Burst Buffers Simulation in Dragonfly Network

Jian Peng
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
jpeng10@hawk.iit.edu

Michael Lang
Los Alamos National Laboratory
Los Alamos, NM, USA
mlang@lanl.gov

Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
iraicu@iit.edu

***Abstract*—The dragonfly network has been adopted in new generation supercomputers, meanwhile burst buffers are also placed in this network. Residing in the compute node network, and using solid state drives (SSD), burst buffers are able to bring a significant I/O performance boost compared with traditional external hard disk drive (HDD) storage system. Though it has been proven that burst buffers did manage to enhance the IO bandwidth , there are still a lot of questions remain unexplored. Some questions are like if there are bottlenecks of fully leveraging burst buffer bandwidth, and if there will be interference when IO traffic and compute traffic are both present in the dragonfly network. However, due to the very large system scale and limited machine access time, it is usually too expensive and time-consuming to change system setup to study these problems. So, in this paper, we developed a simulator based on CODES/ROSS framework, which simulates a supercomputer with this "burst buffer/ dragonfly network" structure and used it as the base-stone of some further study of above problems. The highlights of this work are: First, we developed this system simulator based on CODES/ROSS framework and Aries network configuration of Cray XC40 supercomputers. Second, we validated the simulator with IOR test results of Trinity, a Cray XC40 supercomputer at Los Alamos National Laboratory(LANL). At last, we used some Darshan IO traces in the simulator to observe IO interference when applications access burst buffers .**

Keywords*—*burst buffer, dragonfly network, CODES/ROSS, bottleneck, simulator, Trinity.

I. Introduction

Two important components of High-performance computing (HPC) systems are interconnect network of compute nodes and external storage system. These two parts are naturally separate because of huge gaps in their latency, scale and cost. Though tons of work have been done to enhance performance of both sides, the gaps still exist. However, as dragonfly network topology starts to be adopted in design of new HPC systems and the cost of solid-state storage is lower enough to be used in large scale, placing SSD storage systems in compute node network turns out to be a feasible solution of these gaps.

In a dragonfly network, a router has a high radix which reduces the diameter of the network. The Cray Aries network[1] used in Cray XC40 supercomputers is an example network based on dragonfly network topology. One Aries router can be linked up to 30 intra-group and 10 inter-group Aries routers. Thus, the high local radix not only can provide enough mount points for both compute nodes and storage nodes, but also reduce the network latency by using adaptive routing algorithm. The burst buffer nodes residing in Aries are implemented as Datawarp nodes in Cray XC40. These burst buffer nodes are a storage tier that uses flash as storage media. Since they are substantially faster than hard-disk-based parallel file system(PFS), they can play a role of intermediate buffer between compute node and PFS and thus fill the latency gap between memory of compute nodes and hard disks of PFS.

In order to simulate a exascale system with both dragonfly network and burst buffers, we have implemented our simulation using CODES (Co-Design of Multilayer Exascale Storage Architectures)[2] and ROSS (Rensselaer Optimistic Simulation System)[3] simulation frameworks. ROSS is a parallel discrete event driven simulator which uses time warp protocols to simulate discrete events in parallel. ROSS has been proven to support a simulation of 50 million-node dragonfly network and reaches a peak rate of 1.3 billion events per second. CODES is a highly parallel simulation framework developed by Argonne National Laboratory (ANL) and Rensselaer Polytechnic Institute(RPI). It is built on top of ROSS and provides various network models such as torus and dragonfly network.

Darshan is a light weight profiling tool developed that can be used to characterize I/O-load at pet-scale. It can identify and record I/O operations of an application[4]. In addition to run-time libraries, Darshan also provides utility libraries used to parse I/O logs generated in run time. In the newest version, Darshan is able to recognize multiple I/O operation types such as POSIX, MPI-IO, HDF5 and so on. In a log file, information such as total read and write numbers, total I/O time is recorded. Given the detailed characterization of the I/O behavior of an application, it is possible to replay I/O traces to simulation an applications I/O behavior.

Trinity[5] is a Cray XC40 architecture supercomputer that is installed at Los Alamos National Laboratory (LANL). In 2016, