System abstractions to facilitate data movement in supercomputers with deep memory and interconnect hierarchy

François Tessier, Venkatram Vishwanath

Argonne National Laboratory, USA

August 2, 2017

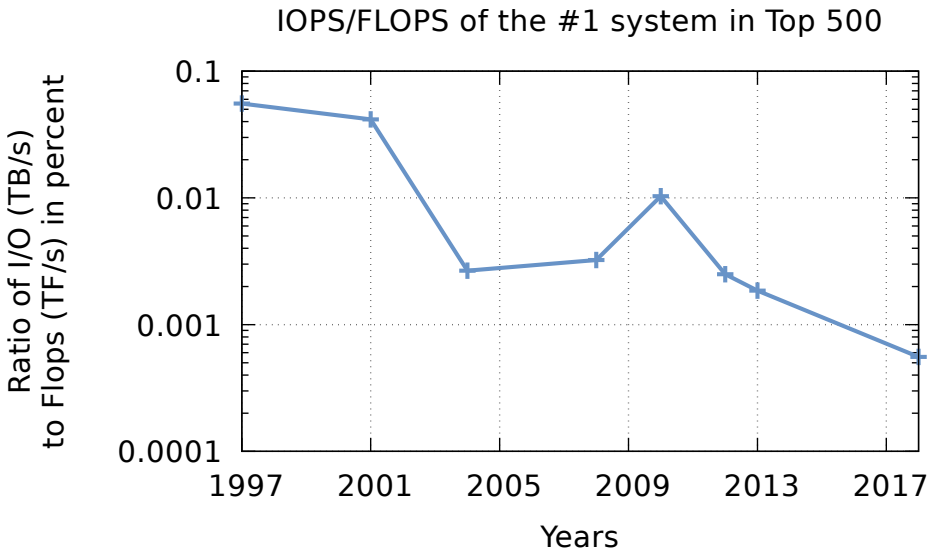
Data Movement at Scale

- Computational science simulation such as climate, heart and brain modelling or cosmology have large I/O needs
  - Typically around 10% to 20% of the wall time is spent in I/O

Table: Example of I/O from large simulations

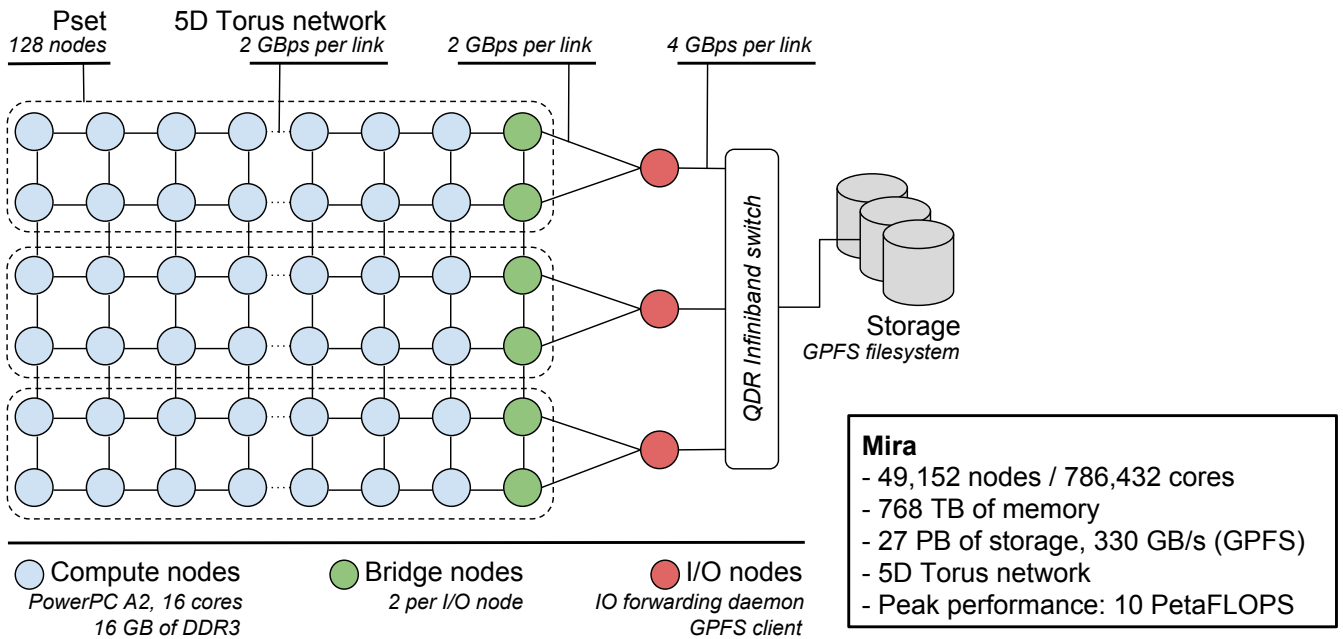
Scientific domain	**Simulation**	**Data size**
Cosmology	Q Continuum	**2 PB / simulation**
High-Energy Physics	Higgs Boson	**10 PB / year**
Climate / Weather	Hurricane	**240 TB / simulation**

