Collective I/O for Exascale I/O Intensive Applications

Sai B. Narasimhamurthy
Lead Researcher

xyratex
A Seagate Company

Agenda

- ❑Goal of the work
- ❑Exascale10: A Quick Background
- ❑DEEP-ER Project: A Short background
- ❑Exascale I/O Intensive applications: Key Requirements
- ❑The small I/O problem
- ❑Existing Collective I/O techniques & drawbacks
- ❑Solution framework
- ❑Current Status and Next Steps

Goal of the work

- Developing a ubiquitous software middleware based solution to address key performance optimization issues for I/O intensive Extreme scale out applications

  - Small I/O is seen is a major problem for I/O even at Petascale

  - Trying to address the small I/O problem through the newly architected E10 middleware

- The solution should be applicable to a wide variety of applications and back-end object stores and file systems, etc

- Solution part of the DEEP-ER EU project and is targeted to be an E10 component

Exascale10 a quick background

- Develop a ubiquitous middleware that helps I/O scaling
  - Works for a wide variety of applications
  - Agnostic of any backend storage/file systems
- Based on requirements captured in 2012/13 from application experts worldwide
- Participation from more than 40 organisations worldwide (Big Labs, Academics and Industry experts)
- E10 now part-funded in **DEEP-ER** and **Mont-Blanc2** EU projects

xyratex
A Seagate Company

Exascale10 a quick background..

- File Systems cannot scale “as is” (examples)
  - File system interfaces too low level for apps to efficient make use of them (for providing hints, optimizations, layouts, etc)
  - Overlapping stripe writes and small I/Os from clients cannot scale , performance wise
    - Intelligent Middleware could detect such scenarios
  - Performance overheads due to locks and synchronisation cannot scale
  - Specialised read ahead techniques (For Ex: Speculative read ahead) not possible
  - Various formats such as HDF5/NetCDF have their own semantics which is repeated in file systems

Exascale10 a quick background..

- Each layer has its own semantics:
  - Re-implementation of the same optimization strategies
  - Layer specific approaches drastically degrade performance, prevent scalability

HPC Application
(eg: Climate Simulation Codes)

High-Level I/O Library/Data Formats
(eg. HDF5, NetCDF4, Grib2)

(a) Mapping application data abstractions(Eg. Matrices, Vectors, etc) to storage abstractions(objects, files)
(b) Portability of data across platforms

Classic I/O Middleware
(eg. MPI-IO)

(a) Co-ordinates file accesses between multiple processes
(b) Enables parallel I/O from processes to files

I/O Forwarding
(eg. ZOID)

(a) Perform I/O on behalf of compute nodes
(b) Optimizations (aggregation, caching, rescheduling, etc)

EIOW Middleware "realm" (High-Level I/O Library/Data Formats to I/O Forwarding)

Parallel File System
(eg. Lustre, GPFS)
Includes communications infrastructure such as LNET connecting clients/servers

