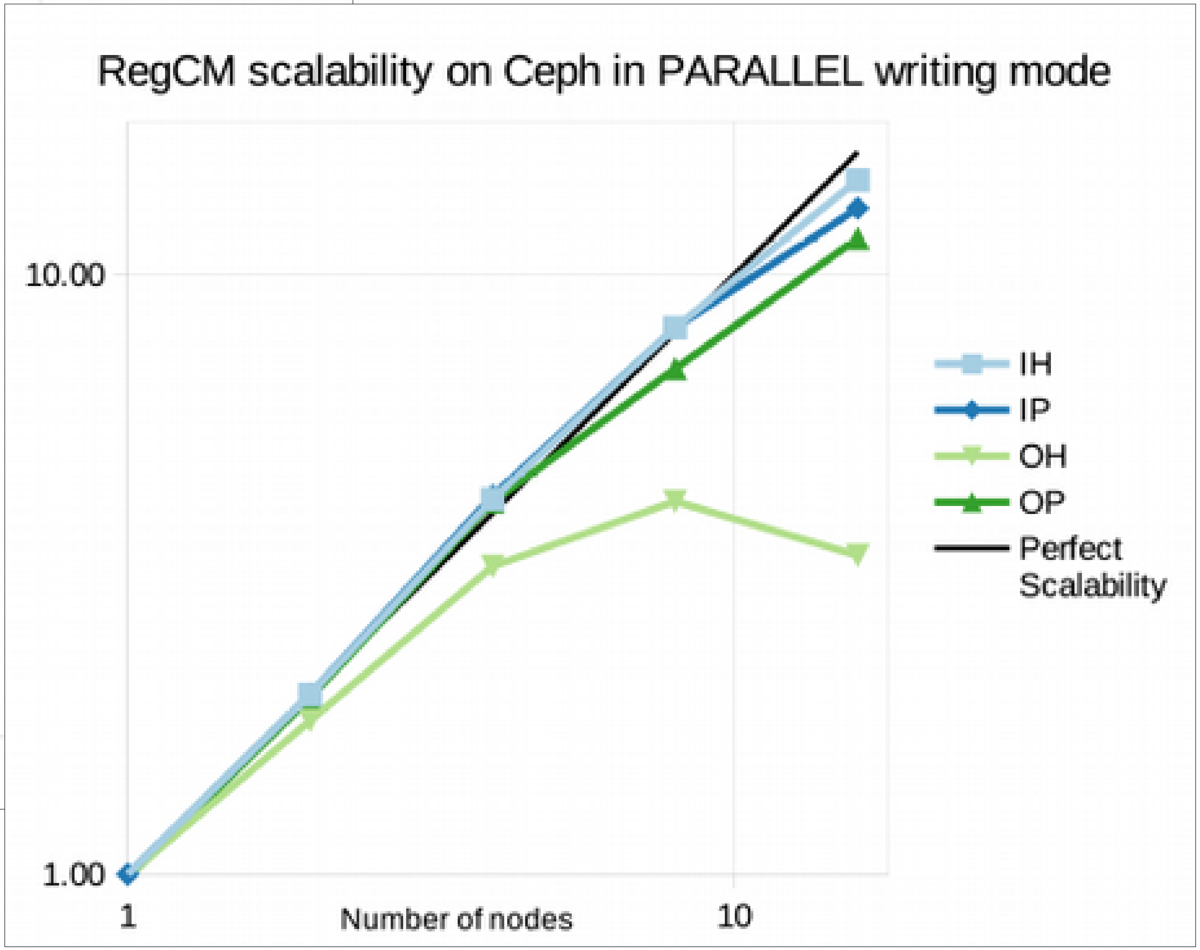


RegCM scalability: Parallel I/O



RegCM scalability: Parallel I/O





RegCM I/O: main results

- Scalability fine for both CephFS and Lustre
- Pathological behavior on OpenMPI with >=320 processors
- No big difference between serial and parallel writing mode on both archs
- Slight difference between PnetCDF and HDF5
  - PnetCDF better on Lustre
  - HDF5 better on CephFS

IOR benchmarking analysis

- Investigate performance with a standard benchmarking tool such as IOR
  - An in-depth further analysis to understand the underlying details of our software stack
- IOR capabilities:
  - Simulate several I/O patterns: MPI-IO, HDF5, PnetCDF
  - Pass MPI-IO hints at runtime

