IO Performance Evaluation on Massive Data Processing at NCI High Performance Computing Platform

Rui Yang, Ben Evans

National Computational Infrastructure (NCI) Australia

NCI – background and key motivation for this work

- NCI is the peak National Computational Centre for Research in Australia
  - ANU Supercomputer Facility (1987-2011)
  - APAC (Australian Partnership for Advanced Computing) (2000-2007)
  - NCI (2007-)
- NCI: move from academic HPC centre to a full national high performance computing centre:
  - Partnership includes ANU, Bureau of Meteorology, CSIRO, Geoscience Australia
  - Particular Focus on Climate, Weather, Earth Systems, Environment, Geophysics & Water Management.
- Two key drivers around the work in this talk
  - HPC Scaling and Optimisation (supported by Fujitsu)
  - High Performance Data analysis
- Application areas:
  - Weather forecasting for the Bureau of Meteorology operations (see Tim Pugh talk)
  - Weather, Climate, Earth systems, water mgt for research (gov & unis) (See Marshall Ward talk)
  - Satellite Earth Observation data with Geoscience Australia and Bureau of Meteorology
  - Geophysics applications for Geoscience Australia , state surveys, and uni's

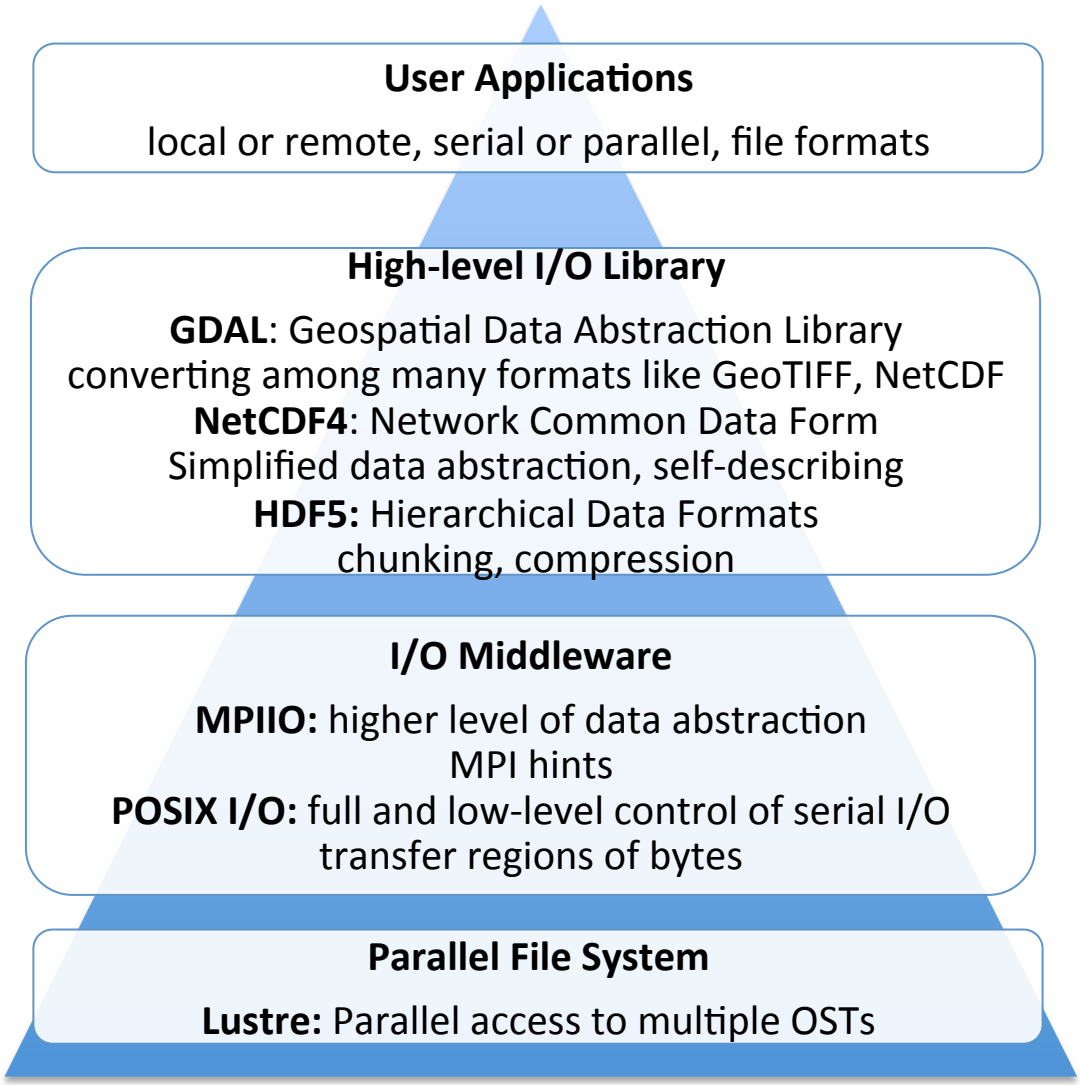
Raijin is a Fujitsu Primergy high-performance, distributed-memory cluster installed in 2013. It comprises:

- 57,472 cores (Intel Xeon Sandy Bridge technology, 2.6 GHz) in 3592 compute nodes
- 160 TBytes of main memory
- Infiniband FDR interconnect
- 10 PBytes of usable fast filesystem (for short-term scratch space).

High Performance Data at NCI

- The National Computational Infrastructure has now co-located a priority set of over 10+ PetaBytes of national data collections for our priority areas.
- The facility provides an integrated high-performance computational and storage platform, or a High Performance Data (HPD) platform, to serve and analyse the massive amounts of data across the spectrum of environmental collections – in particular from the climate, environmental and geoscientific domains.
- The data is managed to support the government agencies, major academic research communities and collaborating overseas organizations.
- By co-locating the vast data collections with high performance computing environments and harmonizing these large valuable data assets, new opportunities have arisen for Data-Intensive interdisciplinary science at scales and resolutions not hitherto possible.
- Note, there are a lot of elements of this that are not touched on in this talk.
- As well as addressing management and performance issues, our work was to also help transform the communities to using better ways of processing, managing and analysing data.

We care about the performant access to data through various levels of IO



IO Tuning activities and interests

Metrics	Serial IO	Parallel IO
User application		
Transfer size	✔	✔
File size	✔	✔
Subset selection	✔	✔
Concurrency	N/A	✔
Access remote DAP server	✔	
IO interfaces		
NetCDF4/HDF5	✔	✔
MPIIO	N/A	✔
POSIX	✔	✔
GDAL/GeoTIFF	✔	
GDAL/NetCDF(4) Classic	✔	

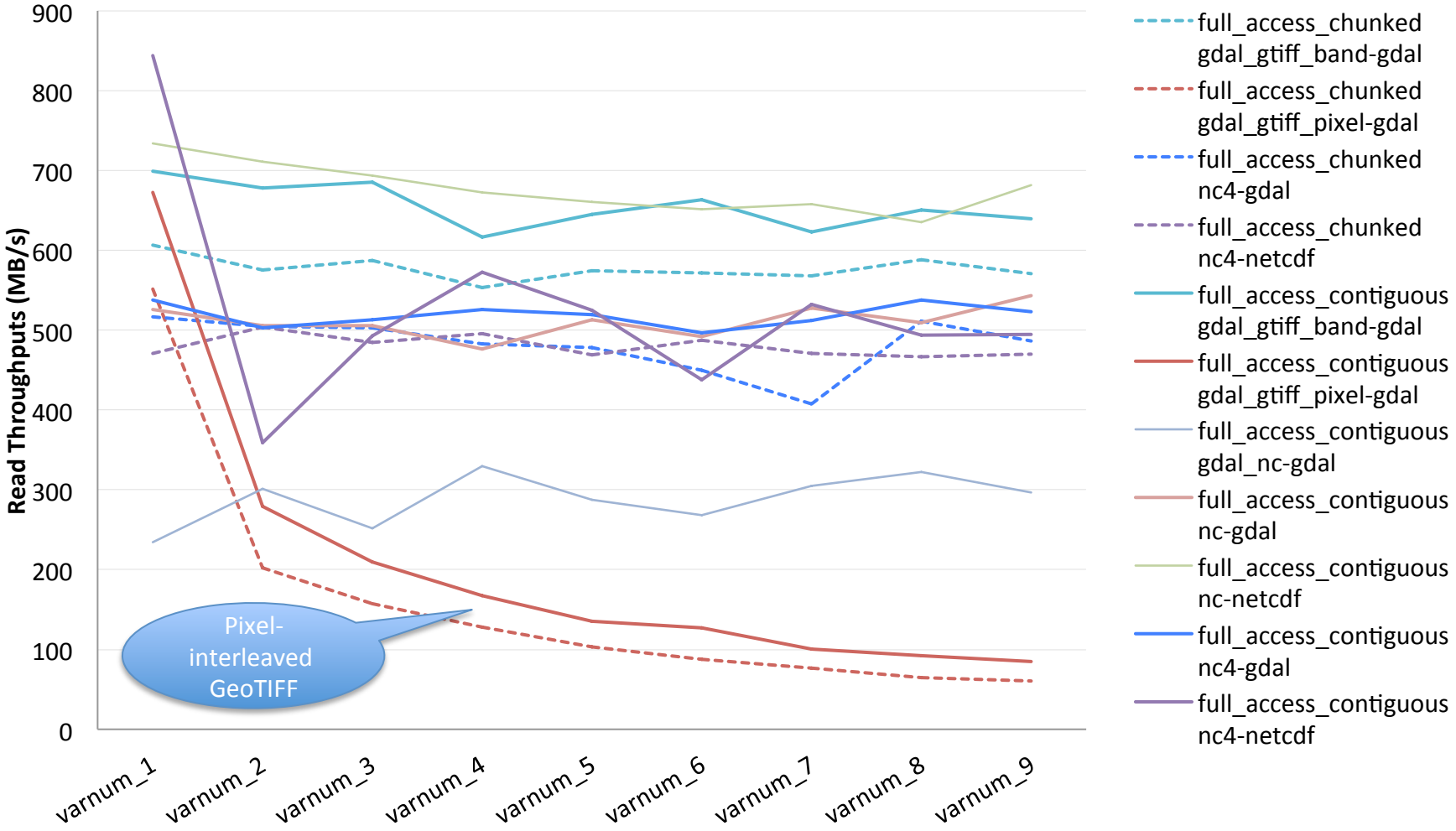
Metrics	Serial IO	Parallel IO
NetCDF/HDF5		
Chunk pattern	✔	✔
Chunk cache	✔	✔
Compression	✔	✔
MPIIO		
Independent & Collective	N/A	✔
Collective buffering	N/A	✔
Data sieving		✔
Lustre file system		
Stripe count	✔	✔
Stripe size	✔	✔
IO profiling & tracing	✔	✔
total	14	16