(a) Organises files and objects in many nodes
(b) Parallelize data access

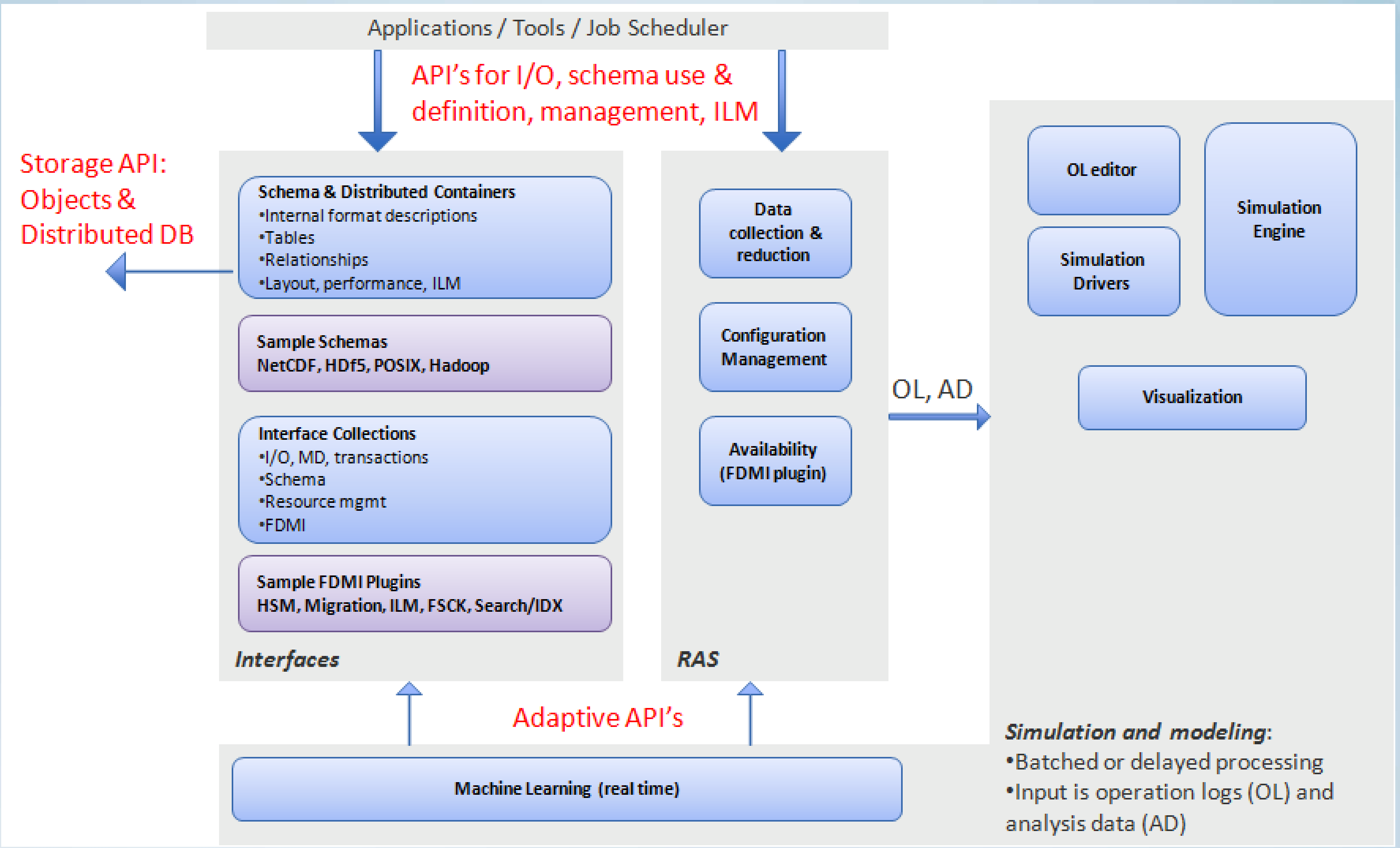
Local File System
(eg. EXT4)

Maps a file/object in bytes and blocks on storage hardware

Substrate (Parallel File System to Local File System)