- Increasing disparity between computing power and I/O performance in the largest supercomputers



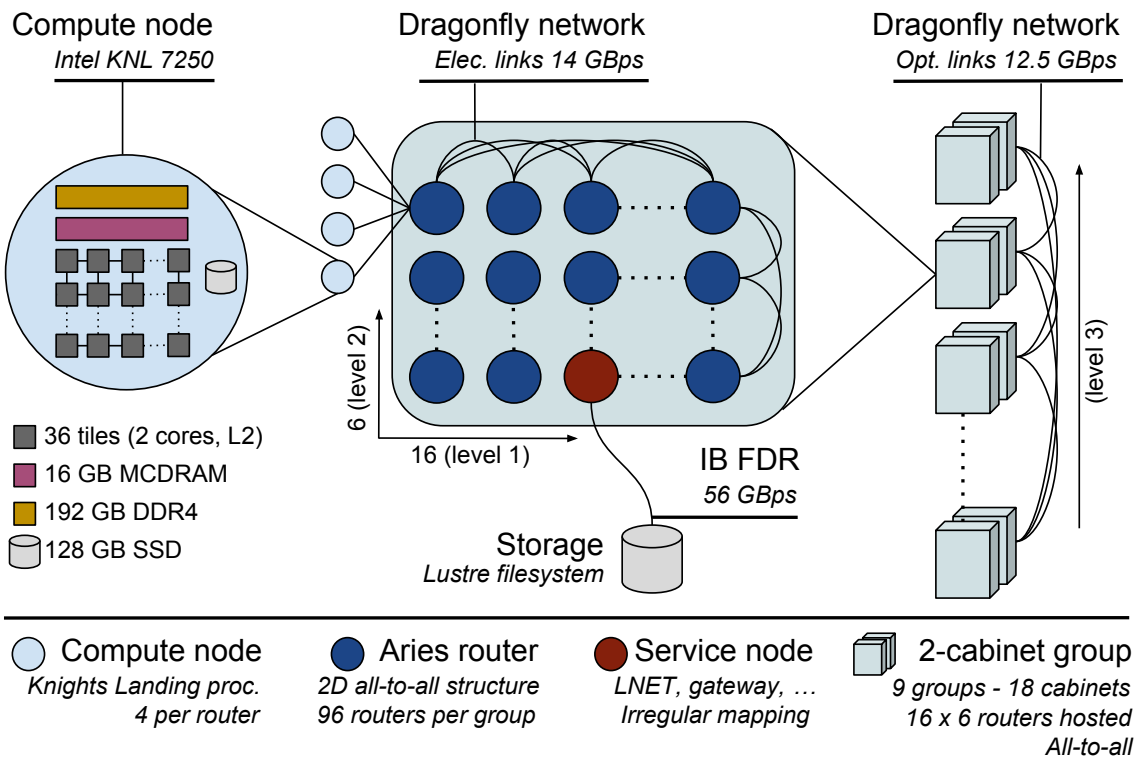
Complex Interconnect Hierarchies

- On BG/Q, data movement needs to fully exploit the 5D-Torus topology for improved performance
- Additionally, we need to exploit the placement of the I/O nodes for performance
- Cray supercomputers have similar challenges with dragonfly-based interconnects together with placement of LNET nodes for I/O



Deep Memory Hierarchies and Filesystem characteristics

- We need to exploit the deep memory hierarchy tiers for improved performance
  - This includes effective ways to **seamlessly** use HBM, DRAM, NVRAM, BurstBuffers, etc.
- We need to leverage filesystem specific features such as OSTs and striping in Lustre, among others.