SERIAL IO

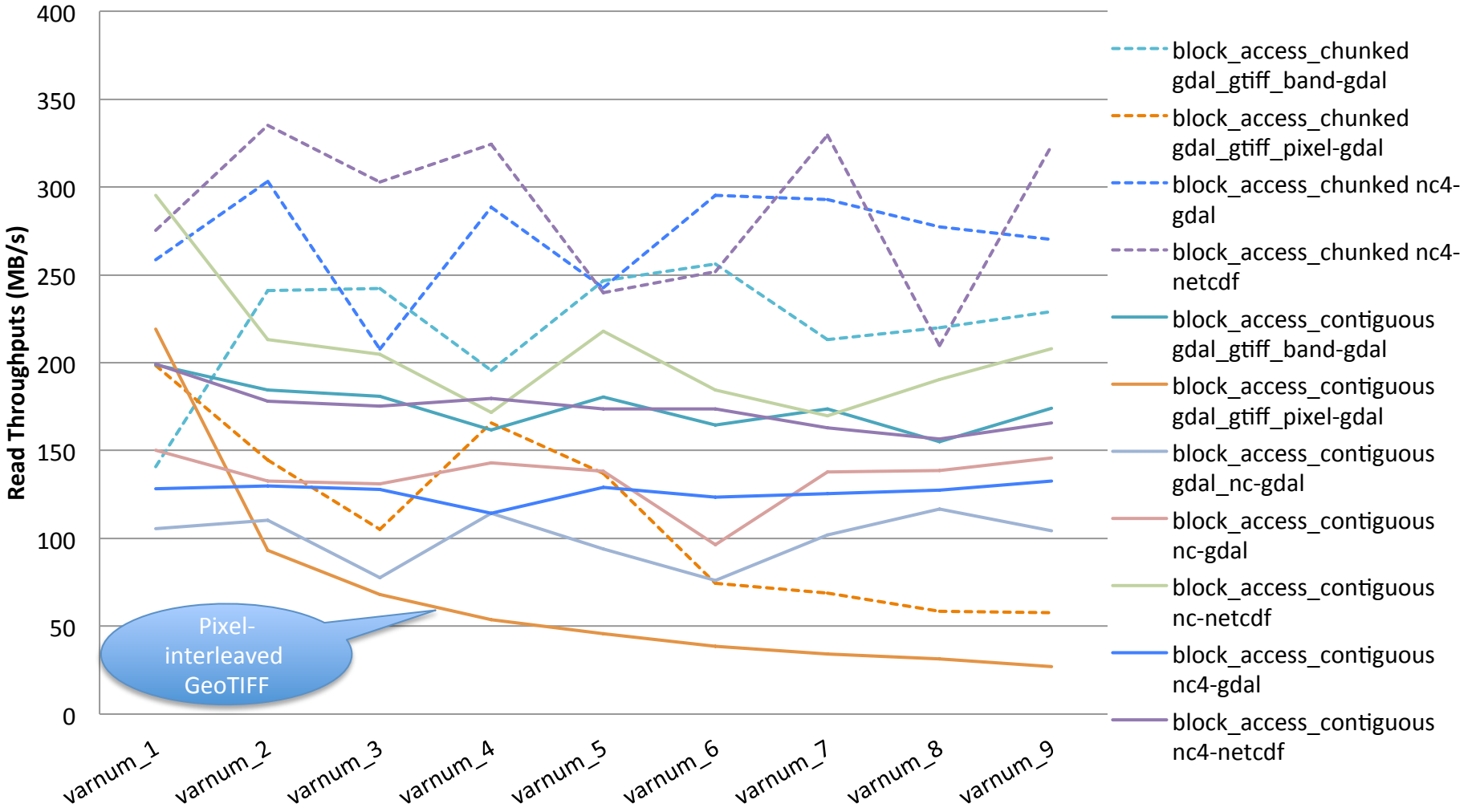
Serial IO: Read benchmark on file format and IO library

File Formats	pixel-interleaved GeoTIFF band-interleaved GeoTIFF NetCDF Classic NetCDF4
Total Bands/Variables	9
Storage Layouts	contiguous, chunked(chunk/tiling size=640×640)
IO Libraries	GDAL, NetCDF
Access patterns	full file access, block access (hyperslab size= 2560×2560)

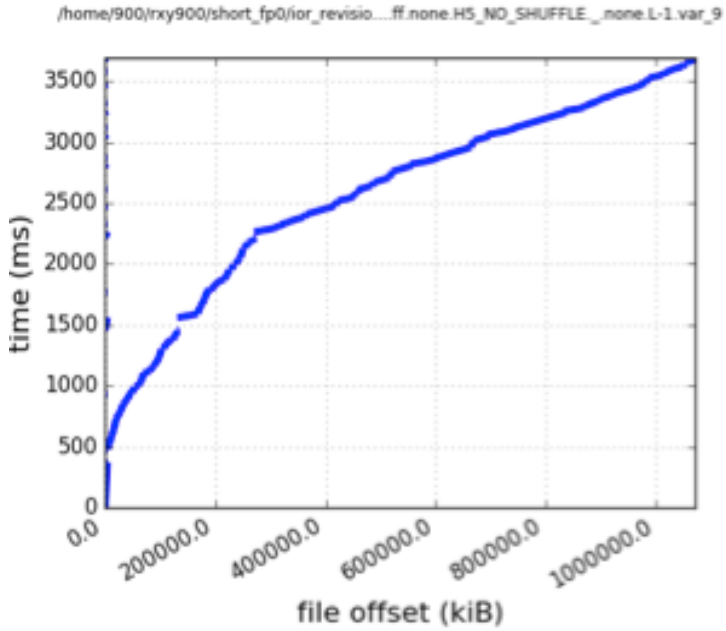
Serial IO: Full file access



Serial IO: Block access

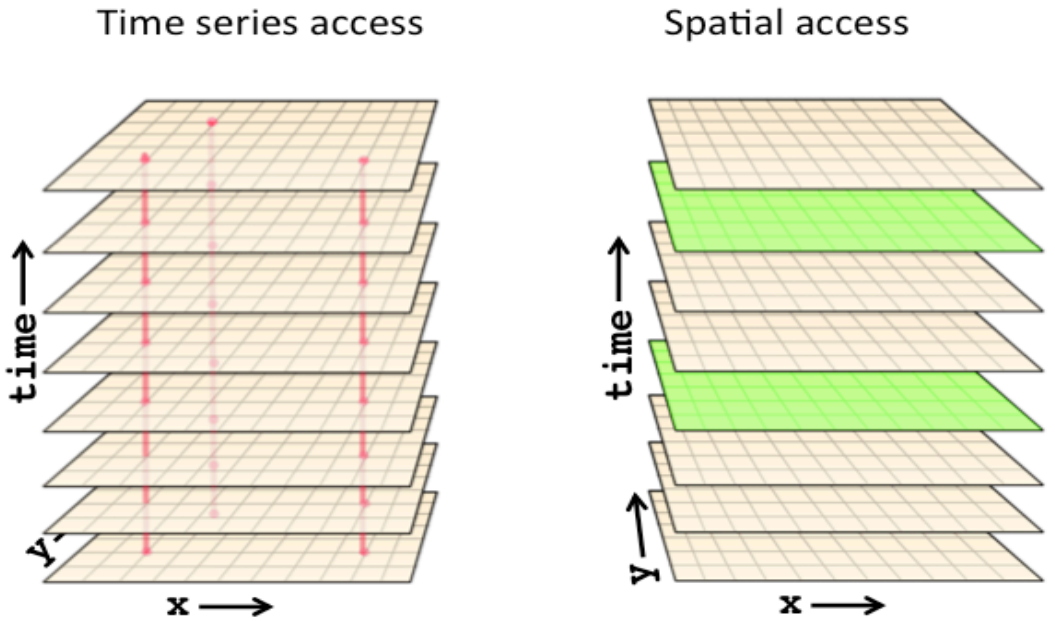


Serial IO: Profiling band-interleaved and pixel-interleaved layout GeoTIFF files

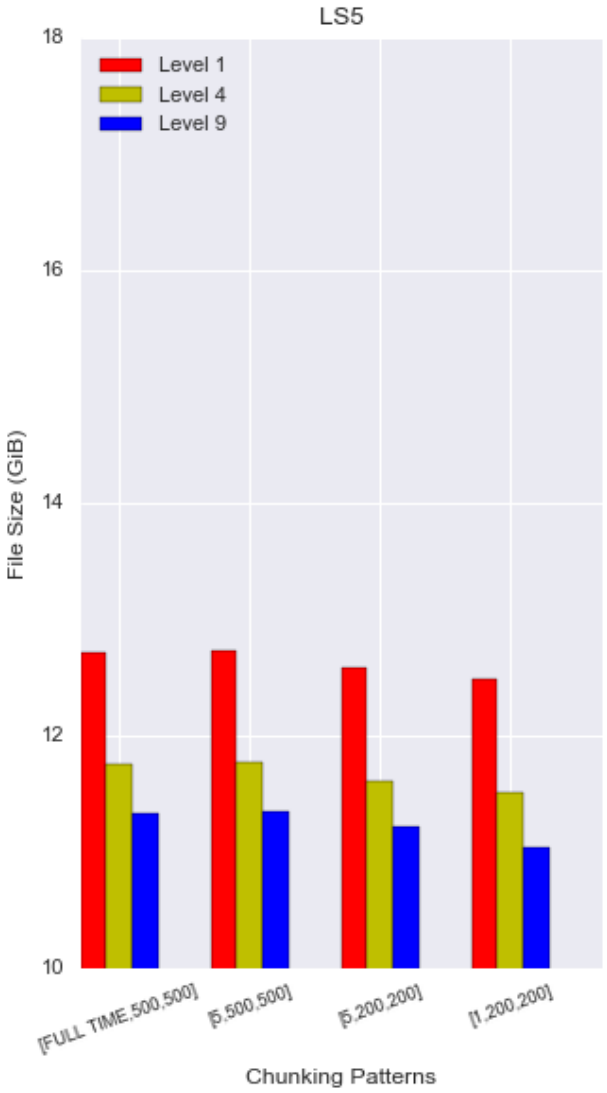
	Band-interleaved	Pixel-interleaved
Read Total	1 variables 9 variables	1 variables 9 variables
Read Time Throughputs	0.273s 446.3MB/s	3.208s 38MB/s
Touch range Touch Size	122 MB 258 MB	1071 MB 1214 MB
Access Patterns		

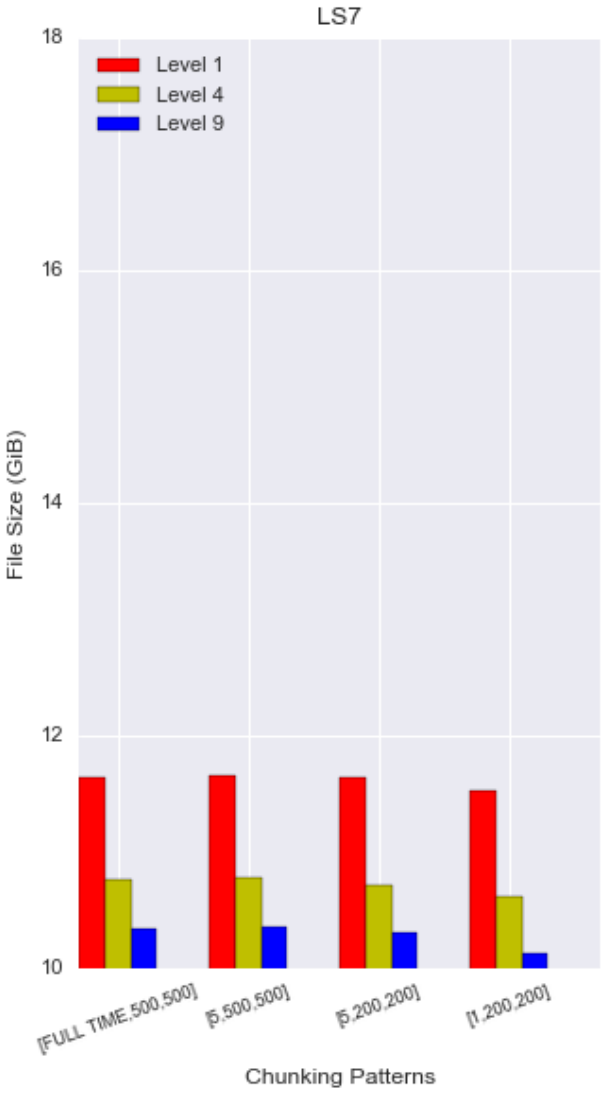
Serial IO: Access pattern, storage layout, chunking and compression of NetCDF