xyratex
A Seagate Company

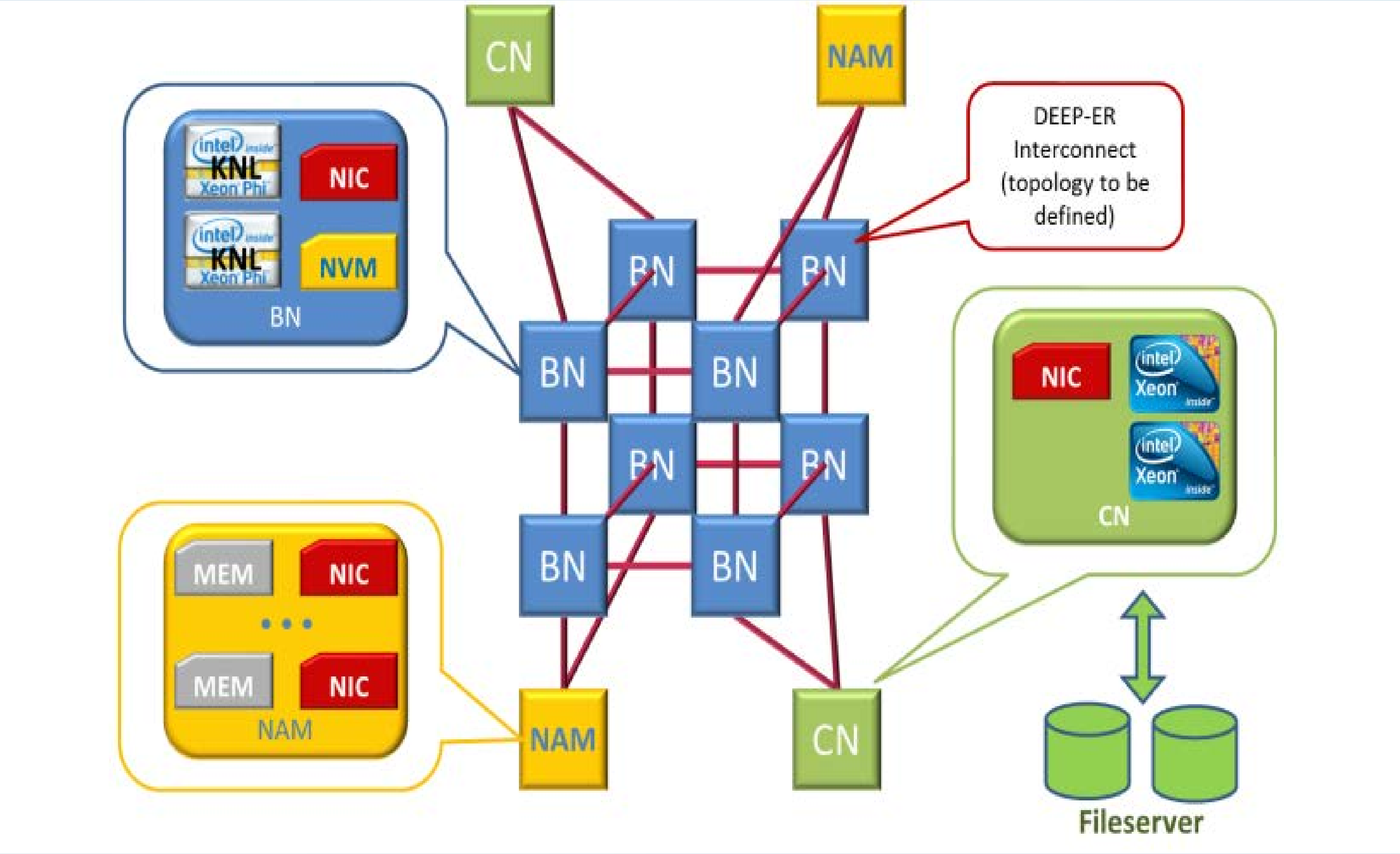
Exascale10 a quick background..



DEEP-ER EU project, a short background

- Extension of the “DEEP” FP7 programme funded EU project addressing Exascale Compute
  - Separately addressing highly scalable code parts in Exascale applications(envisioned in DEEP)
  - Highly scalable, efficient and easy to use Parallel I/O for Exascale
    - Exploration of NVRAM technologies at various levels in the I/O stack
  - Low-over head user-level checkpoint/restart and task recovery for Exascale apps
  - Co-design approach with applications

DEEP-ER project, a short background



DEEP-ER project, a short background

- Supercomputing centres (PRACE):
- Architectural challenges
- Technology and systems
- Application codes

