Memory-Conscious Collective I/O for Extreme Scale HPC Systems

Yin Lu, Yong Chen, Texas Tech University
Rajeev Thakur, Argonne National Laboratory
Yu Zhuang, Texas Tech University
A wide range of HPC applications, simulations, and visualizations[1]
Many applications are increasingly data intensive[2]

Many simulations/applications process $O(1\text{TB}-100\text{TB})$ in a single run

Application teams are projected to manipulate $O(1\text{PB}-10\text{PB})$ on exascale systems

Data requirements for representative INCITE applications at ALCF

<table>
<thead>
<tr>
<th>PI</th>
<th>Project</th>
<th>On-Line Data</th>
<th>Off-Line Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamb, Don</td>
<td>FLASH: Buoyancy-Driven Turbulent Nuclear Burning</td>
<td>75TB</td>
<td>300TB</td>
</tr>
<tr>
<td>Fischer, Paul</td>
<td>Reactor Core Hydrodynamics</td>
<td>2TB</td>
<td>5TB</td>
</tr>
<tr>
<td>Dean, David</td>
<td>Computational Nuclear Structure</td>
<td>4TB</td>
<td>40TB</td>
</tr>
<tr>
<td>Baker, David</td>
<td>Computational Protein Structure</td>
<td>1TB</td>
<td>2TB</td>
</tr>
<tr>
<td>Worley, Patrick H.</td>
<td>Performance Evaluation and Analysis</td>
<td>1TB</td>
<td>1TB</td>
</tr>
<tr>
<td>Wolverton, Christopher</td>
<td>Kinetics and Thermodynamics of Metal and</td>
<td>5TB</td>
<td>100TB</td>
</tr>
<tr>
<td>Washington, Warren</td>
<td>Climate Science</td>
<td>10TB</td>
<td>345TB</td>
</tr>
<tr>
<td>Tsigelny, Igor</td>
<td>Parkinson's Disease</td>
<td>2.5TB</td>
<td>50TB</td>
</tr>
<tr>
<td>Tang, William</td>
<td>Plasma Microturbulence</td>
<td>2TB</td>
<td>10TB</td>
</tr>
<tr>
<td>Sugar, Robert</td>
<td>Lattice QCD</td>
<td>1TB</td>
<td>44TB</td>
</tr>
<tr>
<td>Siegel, Andrew</td>
<td>Thermal Striping in Sodium Cooled Reactors</td>
<td>4TB</td>
<td>8TB</td>
</tr>
<tr>
<td>Roux, Benoit</td>
<td>Gating Mechanisms of Membrane Proteins</td>
<td>10TB</td>
<td>10TB</td>
</tr>
</tbody>
</table>

Source: R. Ross et. al., Argonne National Laboratory
Motivation – Decreased Memory & BW per core at Exascale

- Neither available memory capacity nor memory bandwidth scales by the same factor as the total concurrency
- The disparity of growth between memory and concurrency indicates the average memory and bandwidth per core even drop in exascale system

### Expected Exascale Architecture Parameters and Comparison with Current Hardware

<table>
<thead>
<tr>
<th>System Parameter</th>
<th>2011</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Peak</td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>≤20 MW</td>
<td>3</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225 K</td>
<td>1B</td>
<td>4444</td>
</tr>
<tr>
<td>Node Performance</td>
<td>0.125 Tf/s</td>
<td>1 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td>Node Concurrency</td>
<td>12 CPUs</td>
<td>1000 CPUs</td>
<td>83</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>100 GB/s</td>
<td>66</td>
</tr>
<tr>
<td>System Size (nodes)</td>
<td>18700</td>
<td>1000000</td>
<td>50</td>
</tr>
<tr>
<td>I/O capacity</td>
<td>15 PB</td>
<td>300 PB – 1000 PB</td>
<td>20 - 67</td>
</tr>
<tr>
<td>I/O Bandwidth</td>
<td>0.2 TB/s</td>
<td>20 – 60 TB/s</td>
<td>10 -30</td>
</tr>
</tbody>
</table>

Available memory exhibits imbalance among compute nodes
Available memory per node can vary significantly at an extreme scale
These projected constraints present challenges for I/O solution at exascale including collective I/O
Collective I/O optimizes I/O accesses by merging small & noncontiguous I/O requests into large & contiguous ones, removing overlaps, reducing calls.  
Remains critical for extreme scale HPC systems.  
Performance can be significantly affected under memory pressure.

Motivation – Collective I/O Performs with Limited Memory
Objective: to design and develop collective I/O with awareness of memory capacity, variance, off-chip bandwidth

Contributions

• Identified performance & scalability constraints imposed by memory pressure issue
• Proposed a memory-conscious strategy to conduct collective I/O with memory-aware data partition and aggregation
• Prototyped and evaluated the proposed strategy with benchmarks
• Memory-conscious strategy can be important given the significance of collective I/O and substantial memory pressure at extreme scale
• Towards addressing challenges of an exascale I/O system
Memory-Conscious Collective I/O (cont.)