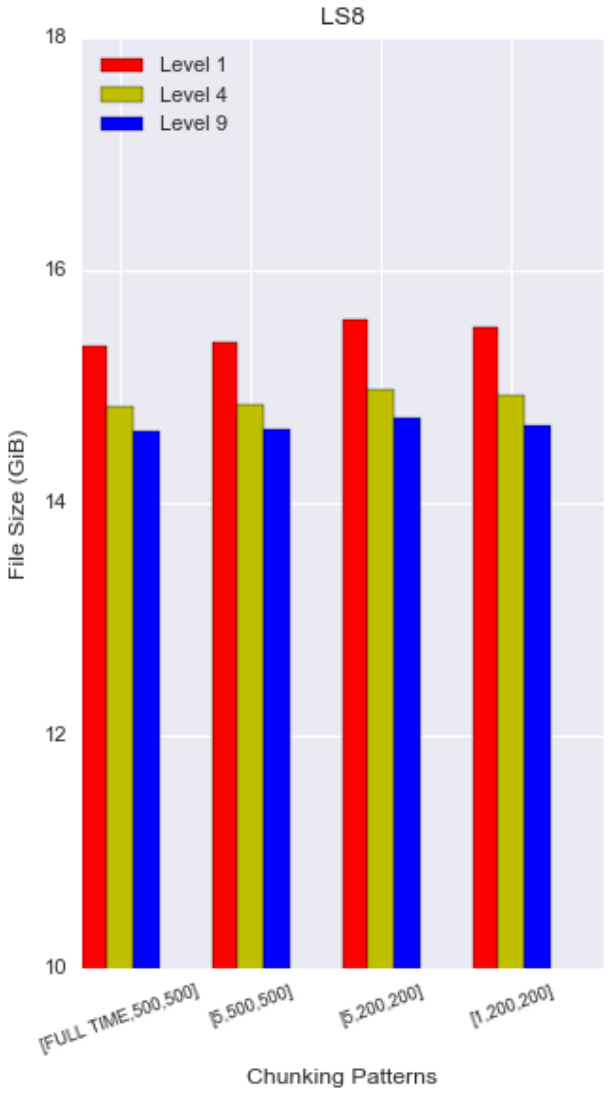
Raw GA NCF	LS5_TM_NBAR,LS7_ETM_NBAR,LS8_OLI_TIRS_NBAR
Variable Size/Band/file	[Full Time. 4000 pixel, 4000 pixel] e.g. 1 year x 100KM x 100KM
Chunking Shapes	[Full Time, 500 pixel, 500 pixel],[5, 500, 500],[5, 200, 200],[1,200,200]
Deflation Levels	1, 4, 9
Data Type	Short
Access patterns	Spatial access: 13 days x 400 KM x 400KM (16 files) Time series access: 49 years x 1 pixel x 1 pixel (49 files)



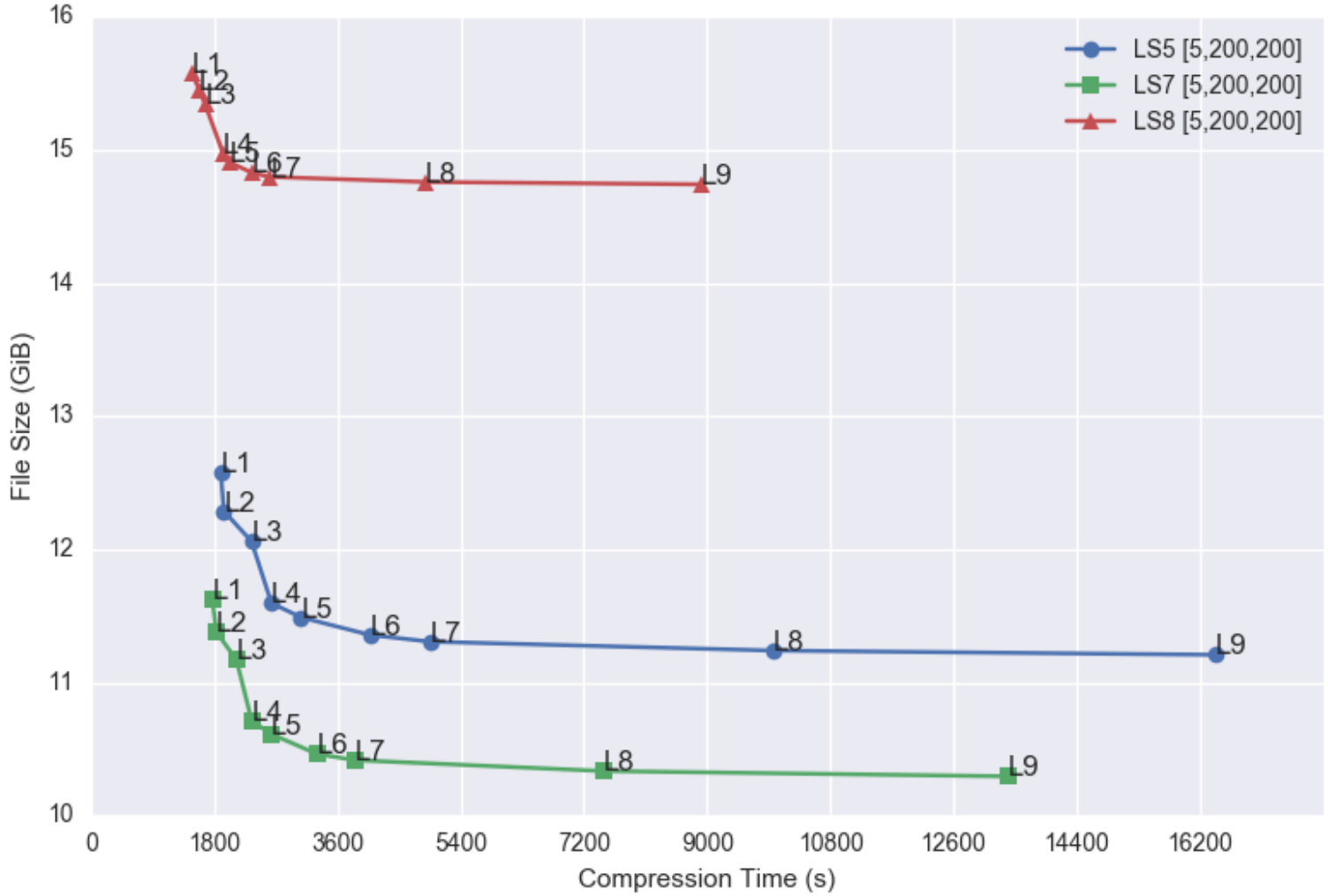
Serial IO: File sizes, chunks and compression levels