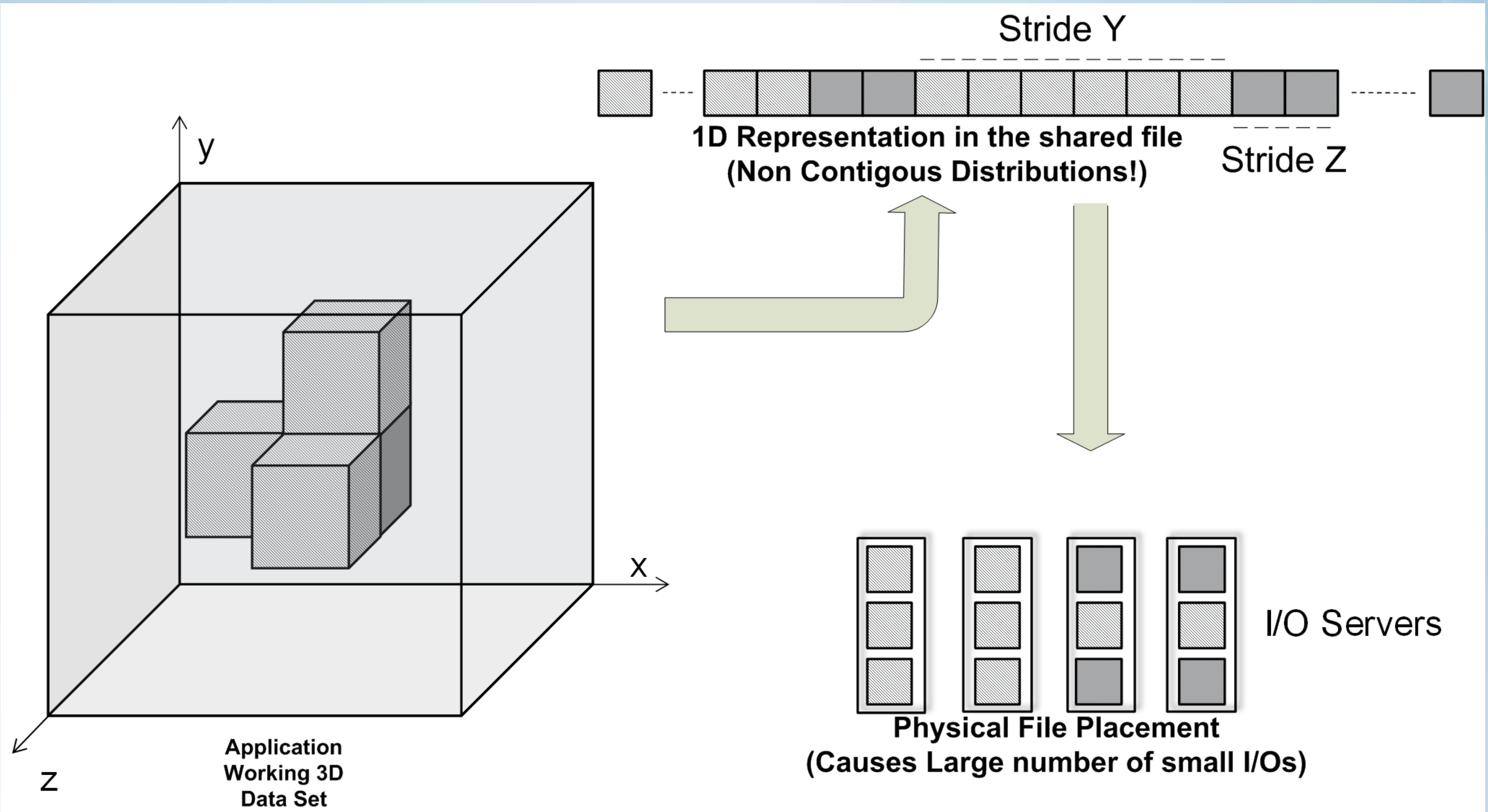
xyratex
A Seagate Company

Exascale I/O Intensive apps: Key Requirements

- Summary of key I/O requirements from the DEEP-ER (Exascale targeted) Applications
  - I/O intensive modes
  - Need to address large shared files
  - I/O issues need to be addressed for both checkpoint restart as well as simulation based file I/O
  - Optimizations to address small I/O on large shared files absolutely essential

