Toward portable I/O performance by leveraging system abstractions of deep memory and interconnect hierarchies

François Tessier, Venkatram Vishwanath, Paul Gressier

Argonne National Laboratory, USA

Wednesday 23rd August, 2017
Computational science simulation in scientific domains such as in materials, high energy physics, engineering, 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

<table>
<thead>
<tr>
<th>Scientific domain</th>
<th>Simulation</th>
<th>Data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosmology</td>
<td>Q Continuum</td>
<td>2 PB / simulation</td>
</tr>
<tr>
<td>High-Energy Physics</td>
<td>Higgs Boson</td>
<td>10 PB / year</td>
</tr>
<tr>
<td>Climate / Weather</td>
<td>Hurricane</td>
<td>240 TB / simulation</td>
</tr>
</tbody>
</table>

Increasing disparity between computing power and I/O performance in the largest supercomputers.
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.
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.
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 taking into account
  - The topology of the architecture
  - The data access pattern
- Capture the data model and data layout to optimize the I/O scheduling
- Pipelining (RMA, non-blocking calls) of aggregation and I/O phases
- Interconnect architecture abstraction
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
- $B_{i \rightarrow j}$: The bandwidth from node $i$ to node $j$

\[
\begin{align*}
C_1 &= \sum_{i \in V_C, i \neq A} \left( l \times d(i, A) + \frac{\omega(i, A)}{B_{i \rightarrow A}} \right) \\
C_2 &= l \times d(A, IO) + \frac{\omega(A, IO)}{B_{A \rightarrow IO}}
\end{align*}
\]

- $\text{TopoAware}(A) = \min (C_1 + C_2)$

- Contention-aware algorithm: static and dynamic routing policies, unknown vendors information such as routing policy or data distribution among I/O nodes, ...
Outperforms MPI I/O on the IO kernel of HACC and two data layouts on a Cray XC40 + Lustre and BG/Q + GPFS

- **HACC**: Large-scale simulation of the mass evolution of the universe with particle-mesh techniques (A particle is defined by 9 variables).
- 1024 nodes, 16 ranks per node
- Best PFS configuration for MPI I/O
  - Lustre: 48 OST, 8 MB stripe size, 192 aggregators
  - GPFS: 16 aggregators per Pset (128 aggr), 16 MB buffer size

![Graphs showing I/O bandwidth comparison between TAPIOCA and MPI-IO for different data layouts.](image)

<table>
<thead>
<tr>
<th>Data size per process (MB)</th>
<th>TAPIOCA - Read</th>
<th>TAPIOCA - Write</th>
<th>MPI-IO - Read</th>
<th>MPI-IO - Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2</td>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.5</td>
<td>6</td>
<td>6</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2.5</td>
<td>10</td>
<td>10</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3.5</td>
<td>14</td>
<td>14</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Figure**: Array of structures data layout
Outperforms MPI I/O on the IO kernel of HACC and two data layouts on a Cray XC40 + Lustre and BG/Q + GPFS

- HACC: Large-scale simulation of the mass evolution of the universe with particle-mesh techniques (A particle is defined by 9 variables).
- 1024 nodes, 16 ranks per node
- Best PFS configuration for MPI I/O
  - Lustre: 48 OST, 8 MB stripe size, 192 aggregators
  - GPFS: 16 aggregators per Pset (128 aggr), 16 MB buffer size

![Graphs showing I/O bandwidth comparison](image1.png)

(a) Cray XC40 + Lustre
(b) BG/Q + GPFS

Figure: Structure of arrays data layout
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, ...
Abstractions for Memory and Storage

- Topology characteristics including spatial location, distance
- Performance characteristics: bandwidth, latency, capacity
- Access characteristics such as byte/block-based, concurrency
- Persistency

### Listing 1: Function prototypes for memory/storage data movements

```c
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 );
```

- Work in progress with open questions
  - Blurring boundary between memory and storage (MCDRAM, 3D XPoint memory, ...)
  - Some data movements need one or more processes involved at destination (RMA window, flushing thread, ...)
  - Scope of memory/storage tiers (PFS vs node-local SSD)
  - Data partitioning to take advantage of fast memories with smaller capacities
Conclusion

- TAPIOCA, an optimized data-movement library incorporating
  - Topology-aware aggregator placement
  - Optimized data movement with I/O scheduling and pipelining
  - Hardware abstraction insuring performance portability
- Performance portability on two leadership-class supercomputers: Mira (IBM BG/Q + GPFS) and Theta (Cray XC40 + Lustre)
  - Same application code running on both platforms
  - Same optimization algorithms using an interconnect abstraction
- Promising preliminary results with memory/storage abstraction
- An appropriate level of abstraction is hard to define
  - Specific abstraction for every system including the architecture, filesystems, capturing every phase of deployment, relevant software versions, ...
  - Generalized abstraction that maps to current and expected future deep memory hierarchies and interconnects
- Future work: Come up with a model helping to take smart decision for data movement

Acknowledgments

- Argonne Leadership Computing Facility at Argonne National Laboratory
- DOE Office of Science, ASCR
- Proactive Data Containers (PDC) project
Thank you for your attention!
ftessier@anl.gov
MPI-IO and TAPIOCA - Data layout