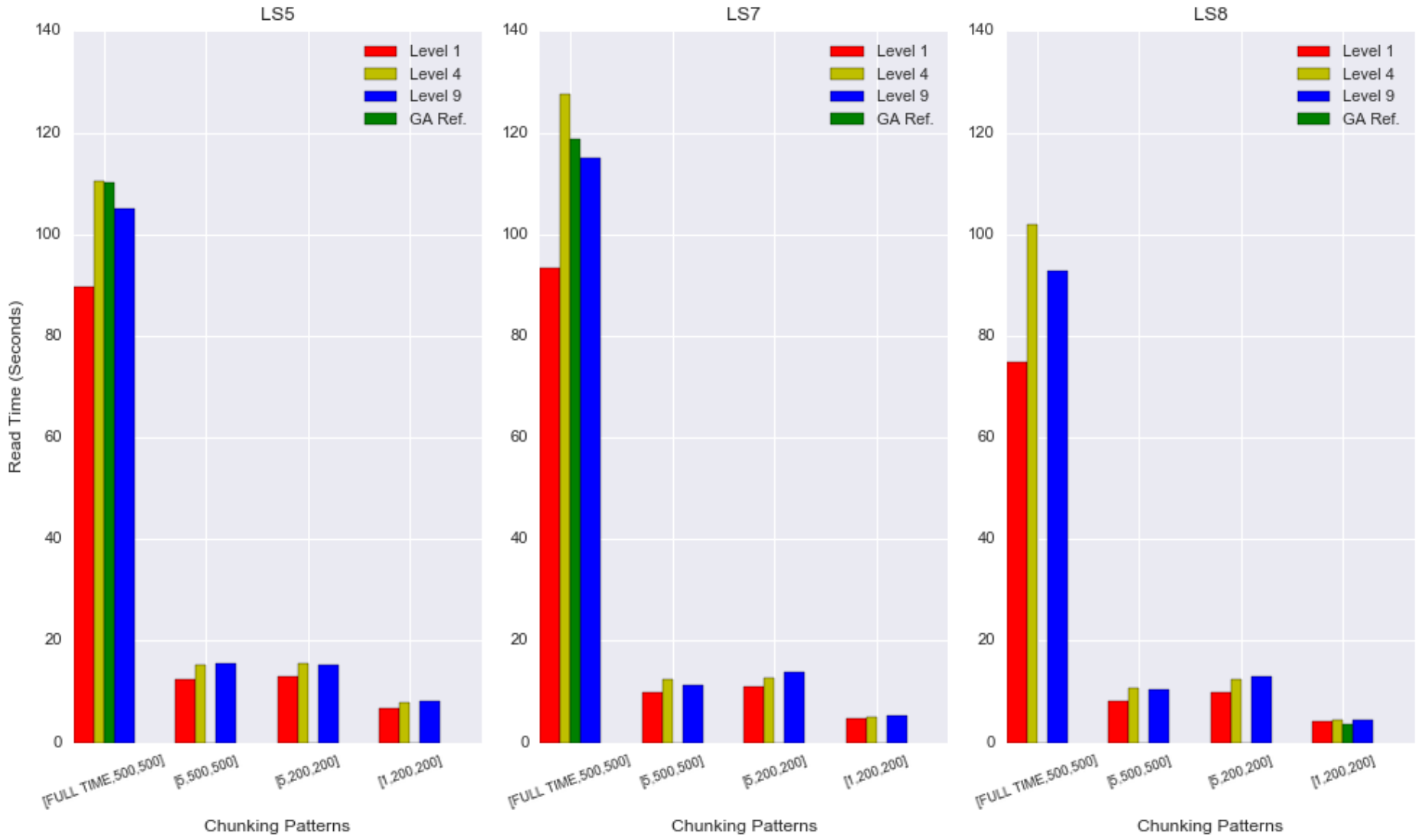




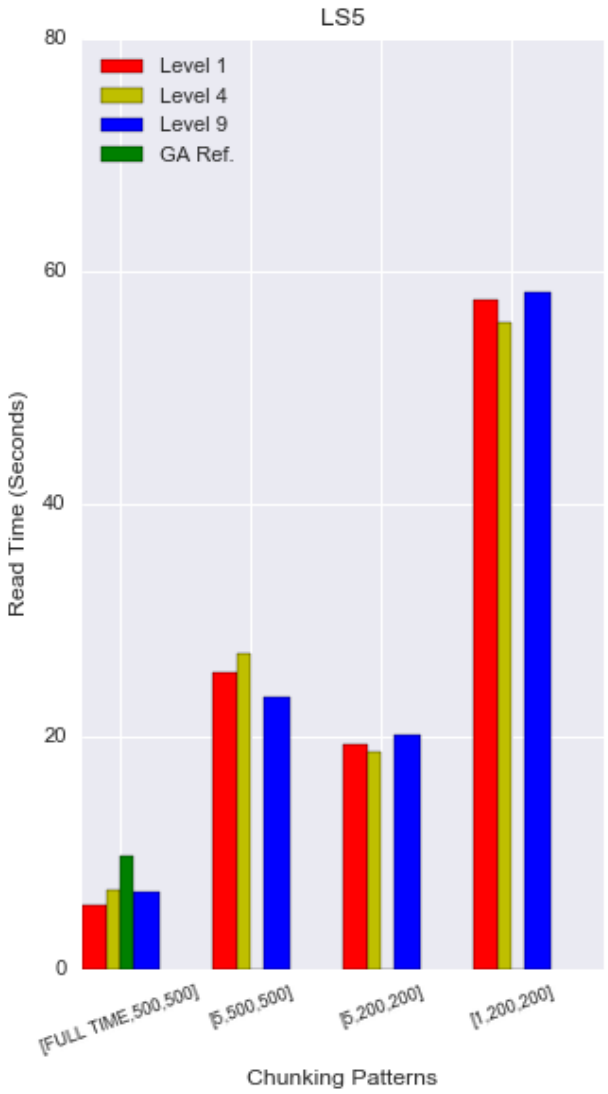
Serial IO: Optimized compression level

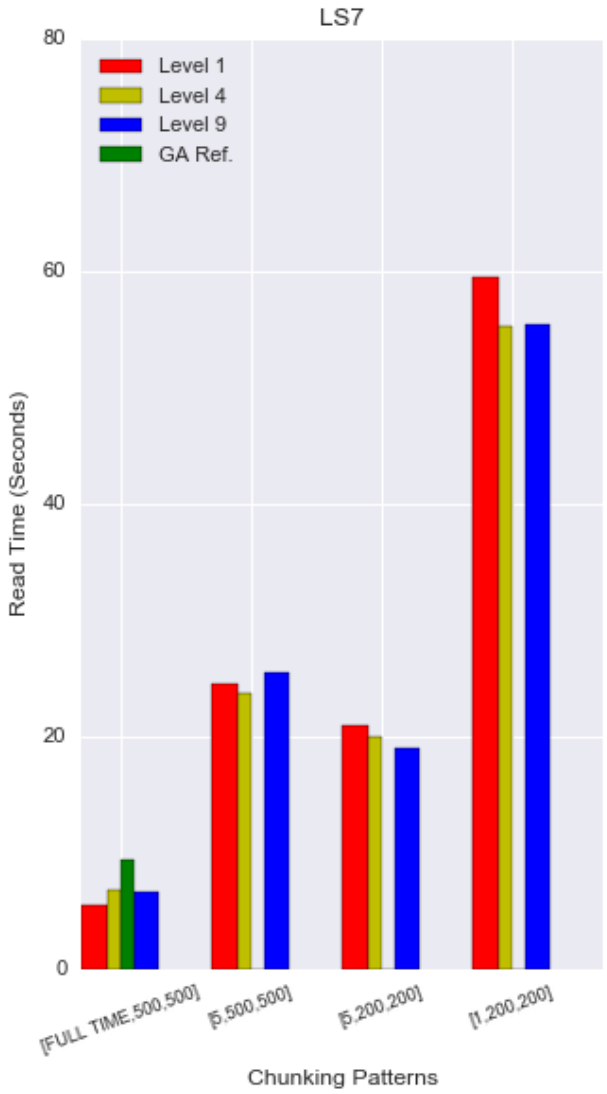


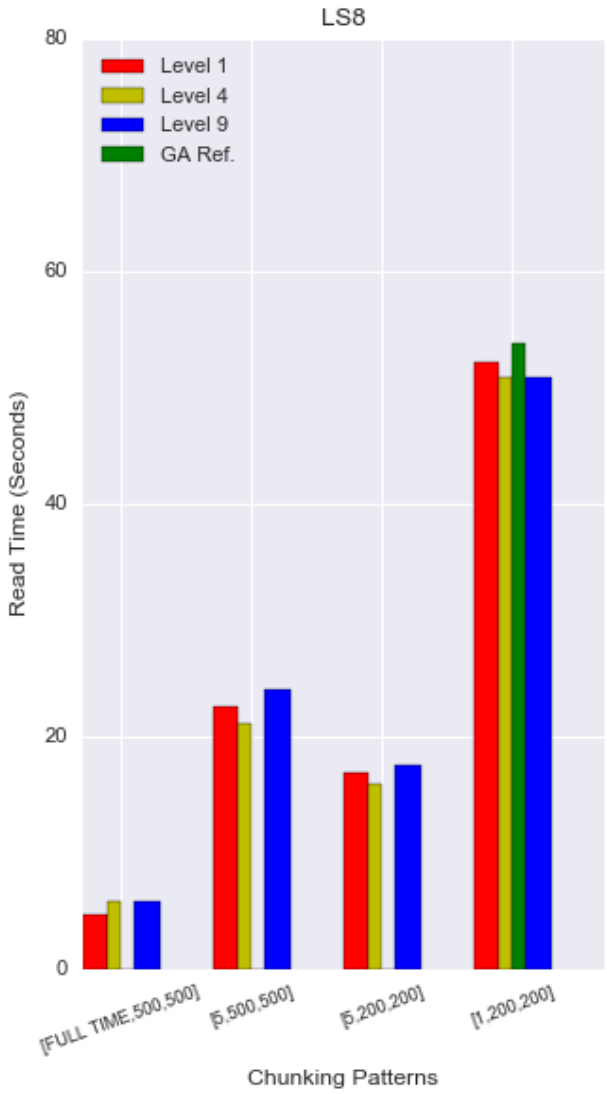
Serial IO: Spatial Access



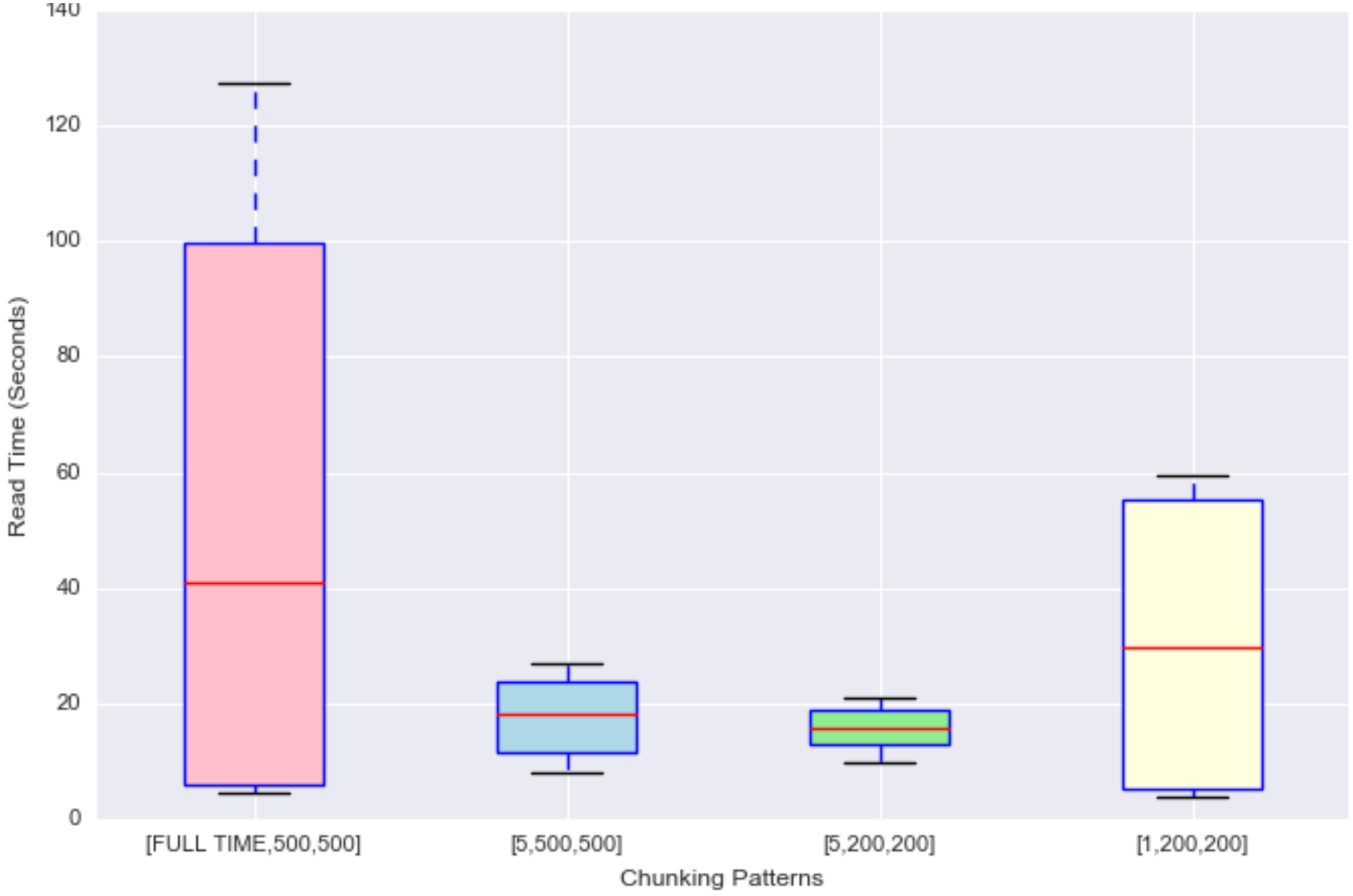
Serial IO: Time series access







Serial IO: optimized chunking patterns

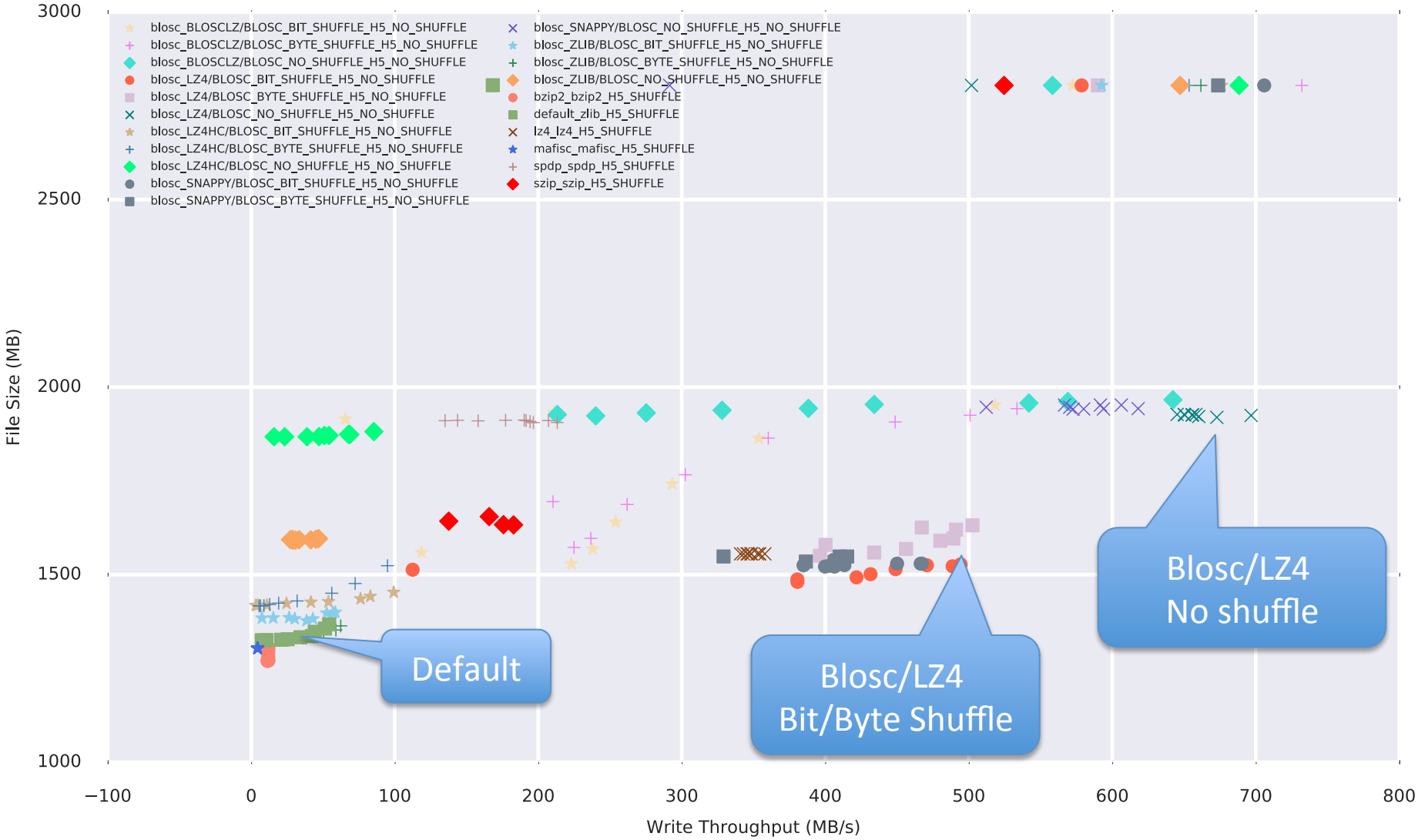


Serial IO: Compression Algorithm

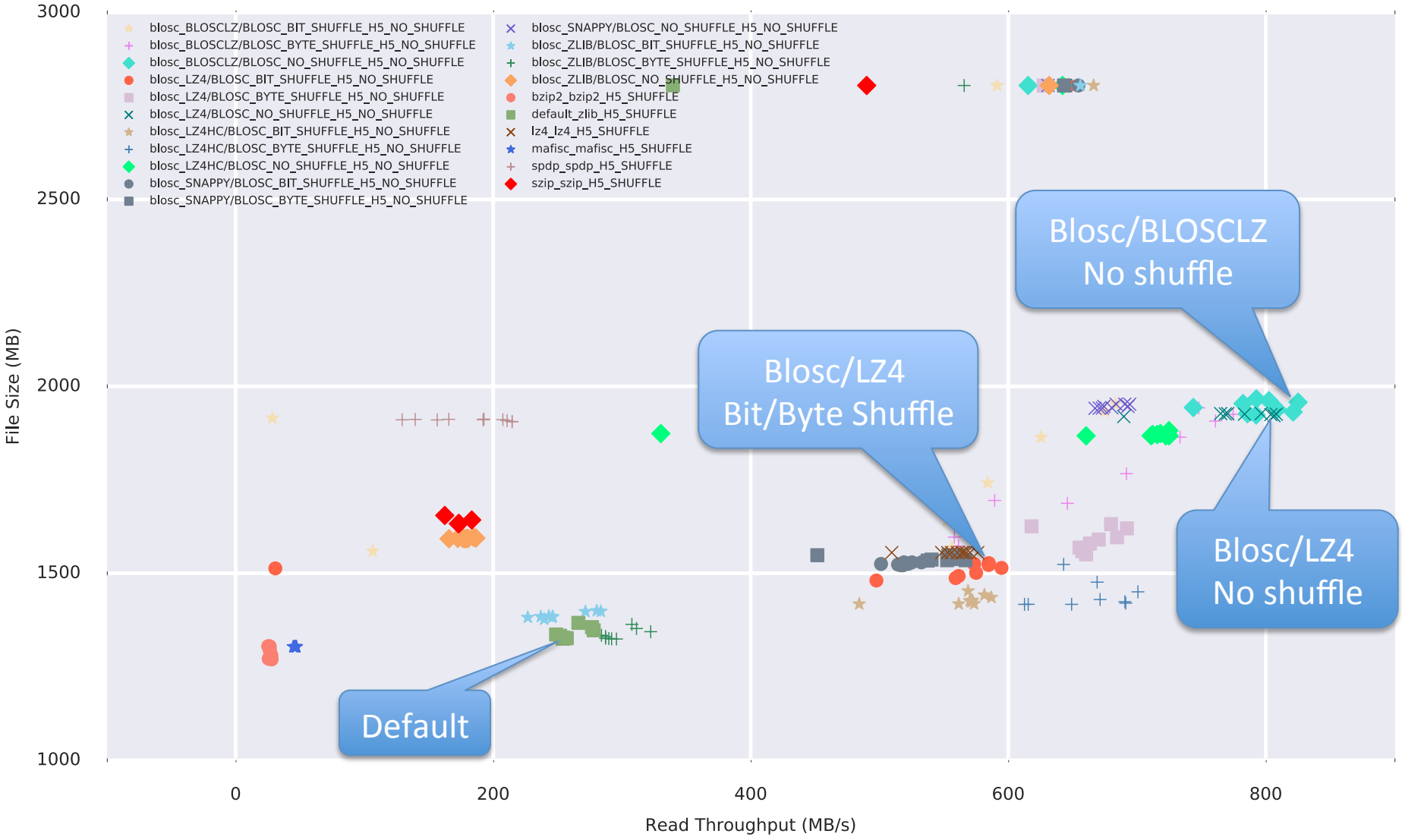
Library	Default	Dynamic Filter
NetCDF4	Deflate(Zlib)	N/A