IOR benchmarking analysis

- Permutations of MPI-IO hints: Collective Buffering (CB) and Data Sieving (DS)
  - romio_cb_write + romio_cb_read
  - romio_ds_write + romio_ds_read

IOR benchmarking analysis

- Permutations of MPI-IO hints: Collective Buffering (CB) and Data Sieving (DS)
  - romio_cb_write + romio_cb_read
  - romio_ds_write + romio_ds_read
- CB or “2-phase I/O”
  - M "aggregator" proxying to FS (M«N procs)
- DS
  - R/W non-contiguous regions as contiguous

IOR benchmarking analysis

Config	cb_write	ds_write	cb_read	ds_read
1	enable	enable	enable	enable
2	enable	disable	enable	disable
3	enable	auto	enable	auto
4	disable	enable	disable	enable
5	disable	disable	disable	disable
6	disable	auto	disable	auto
7	auto	enable	auto	enable
8	auto	disable	auto	disable
9	auto	auto	auto	auto

Every MPI hint by default is set to 'auto'

IOR benchmarking analysis

Config	cb_write	ds_write	cb_read	ds_read
1	enable	enable	enable	enable
2	enable	disable	enable	disable
3	enable	auto	enable	auto
4	disable	enable	disable	enable
5	disable	disable	disable	disable
6	disable	auto	disable	auto
7	auto	enable	auto	enable
8	auto	disable	auto	disable
9	auto	auto	auto	auto

} Best I/O perf on CephFS

Every MPI hint by default is set to 'auto'

exact

IOR tests performed

- 2 different tests with IOR:
  - 20 GiB test file ~ 1 day-worth of RegCM I/O
  - 200 GiB
- Each processor writes a portion of file
  - From 1 node (20 procs) to 32 nodes (640 procs)
- 640 procs on 20 GiB: each process writes only 32 MiB
  - simulating an HPC run, not interested in throughput
- Each test executed 3 times

IOR benchmarks CephFS

Procs (nodes)		160 (8)	320 (16)	640 (32)
		Max MiB/s	Max MiB/s	Max MiB/s
MPIIO	Hint 4	3,378.77	3,763.28	3,130.25
	Hint 5	3,351.80	3,527.81	3,163.44
	Hint 6	3,438.99	3,993.69	3,195.81
	Hint 7	1,126.31	1,083.58	1,387.34
	Hint 8	1,179.27	872.02	1,368.01
	Hint 9	**1,178.31**	**849.05**	**1,284.46**
HDF5	Hint 4	816.07	1,206.80	1,582.53
	Hint 5	1,052.07	1,258.99	1,656.83
	Hint 6	854.59	1,167.77	1,697.55
	Hint 7	865.17	1,240.44	1,347.02
	Hint 8	1,136.30	1,330.69	1,543.40
	Hint 9	**1,036.81**	**1,245.88**	**1,662.22**
PNETCDF	Hint 4	919.87	1,182.61	1,004.86
	Hint 5	1,045.32	1,295.07	1,555.09
	Hint 6	1,158.90	1,235.74	1,598.70
	Hint 7	1,150.99	1,129.65	1,716.50
	Hint 8	1,143.79	1,199.30	1,635.51
	Hint 9	**1,115.57**	**1,229.15**	**1,688.75**

20 GiB test file
(~ 1 day of RegCM sim.)

Procs (nodes)		160 (8)	320 (16)	640 (32)
		Max MiB/s	Max MiB/s	Max MiB/s
MPIIO	Hint 4	2,301.30	2,413.02	3,715.37
	Hint 5	2,278.89	2,402.44	3,722.50
	Hint 6	2,168.65	2,407.73	3,761.83
	Hint 7	511.34	667.44	1,490.77
	Hint 8	484.44	666.22	1,480.85
	Hint 9	**468.73**	**658.90**	**1,489.82**
HDF5	Hint 4	770.20	747.12	1,554.31
	Hint 5	1,001.40	776.99	1,666.32
	Hint 6	722.60	746.89	1,539.55
	Hint 7	487.86	748.90	1,540.02
	Hint 8	1,041.18	773.80	1,643.31
	Hint 9	**758.98**	**744.98**	**1,573.00**
PNETCDF	Hint 4	1,051.58	782.48	1,595.43
	Hint 5	1,011.18	781.86	1,658.13
	Hint 6	1,002.34	787.97	1,541.55
	Hint 7	1,004.47	806.74	1,682.22
	Hint 8	1,005.13	796.40	1,567.61
	Hint 9	**998.27**	**786.41**	**1,498.40**

