I/O optimizations with Data Aggregation

Florin Isaila‡∗, **Emmanuel Jeannot**†, Preeti Malakar∗, François Tessier∗, Venkatram Vishwanath∗,

∗Argonne National Laboratory, USA
†Inria Bordeaux Sud-Ouest, France
‡University Carlos III, Spain

October 8, 2018

Data Movement at Scale

- JLESC project started early 2015
- Focus from flops to bytes:
  - allocate data
  - move data
  - store data
  - **bring data to the right place at the right time**
- Computer simulation: climate simulation, heart or brain modelling, cosmology, etc

Data Movement at Scale

- Large needs in terms of I/O: high resolution, high fidelity

Table: Example of large simulations I/O coming from diverse papers

Scientific domain	Simulation	Data size
Cosmology	Q Continuum	**2 PB / simulation**
High-Energy Physics	Higgs Boson	**10 PB / year**
Climate / Weather	Hurricane	**240 TB / simulation**

- Growth of supercomputers to meet the performance needs but with increasing gaps

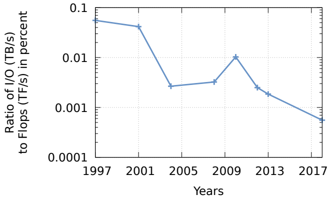
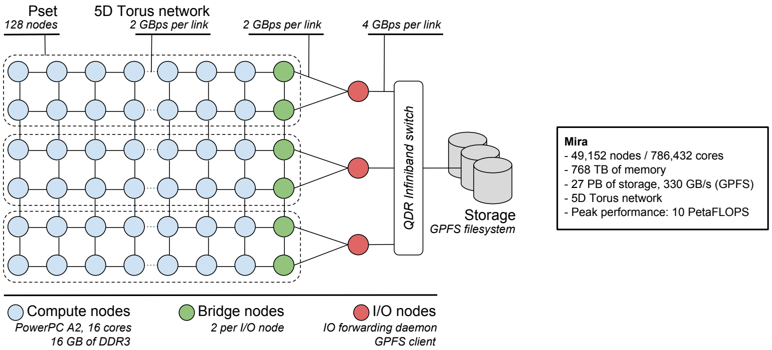


Figure: Ratio IOPS/FLOPS of the #1 Top 500 for the past 20 years. Computing capability has grown at a faster rate than the I/O performance of supercomputers.

Complex Architectures

- Complex network topologies tending to reduce the distance between the data and the storage
  - Multidimensional tori, dragonfly, ...
- Partitioning of the architecture to avoid I/O interference
  - IBM BG/Q with I/O nodes (Figure), Cray with LNET nodes
- New tiers of storage/memory for data staging
  - MCDRAM in KNL, NVRAM, Burst buffer nodes



Two-phase I/O

- Present in MPI I/O implementations like ROMIO
- Optimize collective I/O performance by reducing network contention and increasing I/O bandwidth
- Chose a subset of processes to aggregate data before writing it to the storage system

- Better for large messages (from experiments)
- No real efficient aggregator placement policy
- Informations about upcoming data movement could help

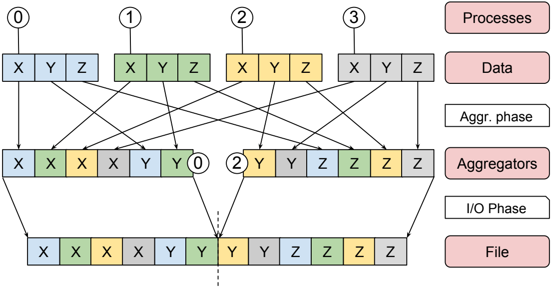


Figure: Two-phase I/O mechanism

Outline

Approach

- Relevant aggregator placement while taking into account:
  - The topology of the architecture
  - The data pattern
- Efficient implementation of the two-phase I/O scheme
  - I/O scheduling with the help of information about the upcomming readings and writings
  - Pipelining aggergation and I/O phase to optimize data movements
  - One-sided communications and non-blocking operations to reduce synchronizations
- **TAPIOCA** (Topology-Aware Parallel I/O: Collective Algorithm)
  - MPI Based library for 2-phase I/O
  - portable through abstracted representation of the machine
  - generalize toward data movement at scale on HPC facility
  - results published in Cluster 2017

Aggregator Placement

- **Goal**: find a compromise between aggregation and I/O costs
- Four tested strategies
  - Shortest path: smallest distance to the I/O node
  - Longest path: longest distance to the I/O node
  - Greedy: lowest rank in partition (can be compared to a MPICH strategy)
  - **Topology-aware**

How to take the topology into account in aggregators mapping?