HDF5	Deflate(Zlib)	Bzip2,mafisc,spdp,szip,LZ4 Blosc(blosclz,lz4hc,lz4,SNAPPY,ZLIB)

The NetCDF4 library is revised to enable different compression algorithms via dynamic filter.

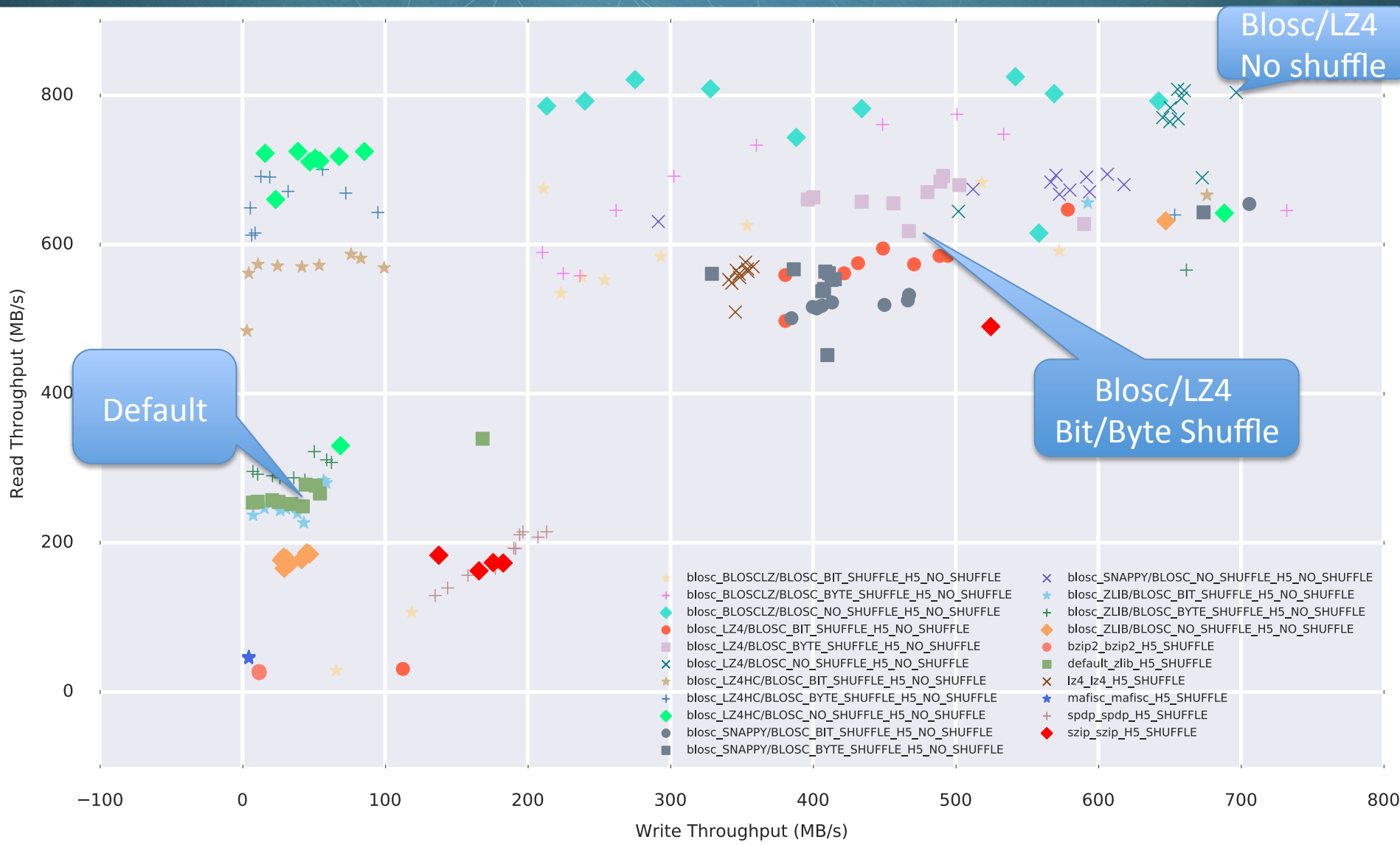
Serial IO: Write performance of different compression algorithms



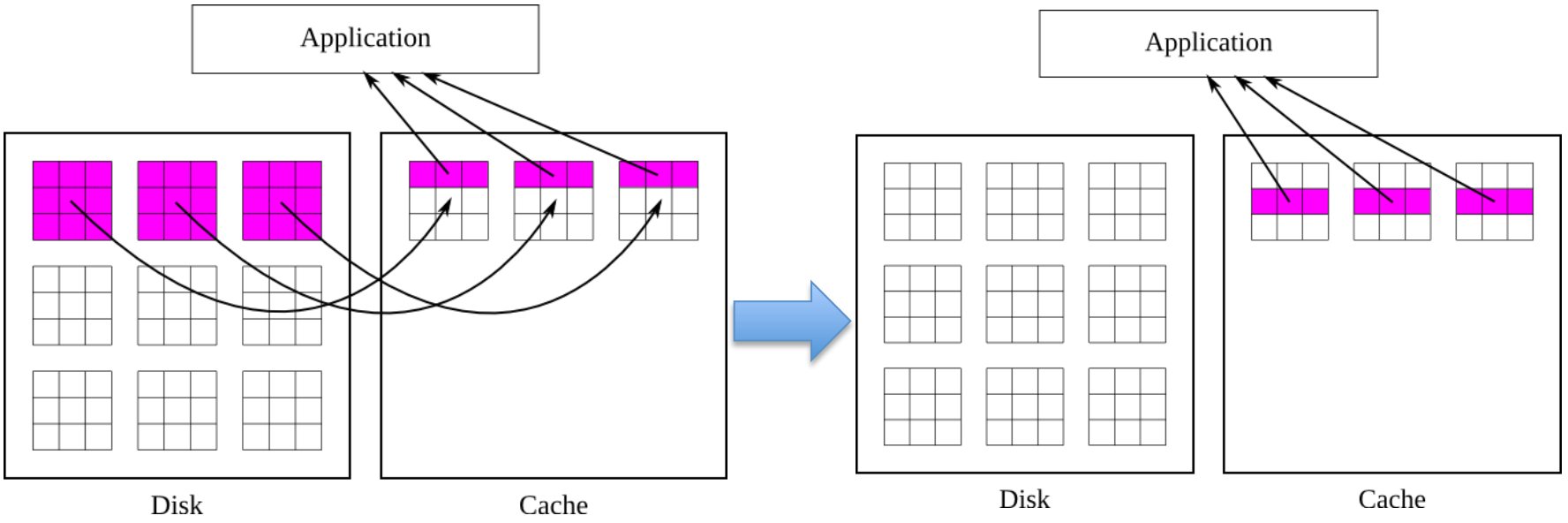
Serial IO: Read performance of different compression algorithms



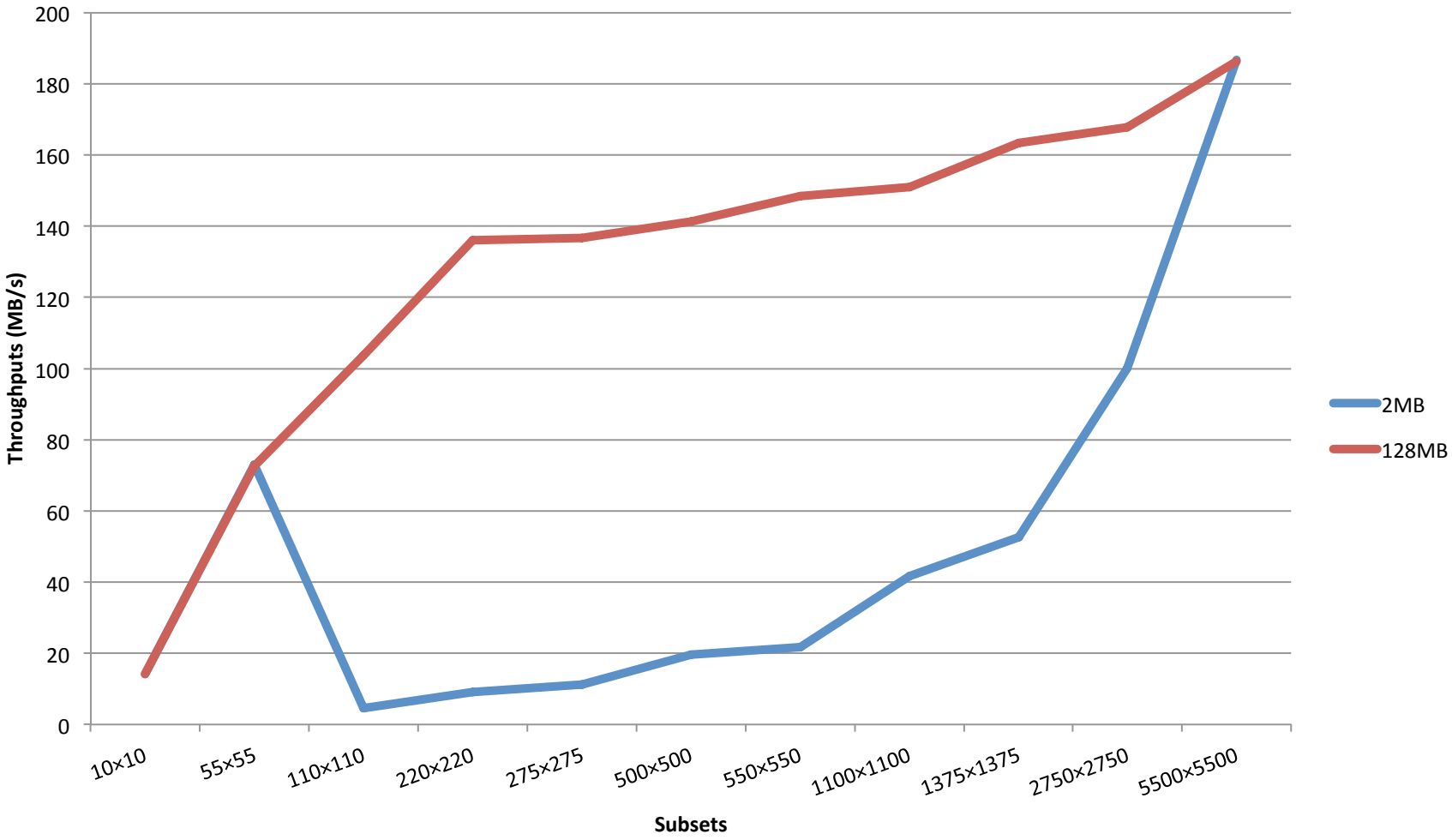
Serial IO: Read/Write performance of different compression algorithm