200 GiB test file

IOR benchmarks Lustre

Procs (nodes)		80 (4)	160 (8)
		Max MiB/s	Max MiB/s
MPIIO	Hint 4	289.27	196.05
	Hint 5	330.48	215.76
	Hint 6	290.73	247.73
	Hint 7	308.47	221.06
	Hint 8	183.08	227.24
	Hint 9	**289.26**	**236.93**
HDF5	Hint 4	287.57	201.68
	Hint 5	249.26	200.19
	Hint 6	283.03	210.35
	Hint 7	229.19	183.50
	Hint 8	223.25	219.11
	Hint 9	**250.36**	**204.23**
PNETCDF	Hint 4	265.47	234.77
	Hint 5	253.04	227.05
	Hint 6	314.74	221.21
	Hint 7	299.37	246.28
	Hint 8	223.45	244.00
	Hint 9	**282.19**	**251.31**

20 GiB test file
(~ 1 day of RegCM sim.)

Procs (nodes)		80 (4)	160 (8)
		Max MiB/s	Max MiB/s
MPIIO	Hint 4	275.29	236.03
	Hint 5	251.81	220.25
	Hint 6	310.16	247.23
	Hint 7	281.74	228.86
	Hint 8	279.86	238.95
	Hint 9	**249.94**	**217.35**
HDF5	Hint 4	227.71	221.82
	Hint 5	282.96	228.63
	Hint 6	271.99	211.22
	Hint 7	251.56	212.68
	Hint 8	255.89	236.18
	Hint 9	**272.25**	**224.26**
PNETCDF	Hint 4	258.78	224.59
	Hint 5	270.34	207.63
	Hint 6	277.76	213.95
	Hint 7	246.87	210.36
	Hint 8	270.64	220.18
	Hint 9	**270.75**	**194.13**

200 GiB test file

Summary of IOR results

- On both FS: enabling CB degrades perf (hints 1, 2, 3) for all software stacks; DS produces no changes
- On CephFS:
  - Disabling CB, hints 4, 5, 6 improves MPI-IO perf notably
  - No significant difference in HDF5 and PnetCDF for any hint
- On Lustre:
  - No significant difference for any hint combination on all software stacks

Lazy I/O on CephFS

- Multiple processes open file <u>writing</u> mode:
  - FS serializes writes to avoid data inconsistency
- Often scientific applications need to dump a matrix to disk
  - Embarrassingly parallel task
  - Perfect for async calls/buffers on writes
- CephFS provides Lazy I/O, since v.14
  - Proposed POSIX standard extension
  - Removes "consistency locks"

CephFS Lazy I/O testbeds

- CNR-IOM (Trieste, IT):
  - 2 storage nodes with:
    - 1 Intel E5-2620 v4 CPU
    - 2 x 2 TB Samsung NVMe
    - 1 Gbit Ethernet (next experiments on InfiniBand)
  - 2 client nodes with 20 procs each
- Pawsey Supercomputing Center (Perth, AU):
  - 6 storage nodes also acting as clients (hyperconverged):
    - AMD EPYC 7351 CPU (16 physical cores)
    - 1 Intel P4600 NVMe
    - 100 Gbps InfiniBand connection