TAPIOCA

- Library based on the two-phase I/O scheme for topology-aware data aggregation at scale on IBM BG/Q with GPFS and Cray XC40 with Lustre (Cluster'17)
  - Topology-aware aggregator placement
  - Pipelining (RMA, non-blocking calls)
  - Interconnect architecture abstraction
- Outperfoms MPI I/O on the IO kernel of HACC and two data layouts on a Cray XC40 + Lustre architecture

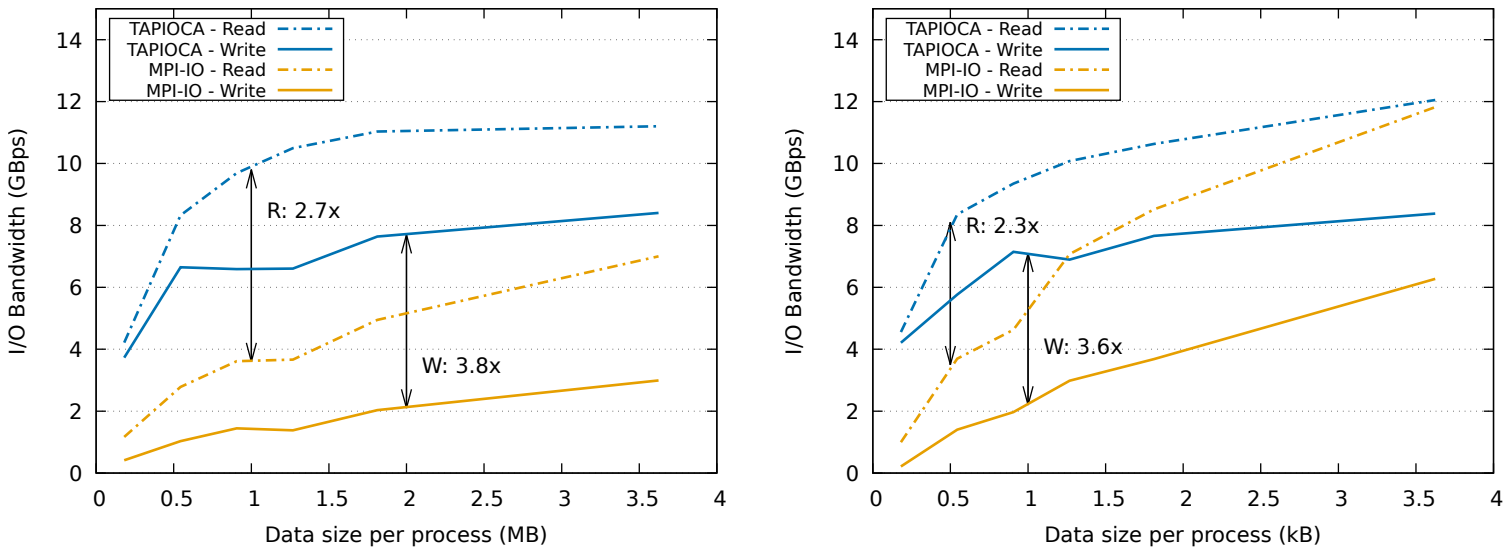
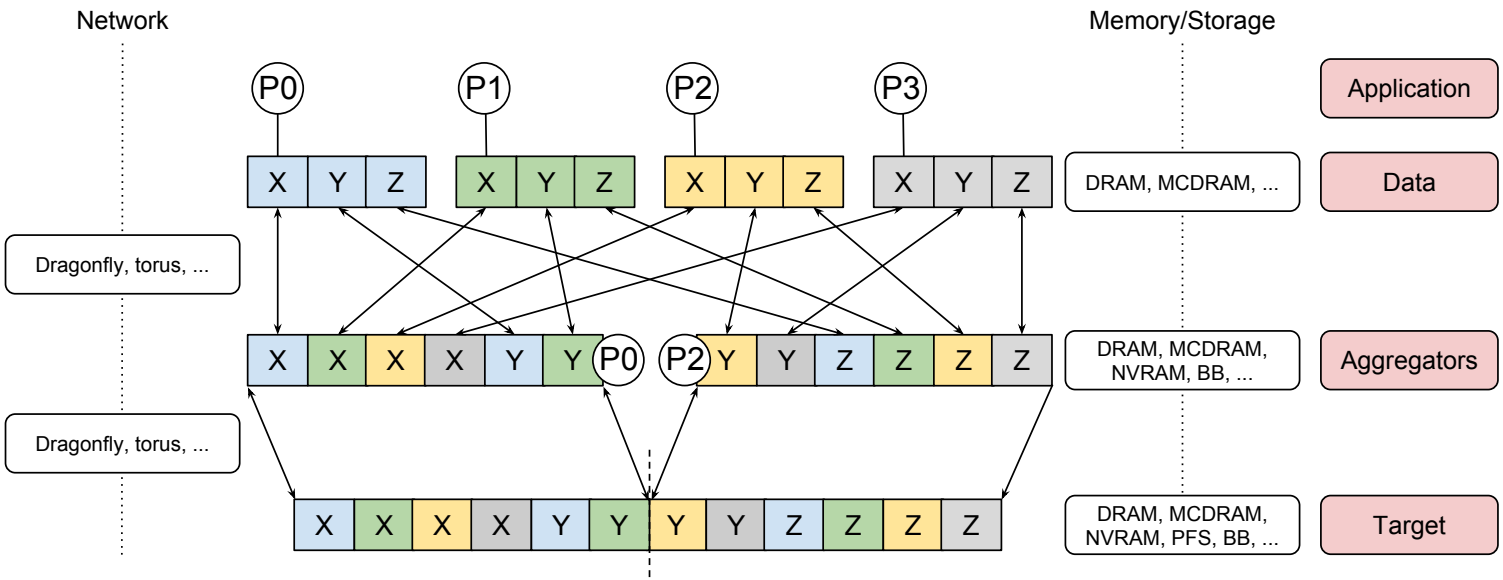