Serial IO: Chunk cache

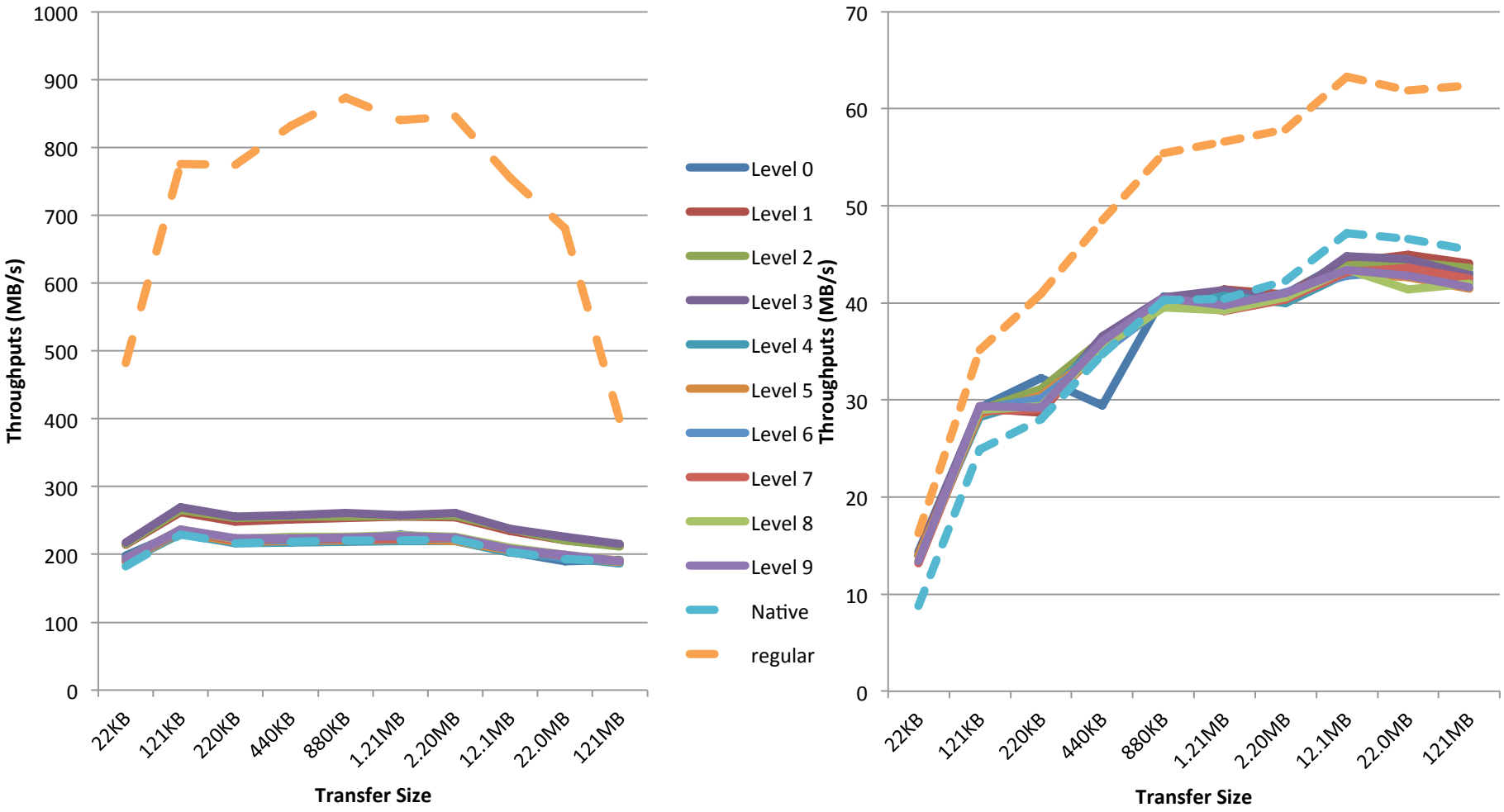


Serial IO: chunk cache size



Chunk size=1×5500

Serial IO: Accessing local file system vs. remote THREDDS server



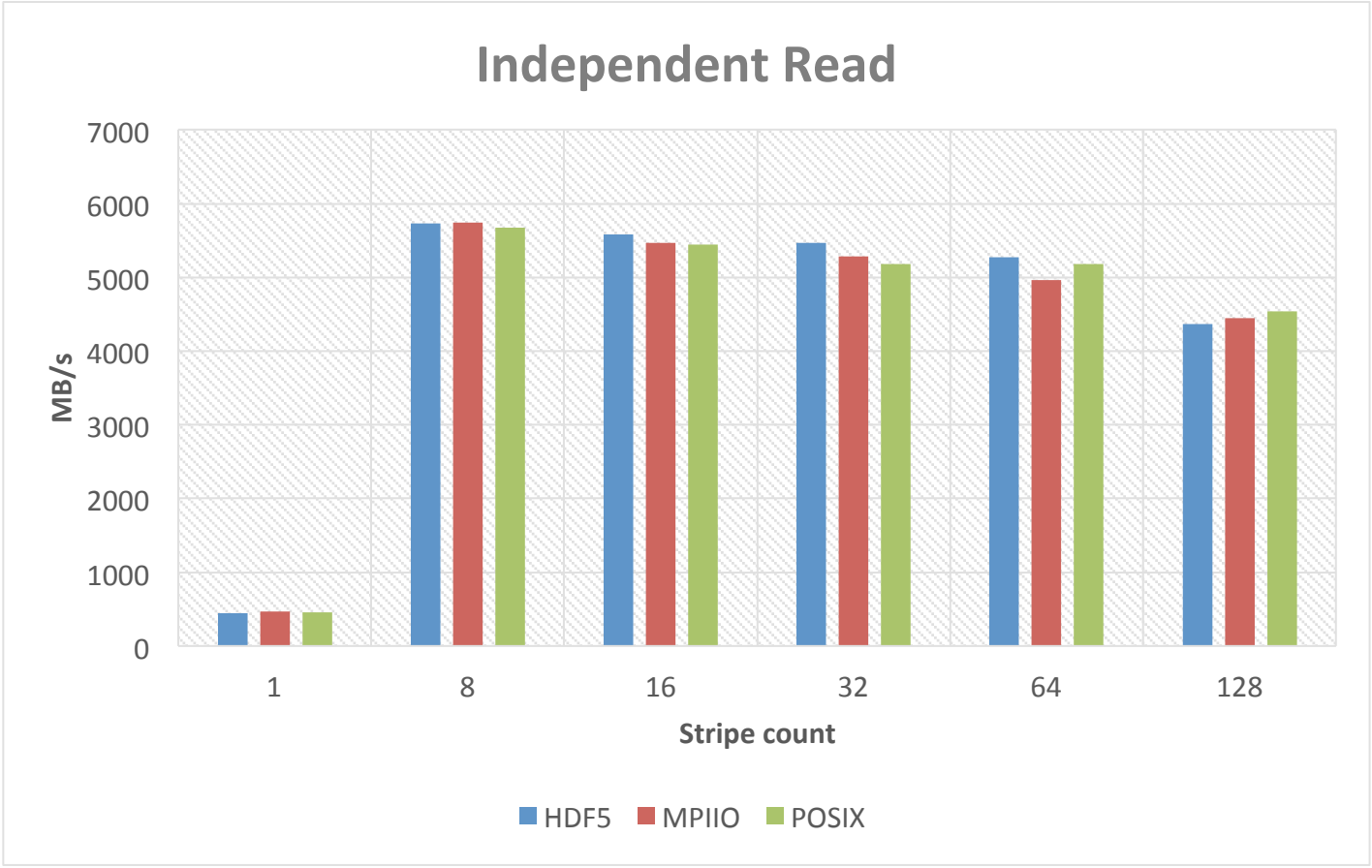
PARALLEL IO

Parallel IO: Lustre tuning on stripe count

IOR Benchmark
MPI size = 16
Stripe size =1M
Block size = 8G
Transfer size = 32M

Parallel IO: Lustre file system tuning on stripe count

IOR Benchmark
MPI size = 16
Stripe size =1M
Block size = 8G
Transfer size = 32M



MPI-IO tuning

- Default value for MPI ranks using 8 cores/node * 2 nodes:
  - *direct_read = false*
  - *direct_write = false*
  - *romio_lustre_co_ratio = 1*
  - *romio_lustre_coll_threshold = 0*
  - *romio_lustre_ds_in_coll = enable*
  - cb_buffer_size = 16777216 (16M)
  - romio_cb_read = automatic
  - romio_cb_write = automatic
  - cb_nodes = 2
  - *romio_no_indep_rw = false*
  - *romio_cb_pfr = disable*
  - *romio_cb_fr_types = aar*
  - *romio_cb_fr_alignment = 1*
  - *romio_cb_ds_threshold = 0*
  - *romio_cb_alltoall = automatic*
  - *ind_rd_buffer_size = 4194304*
  - *ind_wr_buffer_size = 524288*
  - romio_ds_read = automatic
  - romio_ds_write = automatic
  - cb_config_list = *:1
  - striping_unit = 1048576
  - striping_factor = 1
  - *romio_lustre_start_iodevice = 0*

- Collective buffering
  - Romio_cb_read/write
    - auto
    - enable
    - disable
  - Romio_config_list (aggregators)
  - Cb_buffer_size