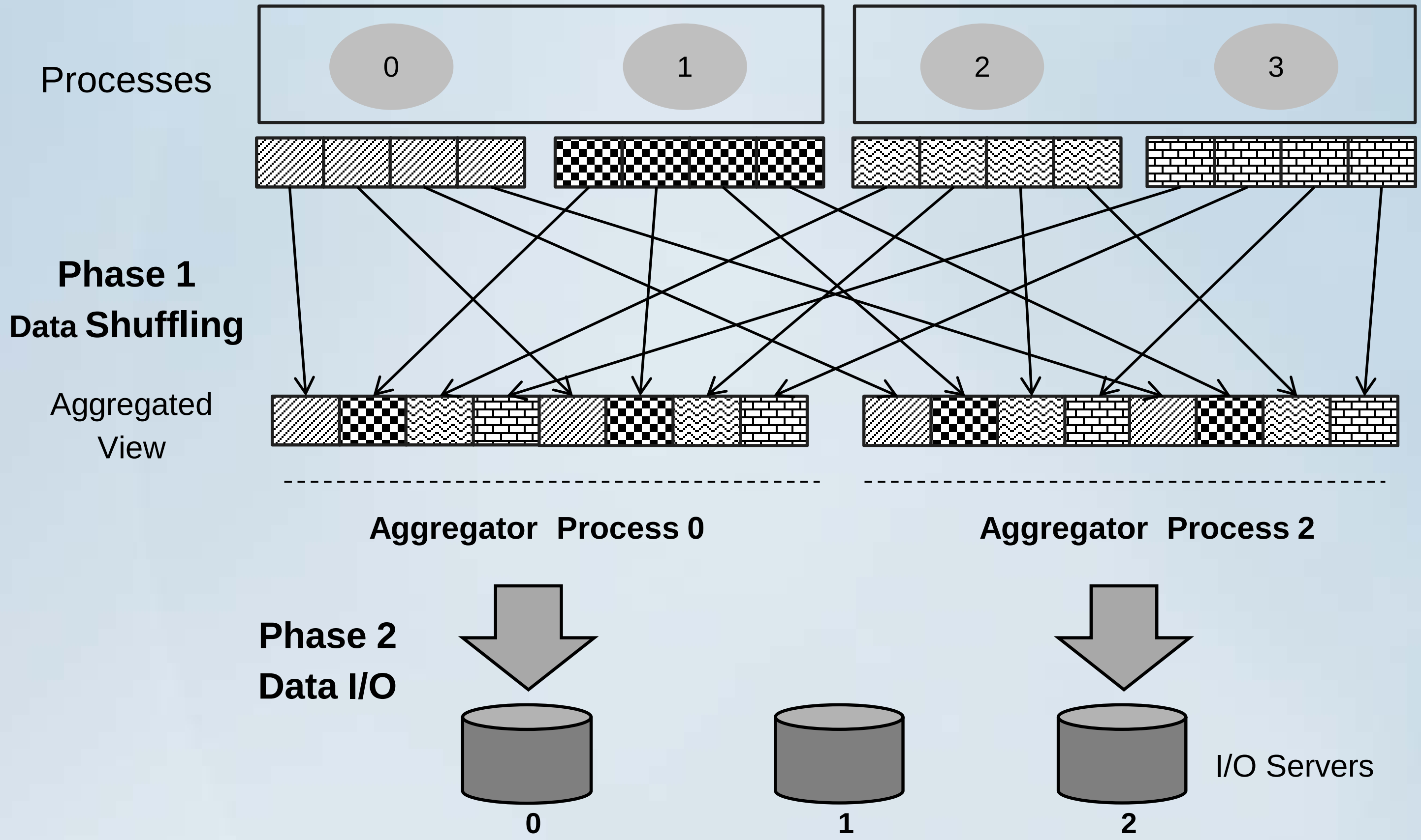
Collective I/O at Exascale needs to be a key optimization!

The Small I/O Problem



- ❑Congested I/O Servers
- ❑Reduced Disk I/O Bandwidth

Existing Collective I/O (2 Phase I/O)