Lazy I/O setup

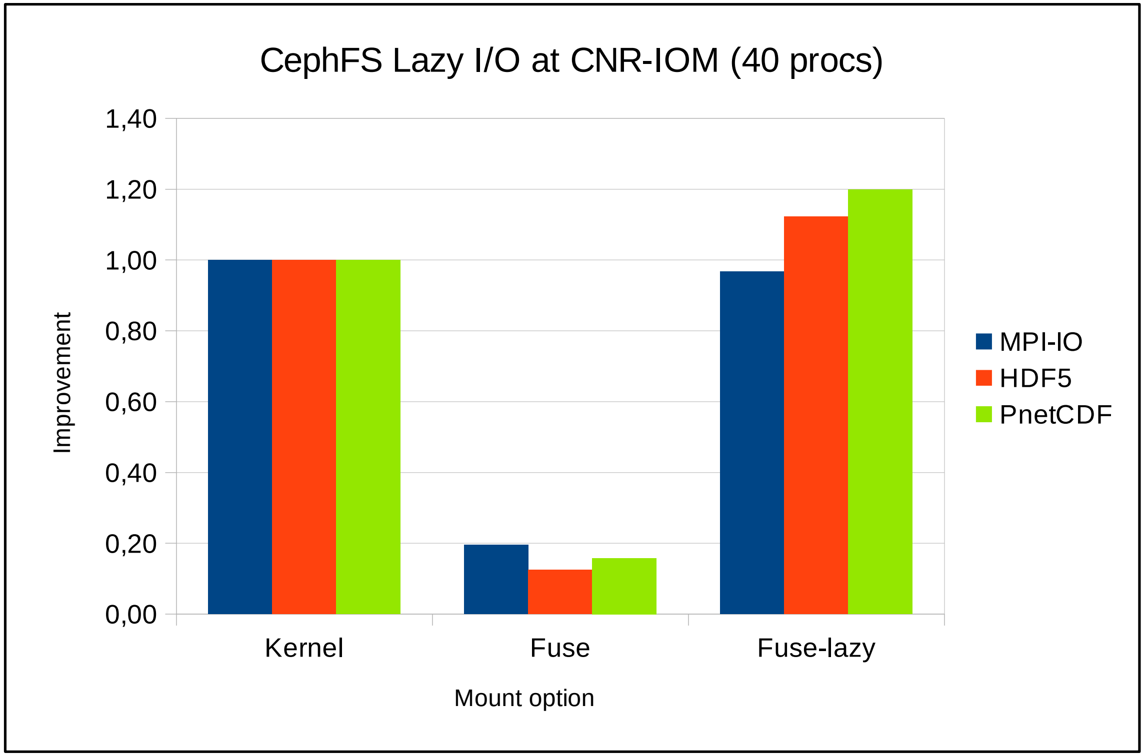
- Both data and metadata on NVMe
- Mounted with 3 different methods:
  - kernel client mount
  - FUSE mount
  - FUSE mount w/ option `client_force_lazyio=true`

Tests performed

- IOR with same software stack as before
- Test file size fixed at 20 GiB for CNR-IOM
- Test file size variable with the number of procs for Pawsey
  - 256 MiB per proc, producing 12 GiB with 96 procs
- Only MPI-IO hint: CB=disabled, others were slower from preliminary tests
  - We wanted to focus on Lazy I/O, not MPI-IO hints