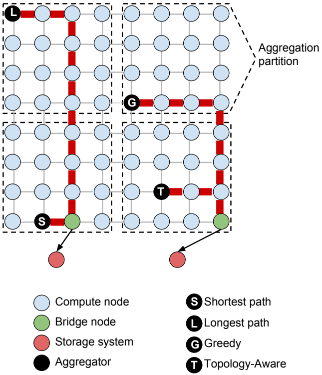
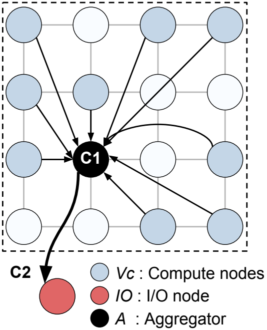


Figure: Data aggregation for I/O: simple partitioning and aggregator election on a grid.

Aggregator Placement - Topology-aware strategy

- $\omega(u,v)$: Amount of data exchanged between nodes u and v
- $d(u,v)$: Number of hops from nodes u to v
- l: The interconnect latency
- $B_{i\rightarrow j}$: The bandwidth from node i to node j
- $C_1 = max\left(l \times d(i,A) + \frac{\omega(i,A)}{B_{i\rightarrow A}}\right), i \in V_C$
- $C_2 = l \times d(A,IO) + \frac{\omega(A,IO)}{B_{A\rightarrow IO}}$



Objective function:

$$TopoAware(A) = min\left(C_1 + C_2\right)$$

- Computed by each process independently in $O(n), n = |V_C|$

Micro-benchmark - Placement strategies

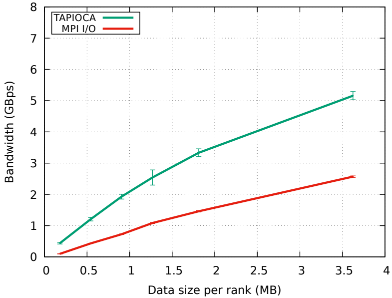
- Evaluation on Mira (BG/Q), 512 nodes, 16 ranks/node
- Each rank sends an amount of data distributed randomly between 0 and 2 MB
- Write to /dev/null of the I/O node (performance of just aggregation and I/O phases)
- Aggregation settings: 16 aggregators, 16 MB buffer size

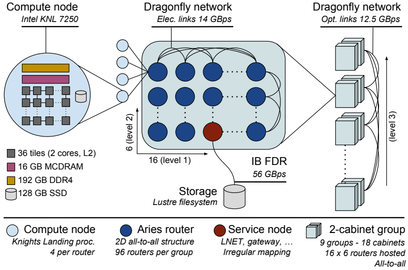
Table: Impact of aggregators placement strategy

Strategy	I/O Bandwidth (MBps)	Aggr. Time/round (ms)
Topology-Aware	2638.40	310.46
Shortest path	2484.39	327.08
Longest path	2202.91	370.40
Greedy	1927.45	421.33

Micro-benchmark - Theta

- 10 PetaFLOPS Cray XC40 supercomputer
- 512 Theta-nodes, 16 ranks per node
- 48 aggregators, 8 MB aggregation buffer size





HACC-IO – MIRA (BG/Q)

- I/O part of a large-scale cosmological application simulating the mass evolution of the universe with particle-mesh techniques
- Each process manage particles defined by 9 variables (*XX*, *YY*, *ZZ*, *VX*, *VY*, *VZ*, *phi*, *pid* and *mask*)

