ILLINOIS INSTITUTE OF TECHNOLOGY

PROJECT PROPOSAL

On

Interference-aware Application Scheduler for Supercomputers

CS 550: Advanced Operating Systems

Authors:

Antoine Tehio
A20432639

Rhea Shetty
A20432742

1. Introduction

Many scientific applications running in HPC systems deal with large amounts of data. On the other hand,

the computation power of current multicore/manycore architectures has followed a faster trend compared to data access performance improvement (latency and bandwidth). This gap is very unlikely to be overcome in the near future. Thus, accessing large amount of data becomes a bottleneck for many applications [1]. This movement of large amount of data constantly in an application can cause severe degradation to the overall performance. Introducing intermediate levels of memory with higher bandwidth and faster data access between the applications and the disk improves performance significantly. Apart from the local RAM, each node can have a NVMe (non-volatile memory express). Every node can share burst buffer nodes, that are made up of SSDs, to increase I/O performance and caching [2]. Interference-aware application scheduler is a technique to manage this shared resource. It is the interface between the applications and the SSDs.

2. Background Information

DEP is the first paradigm enabling users to identify and handle data-intensive operations separately. It can significantly reduce costly data movement and is better than the existing execution paradigms for data-intensive applications. Data nodes can provide simple data forwarding without any data size reduction, but the idea behind data nodes is to let the data nodes conduct the decoupled data-intensive operations and optimizations to reduce the data size and movement. Figure 1 depicts this architecture. Active storage [5], [6], [7] leverages the computing capability of storage nodes and performs certain computation to reduce the bandwidth requirement between storage and compute nodes. Active disks [8] and smart disks integrate a processing unit within disk storage devices and offload computations to embedded processing unit. Reserving burst buffers allocation through the main system batch scheduler does not account for interference on the buffering destination. The scheduler needs to be aware of concurrent accesses and avoid interference. In the case when all buffer capacity is already allocated, any application trying to gain access will be put in queue.

3. Problem Statement

Every application sends its I/O requests to the burst buffer nodes. This causes interference of the requests. Therefore, there is a need for a scheduler that is aware of the interference caused by the applications. To do so, the scheduler has to be aware of the phases of each I/O operation of each of the applications. Using the information of the I/O phases, the scheduler then needs to assign jobs to burst buffer nodes based on the state of the shared nodes.

4. Related Work

There have been some development of advanced I/O libraries, such as Hierarchical Data Format (HDF), Parallel netCDF (pnetCDF) and Adaptable IO System (ADIOS), that all provide high level abstractions, map the abstractions to the I/O and try to complement parallel programming models, such as Message Passing Interface (MPI) [4] and others, by managing I/O activities. Hermes [3] tries to bridge the I/O gap by adding more layers of memory that work coherently. Harmonia [10] is an approach that supports multi-platform shared memory access. It improves I/O performance significantly as compared to state-of-the-art buffering management solutions.

5. Proposed Solution

To solve the I/O bottleneck problem and improve the utilization of burst buffers, an interference-aware application scheduler has been proposed. This scheduler acts as an application orchestrator. It accepts as input, jobs from the applications and the collection of the buffer nodes. The jobs consist of I/O phases of each of the applications. The application orchestrator would check the state of the buffer nodes and depending on the I/O requests, schedule jobs for idle burst buffer nodes. This schedule is given as output to the DataWarp. The following are the proposed specifications of the solution:

- Application scheduler would be a multi-process program in MPI
- Queue for incoming jobs would be implemented using NATS Server
- The states of the burst buffer nodes would be stored using a global distributed hashmap like memcached
- Schema mapping I/O phases and burst buffers nodes would be the output