- Data sieving
  - Romio_ds_read/write
    - auto
    - enable
    - disable

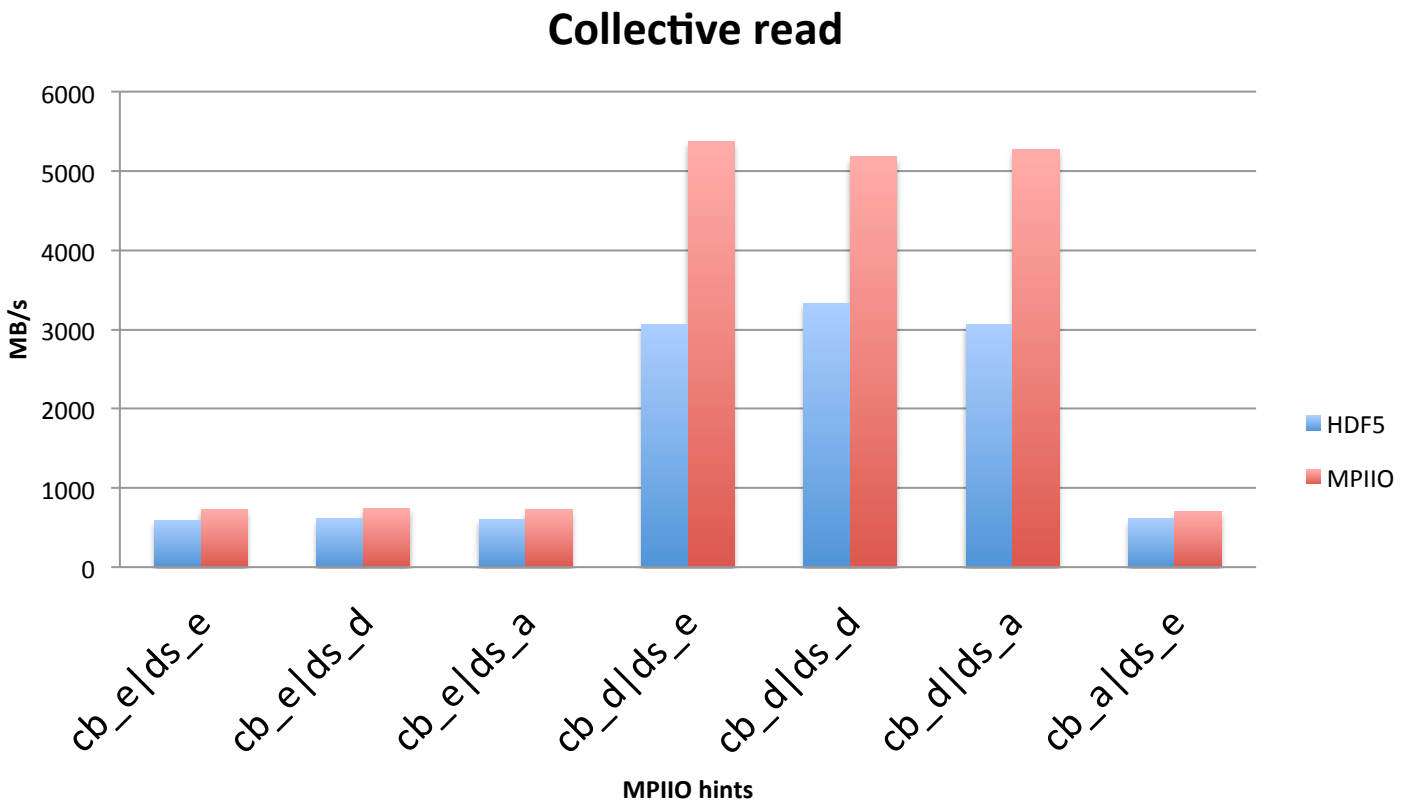
Parallel IO: MPI-IO Tuning on data sieving and collective buffering

- Data sieving
  - I/O performance suffers considerably when making many small I/O requests
  - Access on small, non-contiguous regions of data can be optimized by grouping
  - requests and using temporary buffers
  - This optimization is local to each process (non-collective operation)

- Collective buffering
  - processes to match data layout in file.
  - Mix of I/O and MPI communications to read or write data
  - Communication phase to merge data from different processes into large chunks
  - File accesses are done only by selected processes (called aggregators), the others communicates with them
  - Large operations are split into multiple phases (to limit the size of the buffers and to overlap communications and I/O)

MPI-IO tuning: contiguous access

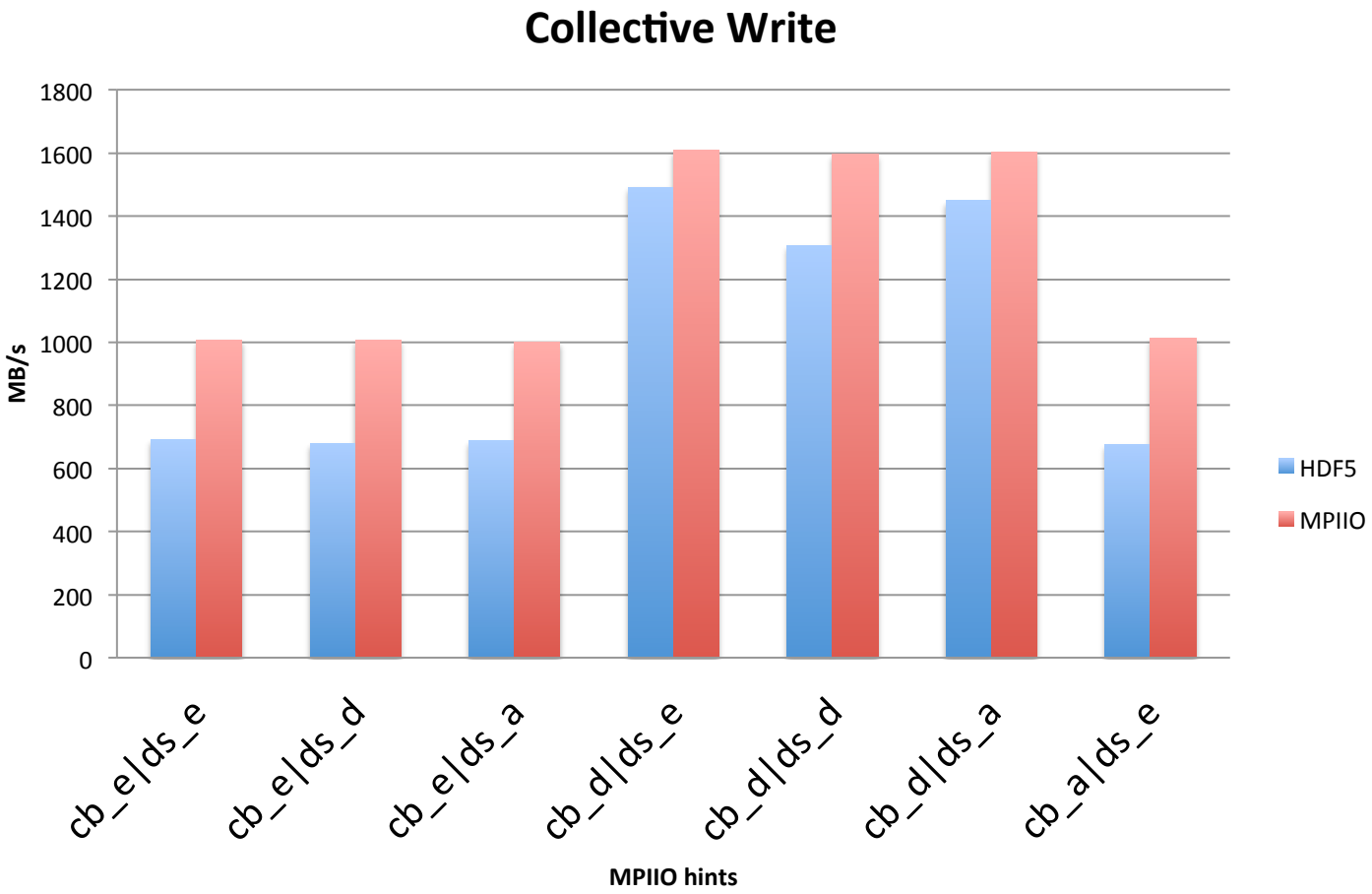
MPI size = 16
Stripe count = 8
Stripe size: 1M
Block size:
1G (cfg 1,2,3,7)
4G (cfg 4,5,6)
Transfer size = 8M



Disable collective buffering to get better performance for contiguous access.

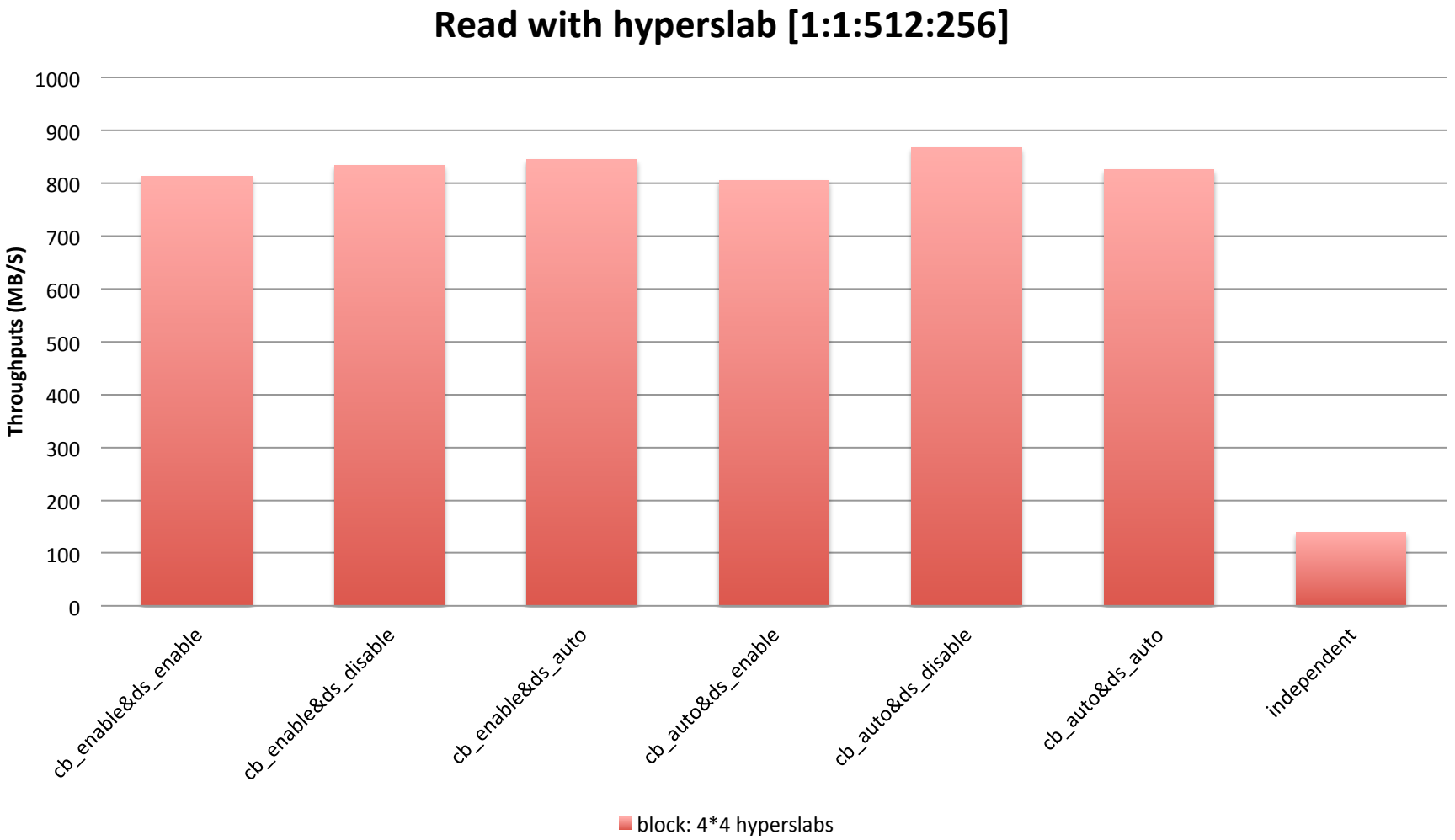
MPI-IO tuning: contiguous access

Raijin:/short
MPI size = 16
Stripe count = 8
Stripe size: 1M
Block size:
1G (cfg 1,2,3,7)
4G (cfg 4,5,6)
Transfer size = 8M