Figure: HACC-IO on 1024 Theta-nodes (16 ranks per node) - 48 OSTs, 8 MB stripe size, 192 aggr, 16 MB buffer size - AoS (left) and SoA (right) data layouts.

TAPIOCA - Ongoing research

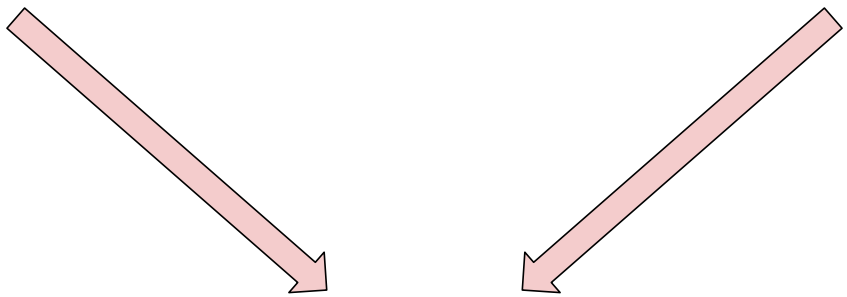
- Move toward a **generic data movement library for data-intensive applications** exploiting deep memory/storage hierarchies as well as interconnect to facilitate I/O, in-transit analysis, data transformation, data/code coupling, workflows, ...



What is the right level of abstraction?

A specific abstraction for every system including the architecture, filesystems, capturing every phase of deployment, relevant software versions, etc.

A generalized abstraction that maps to current and expected future deep memory hierarchies and interconnects (including performance, contention, etc.)