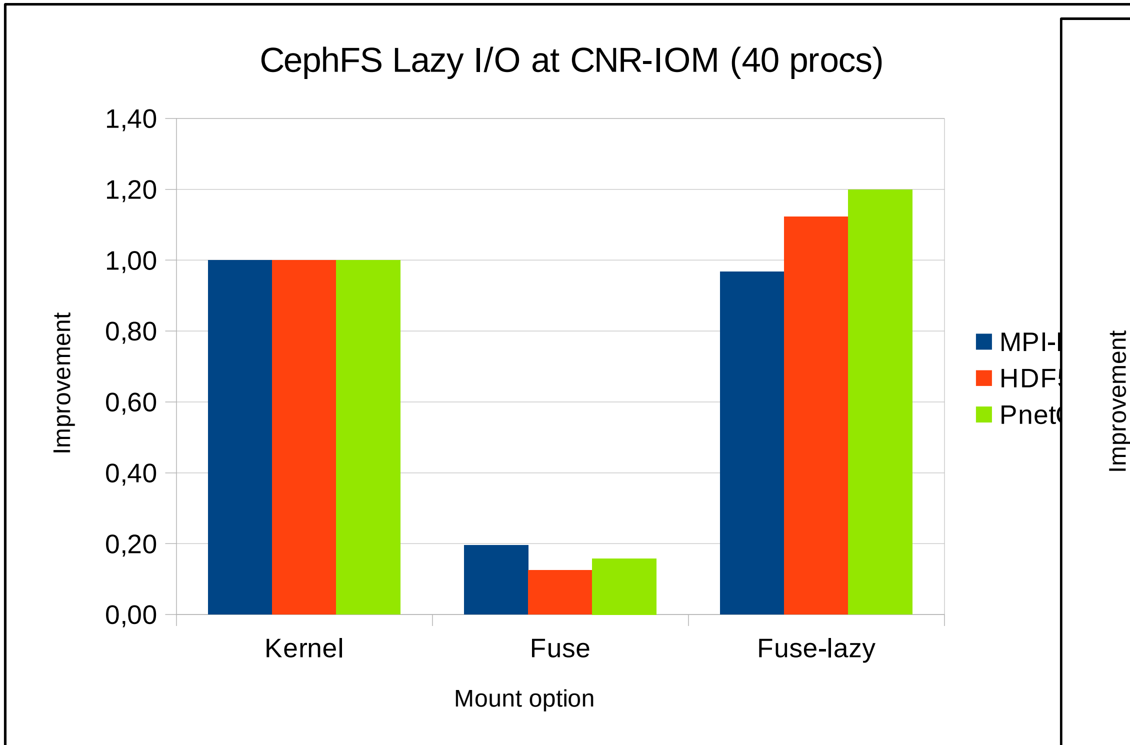
Preliminary results

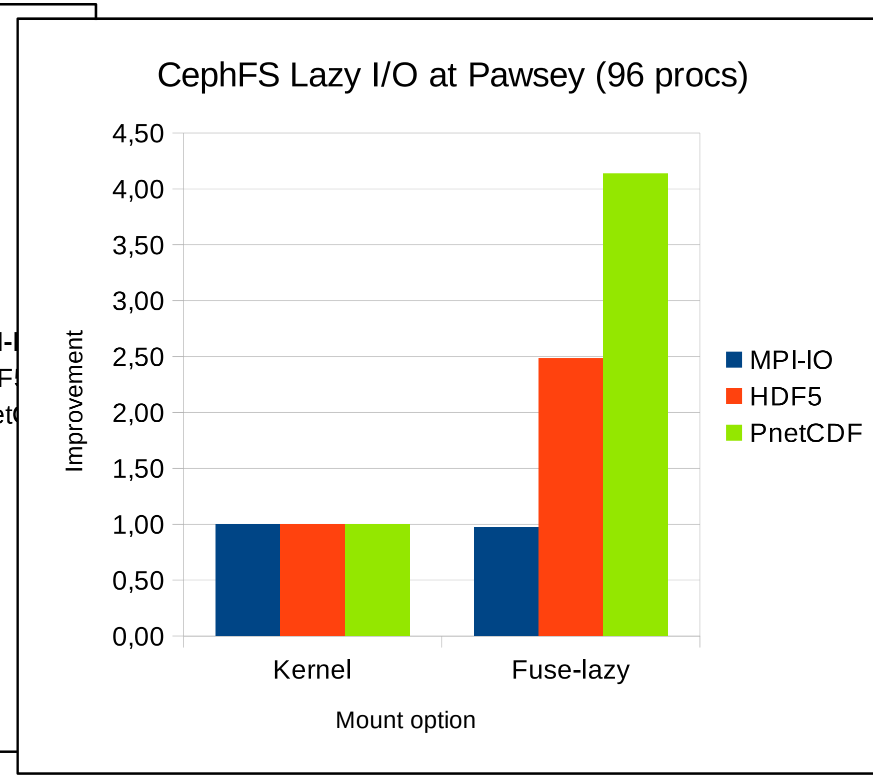
Preliminary results



Behaviour of Fuse and Fuse with forced Lazy I/O against the kernel mount.

Preliminary results





Behaviour of Fuse and Fuse with forced Lazy I/O against the kernel mount.

Lazy I/O observations

- FUSE with forced Lazy I/O: 5x faster than normal FUSE
- We expect to see similar improvements on kernel mount if an application uses the Lazy I/O implementation

Conclusion

- Preliminary results of an ongoing study indicates CephFS is an appropriate FS for HPC workloads
- We already have important suggestions to improve RegCM I/O stack:
  - Fully PnetCDF version available → no more HDF5
- Lazy I/O promising feature for HPC workloads: if available, use in combination with PnetCDF library that seems to scale better
- Work in progress, stay tuned!

Aknowledgments:

Luca Cervigni at Pawsey for collecting data on Lazy I/O on their infrastructure,

Pablo Llopis at CERN for the HPC support.

Thank you for your attention!

Questions?