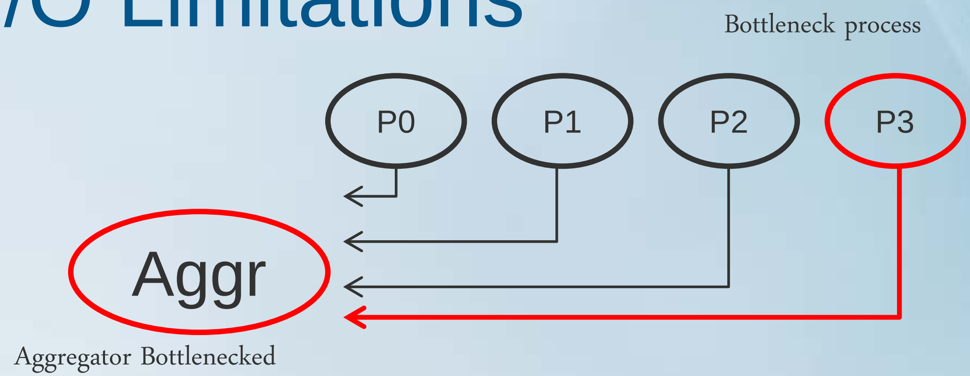


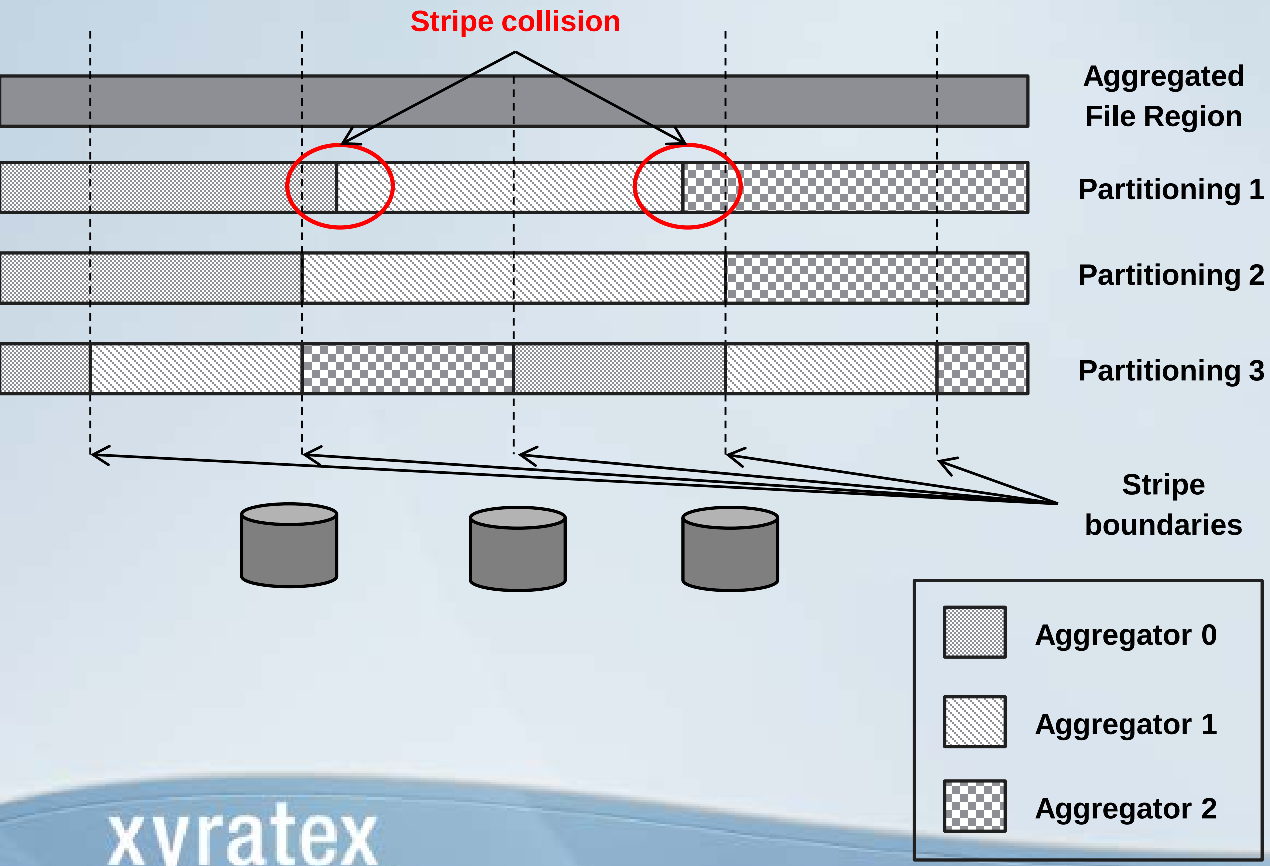
- Better performance than small contiguous I/O

xyratex
A Seagate Company

2-Phase I/O Limitations

❑Synchronisation issues





❑Data layout issues (lack of physical layout awareness among aggregators)

❑Stripe contention

❑I/O server contention

Collective I/O - Limitations

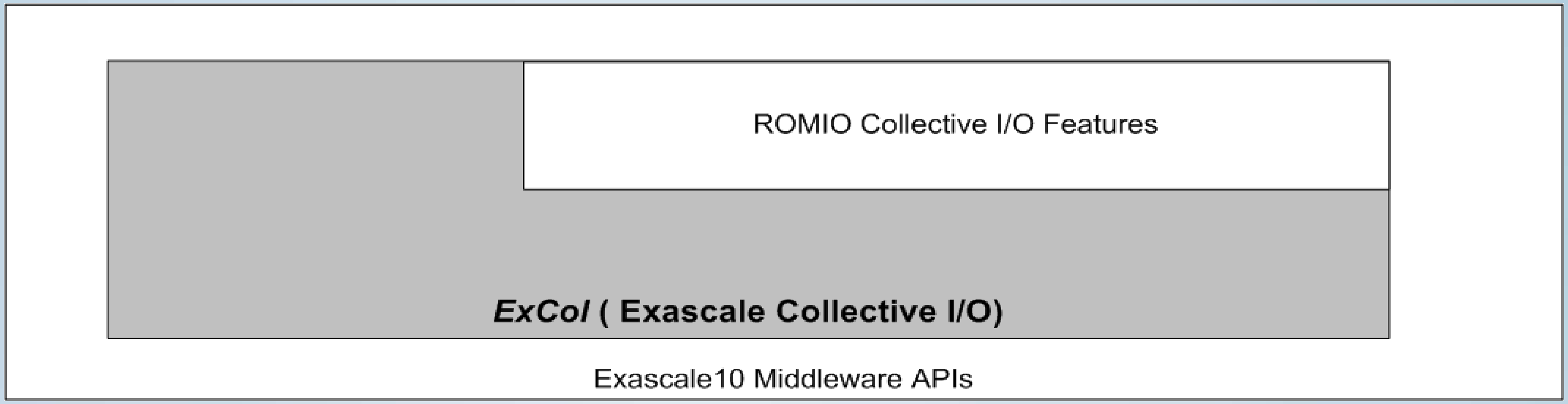
HPC system requirements[ross2013]