The abstractions and tradeoffs for performant and portable data movement

Abstractions for Interconnect Topology

- Topology characteristics include:
  - Spatial coordinates
  - Distance between nodes: number of hops, routing policy
  - I/O nodes location, depending on the filesystem (bridge nodes, LNET, ...)
  - Network performance: latency, bandwidth
- Need to model some unknowns and uncertainties such as routing, contention

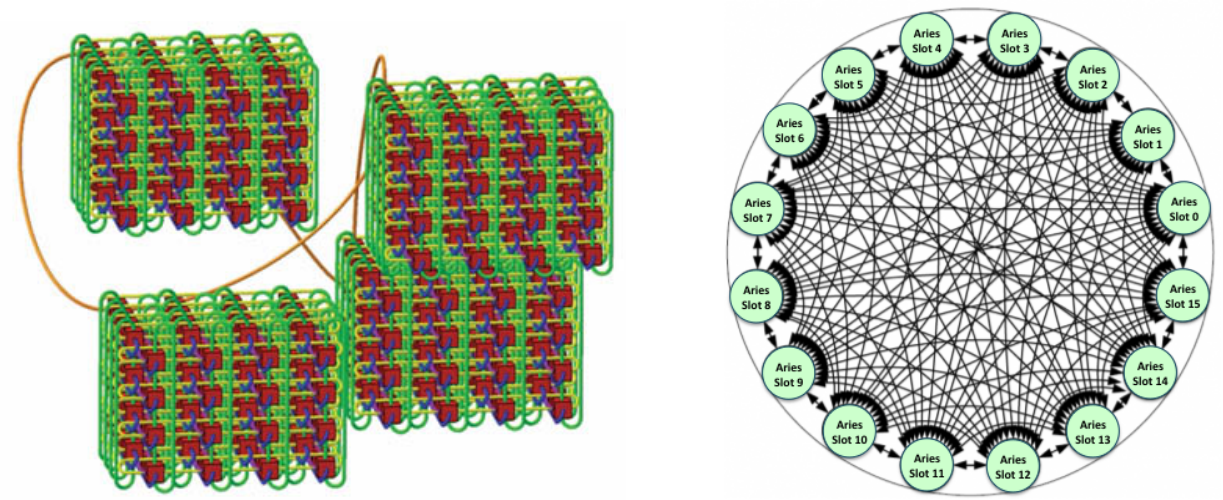
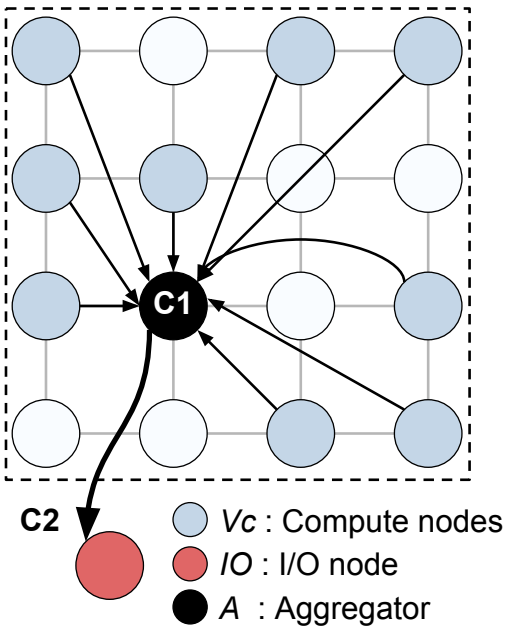


Figure: 5D-Torus on BG/Q and intra-chassis Dragonfly Network on Cray XC30 (Credit: LLNL / LBNL)

Abstractions for Interconnect Topology - Our current approach

- TAPIOCA features a topology-aware aggregator placement
- This approach is based on quantitative information easy to gather: latency, bandwidth, distance between nodes