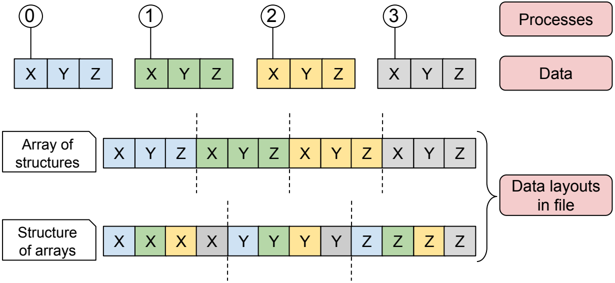
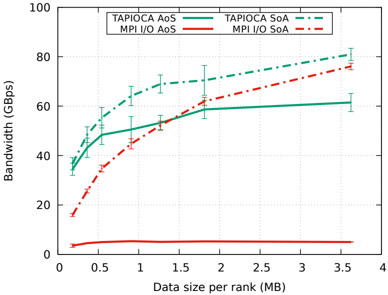


Figure: Data layouts implemented in HACC

HACC-IO – MIRA (BG/Q)

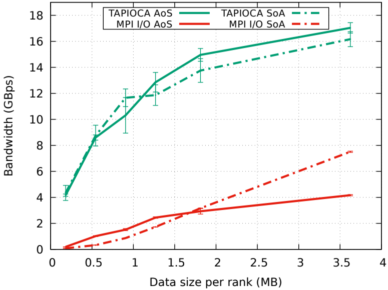
- I/O part of a large-scale cosmological application simulating the mass evolution of the universe with particle-mesh techniques
- Each process manage particles defined by 9 variables (*XX*, *YY*, *ZZ*, *VX*, *VY*, *VZ*, *phi*, *pid* and *mask*)



- BG/Q (Mira) one file per Pset
- 4096 nodes (16 ranks/node)
- TAPIOCA: 16 aggregators per Pset, 16 MB aggregator buffer size
  - Very poor performance from MPI I/O on AoS
  - Up to 2.5 faster than MPI I/O on SoA

HACC-IO – Theta (Cray XC40)

- I/O part of a large-scale cosmological application simulating the mass evolution of the universe with particle-mesh techniques
- Each process manage particles defined by 9 variables (*XX*, *YY*, *ZZ*, *VX*, *VY*, *VZ*, *phi*, *pid* and *mask*)



- Theta from 2048 nodes (16 ranks/node)
- Lustre: 48 OSTs, 16 MB stripe size
- TAPIOCA: 384 aggregators, 16 MB aggregator buffer size
  - AoS, 3.6 MB: 4 time faster than MPI I/O

Number of aggregators

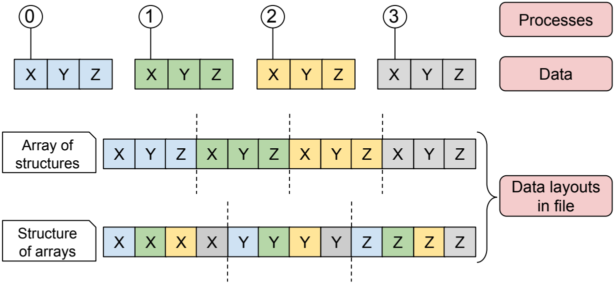
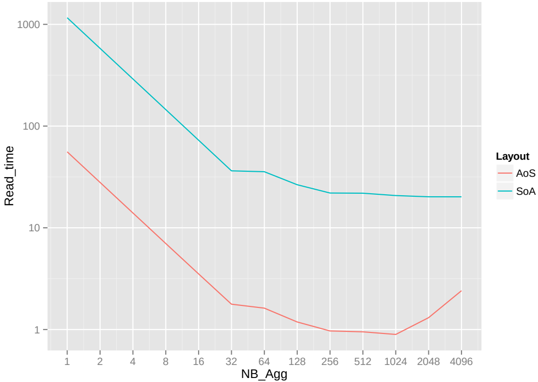


Figure: Array of structure vs structure of array

Trade-Off:

- Many aggregators: lot of I/O rounds
- Few aggregators: I/O Latency dominates

Model Simulation



AoS and SoA aggregation time vs #aggregators

- 4096 nodes
- 9*1MB structures
- 16 ranks/nodes
- 56 I/O streams

Conclusion and Perspectives

Conclusion

- I/O library based on the two-phase scheme developed to optimize data movements
  - Topology-aware aggregator placement
  - Optimized buffering (two pipelined buffers, one-sided communications, block size awareness)
- Very good performance at scale, outperforming standard approaches
- On the I/O part of a cosmological application, up to 15× improvement
- Start modeling I/O performance.