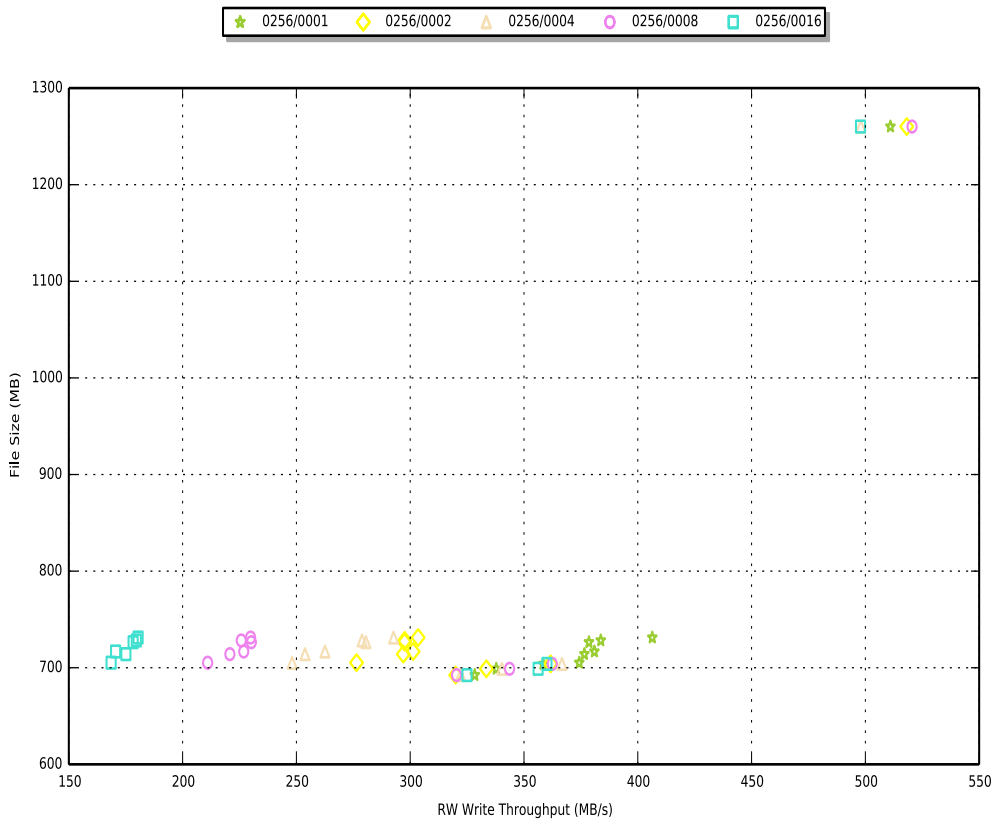


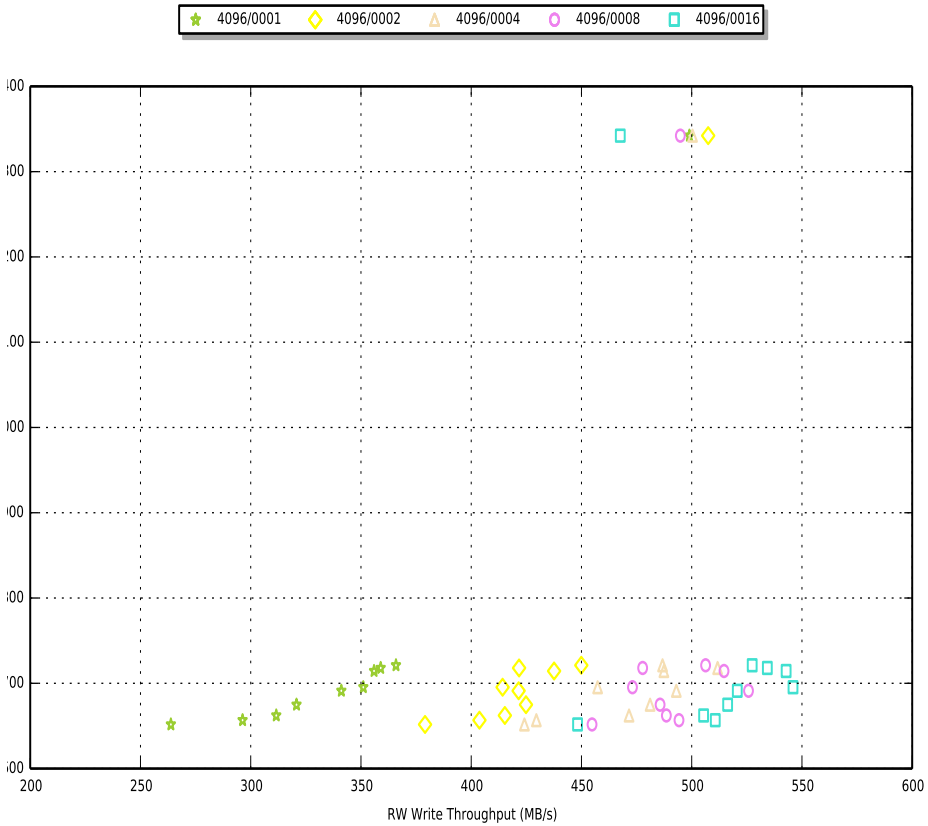
Disable collective buffer to get better performance for contiguous access.

MPI-IO Tuning: Non-Contiguous Access

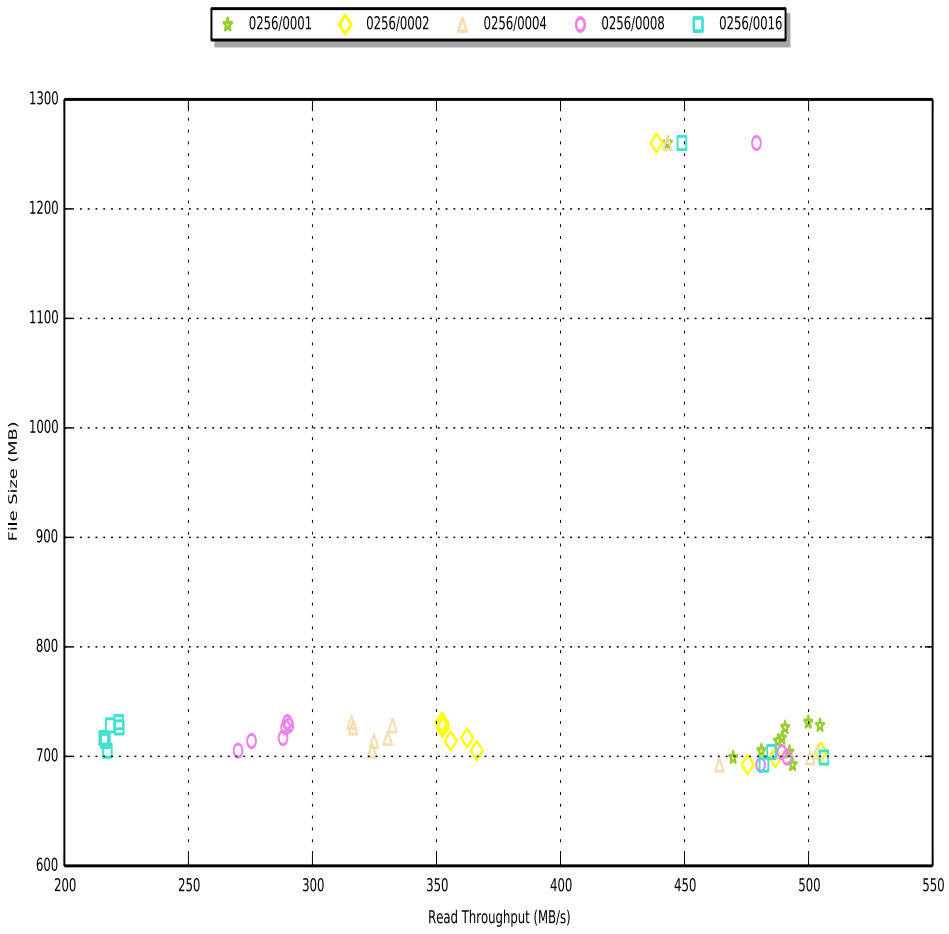


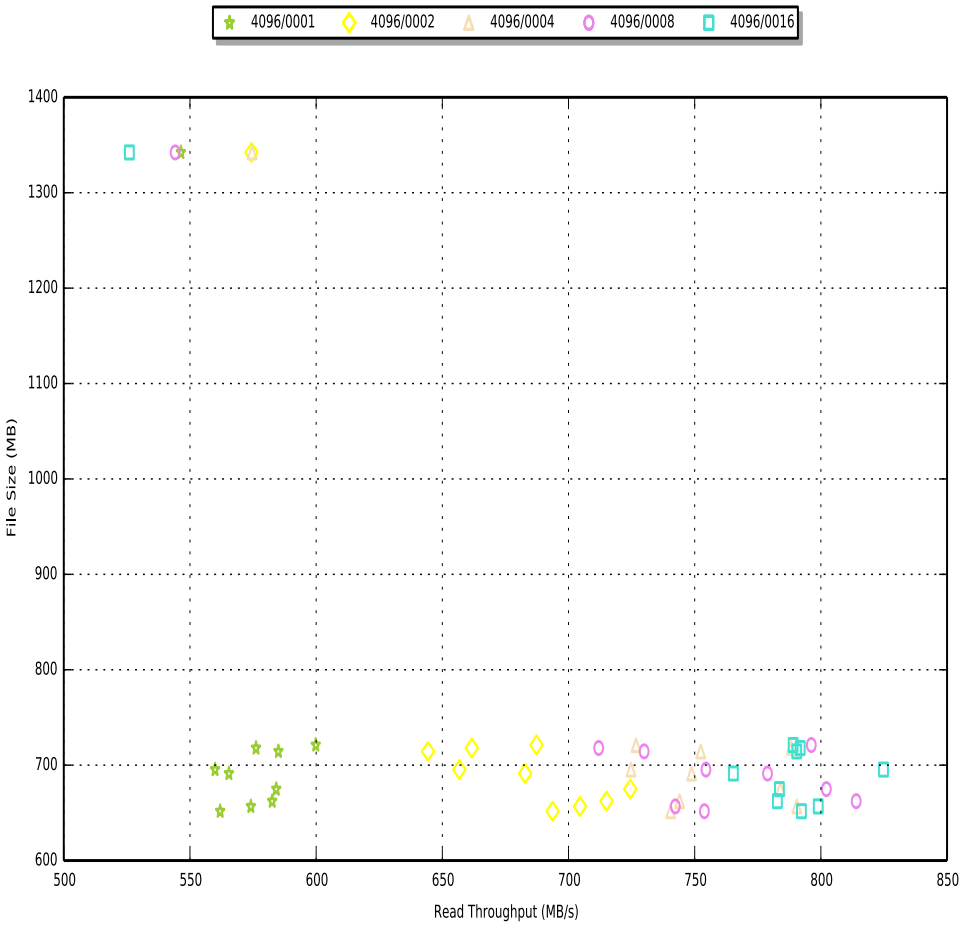
Parallel IO: write performance of Blosc multithreading (LZ4 byte shuffling)





Parallel IO: Read performance of Blosc multithreading (LZ4 byte shuffling)

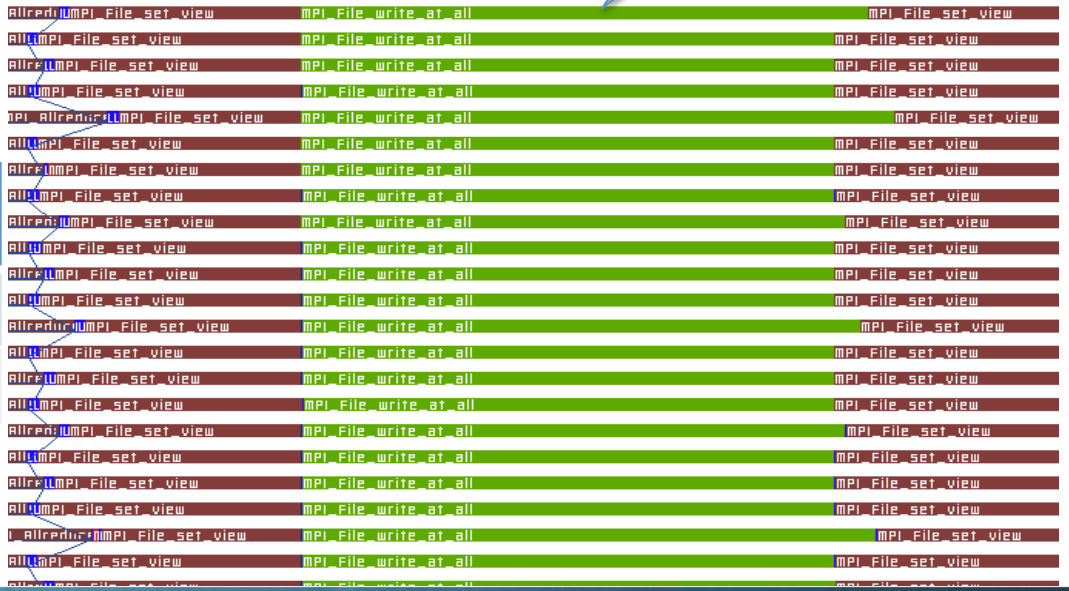




Parallel IO: IO profiling and performance tuning on a climate application

Independent metadata operations

Collective metadata operations



Elapsed Time(s)	Serial IO	Official HDF5/NetCDF4	Patched HDF5/NetCDF4
Case 1	4909	10173	3766
Case 2	1625	8543	1341

- IO performance tuning involves many parameters of interest
  - File System (Lustre): stripe count, stripe unit
  - MPI-I/O: collective buffer size, collective nodes etc.
  - NetCDF/HDF5: alignment, sieve buffer size, data layout, chunking, compression etc.
  - User application: access patterns,
  - Access to data: network (remote) vs in-situ (local)

Thank you!