	2010	2018	Factor Change
System Peak	2 Pf/s	1 Ef/s	500
Power	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 Tf/s	10 Tf/s	80
Node Memory BW	25 GB/s	400 GB/s	16
Node Concurrency	12 CPUs	1000 CPUs	83
Interconnect BW	1.5 GB/s	50 GB/s	33
System Size (nodes)	20 K nodes	1 M nodes	50
Total concurrency	225 K	1 B	4444
Storage	15 PB	300 PB	20
I/O Bandwidth	0.2 TB/s	20 TB/s	100

- *Aggregator operations consume memory resources*

- *Neither the **memory bandwidth** nor the **memory capacity** will scale by the same factor as the total concurrency(the scale of the number of nodes)![Vetter2008]*

ExCol - Solution Framework



- Collective I/O enhancements for Exascale
  - Primarily addressing the small I/O problem as discussed earlier
  - ..but at massive I/O scale-outs
- Implementation will be built around existing collective I/O implementations (in ROMIO) as a base
  - No reinventing the wheel
  - Preserving MPI-IO interoperability semantics for applications
- APIs will be part of Exascale10 Middleware

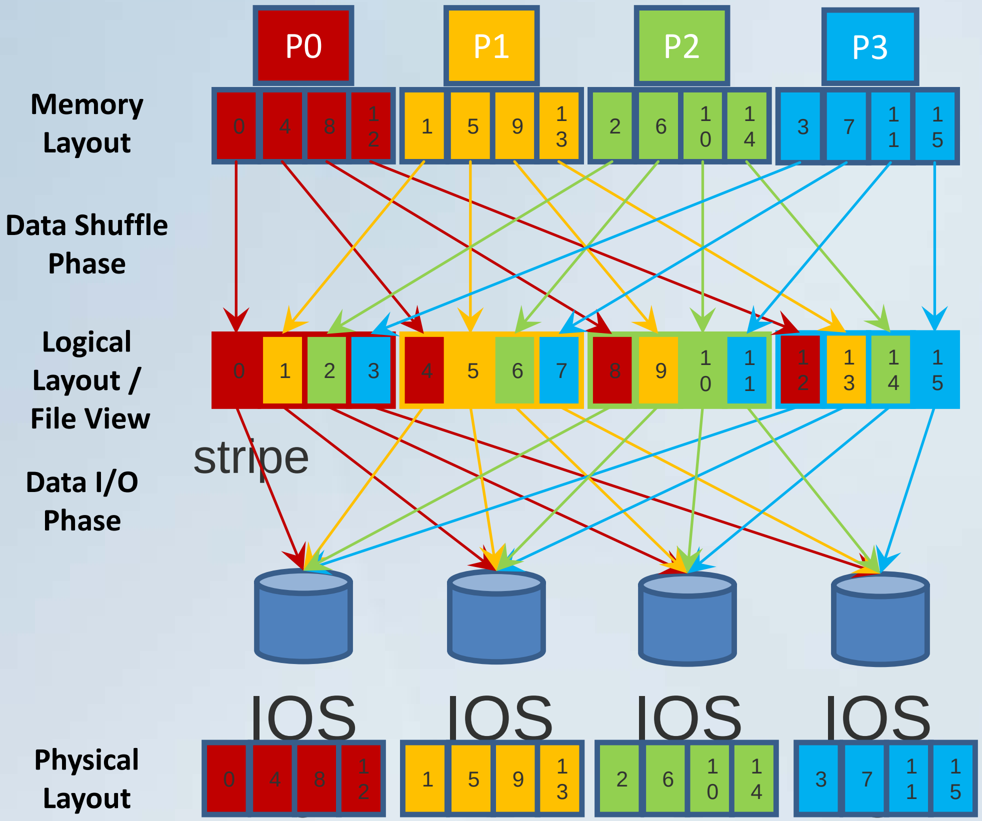
xyratex
A Seagate Company

ExCol- Solution framework

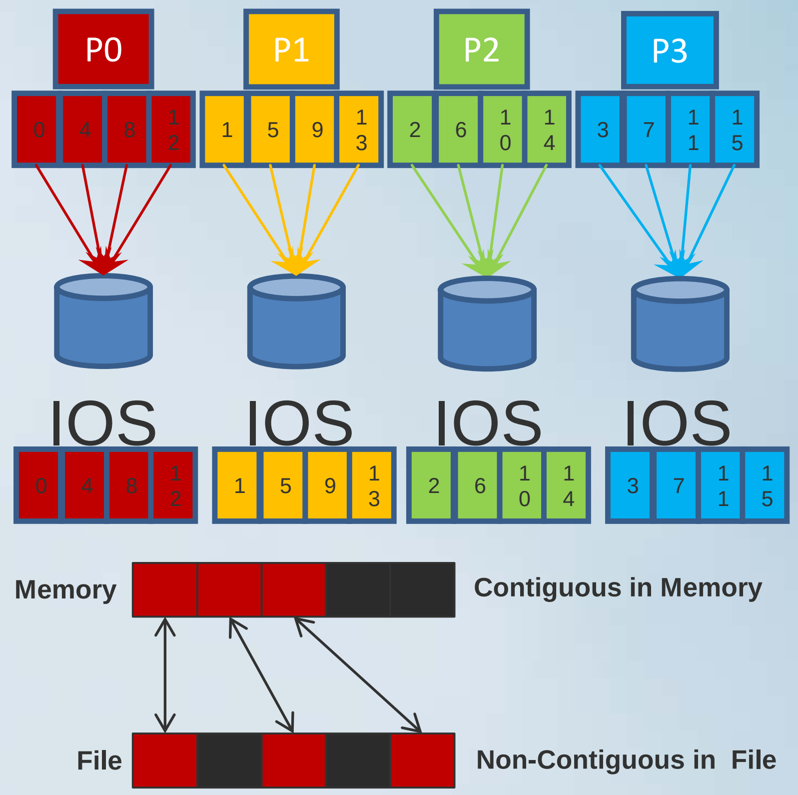
- Avoiding data exchanges between aggregators and processes
  - Conserving memory bandwidth
- Avoiding very large aggregator buffers
- Physical layout awareness
  - Leveraging the concept of advanced file views for aggregators
- Optimizations to deal with NVRAM layers between compute and storage (as we have in the DEEP-ER architecture)

ExCol Methods - Example

Classic Extended Two Phase I/O Partitioning