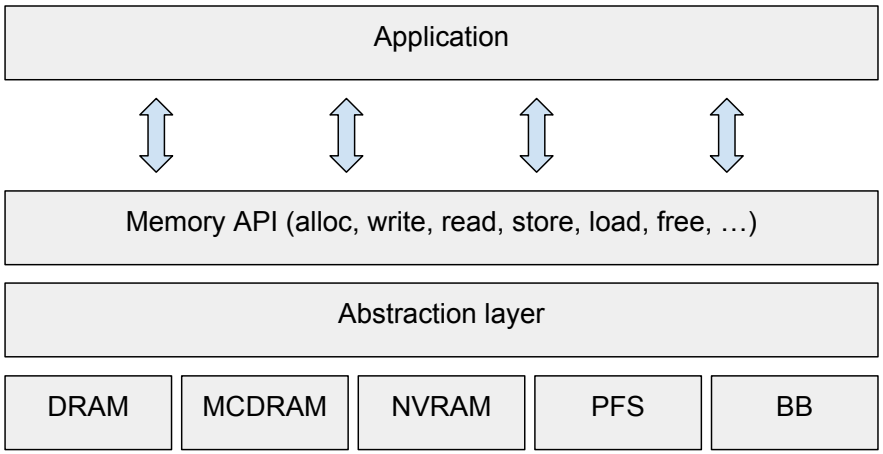
- $\omega(u, v)$: Amount of data exchanged between nodes u and v
- $d(u, v)$: Number of hops from nodes u to v
- l: The interconnect latency
- $B_{i \to j}$: The bandwidth from node i to node j
- $\mathbf{C_1} = \sum_{i \in V_C, i \neq A} \left(l \times d(i, A) + \frac{\omega(i,A)}{B_{i \to A}} \right)$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



- Contention-aware algorithm: static and dynamic routing policies, unknown vendors information, ...

Abstractions for Memory and Storage

- Topology characteristics including spatial location, capacity and distance
- Performance characteristics including bandwidth, latency and support for concurrency
- Access characteristics such as byte-based vs block based
- Persistency



Need to account for application needs in I/O, in-situ vizualisation, in-situ analysis, data transformation, workflows, etc. and map these onto the underlying abstractions for improved performance.

Abstractions for Memory and Storage - Our approach

- Work in progress to move data from the application to any tier of memory or storage
- Some data movements need one or more processes involved
- Scope of memory/storage tiers (PFS vs node-local SSD)
- Model for partitioning data to take advantage of fast memories with smaller capacities

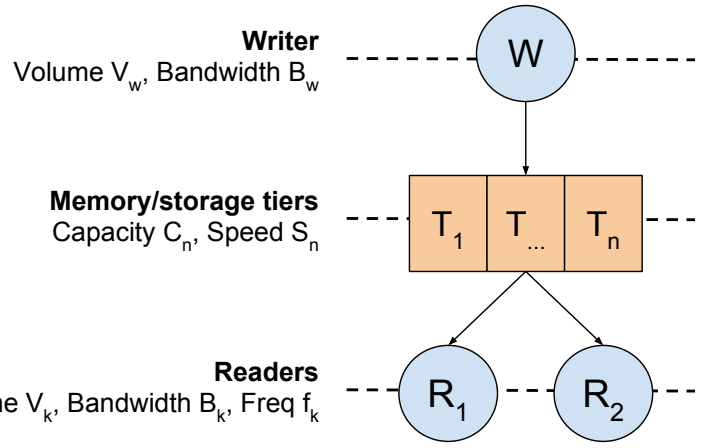


Figure: System with a heterogeneous intermediate memory/storage. One writer and k readers

Listing 1: Function prototypes for memory/storage data movements

```
void   memAlloc           ( void *buff, int64_t buffSize, mem_t mem );
void   memFree            ( void *buff, mem_t mem );
int    mem{Write,Store}   ( void* srcBuffer, int64_t srcSize,
                            void *destBuffer, mem_t mem, int64_t offset );
int    mem{Read,Load}     ( void* srcBuffer, int64_t srcSize,
                            void *destBuffer, mem_t mem, int64_t offset );
void   memFlush           ( void *buff, mem_t mem );
```

Conclusion

- Extending TAPIOCA from an I/O library to a data movement library is challenging
- A compromise between genericity and portability is required
- Our high-level abstraction shows good portable performance on two large-scale systems when considering data locality
- A refined model may have a better impact on performance but leads to an increasingly complex abstraction

Thank you for your attention!

ftessier@anl.gov