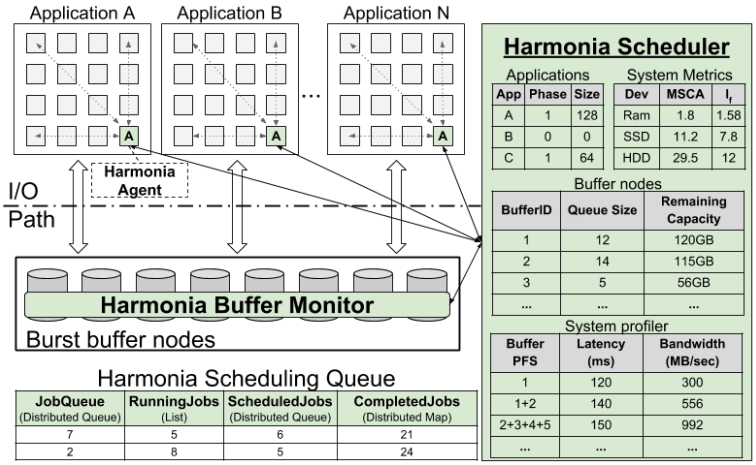

Figure 1, Harmonia An Interference-Aware Dynamic I/O Scheduler for Shared Non-Volatile Burst Buffers

6. Team Responsibilities

The implementation of functionalities of application orchestrator is divided between the team members as follows:

Antoine would be focusing on the application registration table. The application registration table would be used to keep track of all running applications. It would also record the phase that the running applications are in (i.e., I/O or not).

Rhea would take care of the buffer status table in the first place. It keeps track of how busy a node is and what is the remaining capacity on the storage devices of that node.

The scheduling queue and algorithm used for scheduling the I/O phases to burst buffer nodes would be handled by both team members together. The orchestrator schedules I/O phases on buffer nodes using dynamic programming (DP) as mentioned in Harmonia. The algorithm would be able to maximize or minimize the objective of each policy by expressing different constraints and cost functions to the DP algorithm.[2]

7. Conclusions

In order to deal with HPC systems interferences, a scheduler is needed to manage data movements. The application scheduler will be an orchestrator for applications and their use of data. This will optimise bottleneck and interferences. Optimising the use of data storage will maximise the speed potential of the system. To do so every request will go to the orchestrator and will be schedule on their phases and the availability of the burst buffer nodes. This implementation improves efficiency of the system by ensuring the I/O processes are faster and optimized.

8. Additional Resources

8.a) Timeline

Week	*Task*
1	Study relevant papers, gather more info, compile all together
2	Use case algorithmic design
3	Code the new programming model
4	Present intermediate report (midterm)
5	Code the new programming model / debug
6	Do the experimental evaluation
7	Write the final report
8	Write the final presentation

8.b) Deliverables

- One final report in PDF form.
- One final Powerpoint presentation.
- Source code.

References

[1] A. Choudhary, W. Liao, K. Gao, A. Nisar, R. Ross, R. Thakur, and R. Latham, “Scalable I/O and analytics,” *J. Phys. Conf. Ser.*, vol. 180, p. 012048, Jul. 2009.

[2] Anthony Kougkas, Hariharan Devarajan, and Xian-He Sun, “Harmonia: An Interference-Aware Dynamic I/O Scheduler for Shared Non-Volatile Burst Buffers.” Technical Report. Illinois Institute of Technology 2017.

[3] Anthony Kougkas, Hariharan Devarajan, and Xian-He Sun, “Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system,” in Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing. ACM, 2018, pp. 219–230.

[4] S. Kumar and M. Blocksome, “Scalable MPI-3 . 0 RMA on the Blue Gene / Q Supercomputer,” pp.7–12.

[5] C. Chen and Y. Chen, “Dynamic active storage for high performance I/O,” *Proc. Int. Conf. Parallel Process.*, pp. 379–388, 2012.

[6] J. Piernas, J. Nieplocha, and E. J. Felix, “Evaluation of active storage strategies for the lustre parallel file system,” *Proc. 2007 ACM/IEEE Conf. Supercomput. (SC ’07)*, no. 1, 2007.

[7] S. W. Son, S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, P. Kumar, W. K. Liao, and A. Choudhary, “Enabling active storage on parallel I/O software stacks,” *2010 IEEE 26th Symp. Mass Storage Syst. Technol. MSST2010*, 2010.

[8] J. Saltz, “Active Disks : Programming Model , Algorithms and Evaluation.”