Physical layout aware partitioning, stripe contiguous (no data shuffle)



- Example of Physical layout awareness (and usage of listIO type frameworks)

Current Status and Next Steps

- Phase 1 (October’13 – Feb’14)
  - Understand application I/O requirements for Exascale
  - Background work understanding existing collective I/O and their drawbacks
- Phase 2(March’14 – September’14)
  - Develop solution framework
  - Preliminary architecture
- Phase 3 (October’14 – September’15)
  - Implementation
- Phase 4 (October’15 -)
  - Detailed Evaluations for various applications/file system back-ends

Acknowledgements for the work

- Giuseppe Congiu (ExCol Architecture and Development)
- Malcolm Muggeridge (VP, Xyratex ETG) and the Xyratex Team

Key references

•[ross2013]*Memory conscious collective I/O for Extreme Scale HPC systems* , Yin Lu et al

•[Vetter2008]*HPC Interconnection networks: The Key to Exascale Computing, J.S. Vetter, et al*

•*[Yu2008] Parcoll:Partioned Collective I/O on the Cray XT, Weikuan Yu, et al*

•*[Zhang2008]Making Resonance a Common Case:A High Performance Implementation of Collective I/O on Parallel File Systems, Xuechen Zhang, et al*

•*[Liao2008]Dynamically adapting file domain partitioning methods for collective I/O based on underlying parallel file system locking protocols, We-Keng Liao, et al*

•*[Chen2011]LACIO: A New Collective I/O Strategy for Parallel I/O Systems, Yong Chen, et al*

•*[He2011]Pattern Aware File Re-organisation in MPI-IO , He, et al*

•*[ROMIO] http://www.mcs.anl.gov/research/projects/romio/*

xyratex
A Seagate Company

Thank You

Sai_Narasimhamurthy@xyratex.com