- Contains four major components
  - *Aggregation Group Division* divides the I/O requests into separated groups
  - *I/O Workload Partition* partitions the aggregate access file region into contiguous file domains
  - *Workload Portion Remerging* rearranges the file domains considering the memory usage of physical nodes
  - *Aggregators Location* determines the placement of aggregators for each file domain
- Applications, library, parallel file systems
To avoid global aggregation and reduce memory requirements
- The global data shuffling traffic increases the memory pressure on aggregators and leads to off-chip memory bandwidth contention
- Divides the I/O workloads into non-overlapping chunks guided by the optimal group message size $Msg_{group}$
- Aggregation groups perform their own aggregation in parallel
- Limit one node in one group to reduce communications
I/O Workload Partition within Aggregation Group

- Analyzes all I/O accesses within each aggregation group
- Workload dynamically partitioned into distinct domains

Dynamical Workload Partition Algorithm

Bisect
{
  Compute root weight $Root_{wgh}$;
  If $Root_{wgh} > Msg_{ind}$
    Bisect_tree(root);
}

Bisect_tree(vertex)
{
  Create two children for the vertex;
  Split the region belonging to this vertex in half;
  The compute nodes with associated messages in this region are partitioned accordingly into two sets;
  Assign each set to one child;
  For each child
  {
    Compute child weight $Child_{wgh}$;
    If $Child_{wgh} > Msg_{ind}$
      Bisect_tree(child);
  }
}
- Reorganizes the file domains considering the memory consumption for the aggregation
- File domain merged with the domain nearby (still a contiguous file domain)
- To aggregate I/O requests based on available memory & saturate B/W
- Aggregation and file domain partition with memory-conscious strategy
- Compared against conventional evenly partition
- Avoid iterations of carrying out collective I/O
- Limits the number of aggregators in a node
- Identifies the node with required available memory and minimizes communications and B/W requirement

Aggregators Location

Detect physical nodes of all processes within one file domain

Node has less than $N_{ab}$ aggregators

Add the node into candidate list

Checked all the nodes

Identify node with maximum $\text{Mem}_{avl}$

$\text{Mem}_{avl}$ larger than minimum memory requirement

Select the corresponding process as the aggregator for this file domain

Coalesce the file domain with the file domain nearby
Evaluation and Performance Analysis

- **Experimental Environment**
  - 640-node Linux-based cluster test bed with 600TB Lustre file system
  - Each node contains two Intel Xeon 2.8 GHz 6-core processors with 24 GB main memory
  - Nodes connected with DDR InfiniBand interconnection
  - Prototyped with MPICH2-1.0.5p3 library

- **Three well-known MPI-IO benchmarks selected for evaluation & comparison against normal collective I/O**
  - coll_perf from ROMIO software package
  - IOR developed at Lawrence Livermore National Laboratory
  - MPI-IO Test developed at Los Alamos National Laboratory
Experimental Results of coll_perf Benchmark

- 120 MPI processes used to write and read a 32 GB file on Lustre file system
- Evicted cached data with memory flushing after write phase
- Average performance for write and read tests were 34.2% and 22.9% respectively
Experimental Results of IOR Benchmark

- Tests carried out with 120 and 1080 processes
- Maximum write and read improvement up to 121.7% and 89.1% respectively
- Write tests performance improvements were more sensitive to the new strategy
- Average performance for write and read tests were 24.3% and 57.8% respectively
Experimental Results of mpi-io-test Benchmark

- Performance increased at a relatively moderate rate compared with IOR
- Average performance improvements for read and write tests were 32.9% and 14.6% at 120 cores
- Average performance improvements for read and write tests were 29.8% and 24.1% at 1080 cores
Conclusion

- Exascale HPC systems near the horizon
  - Decreased memory capacity per core, increased memory variance, and decreased bandwidth per core are critical challenges for collective I/O
- Studied the constraints at projected exascale systems
- Proposed a new memory-conscious collective I/O strategy
  - Restricts aggregation data traffic within groups
  - Determines I/O aggregation dynamically
  - With memory-aware data partition and aggregation
- Experiments performed on MPICH2+Lustre
- Evaluation results confirmed the proposed strategy outperformed existing collective I/O given memory constraints
### Future Work

- An I/O system aware of memory constraints can be critical on current petascale and projected exascale systems.
- The memory-conscious collective I/O strategy:
  - Retains benefits of collective I/O
  - Reduces memory pressure
  - Alleviates off-chip bandwidth contention
- Plan to further investigate and reduce communication costs
- Plan to investigate the leverage of SCM, burst buffer, caching
Thank You.

For more information please visit

http://discl.cs.ttu.edu/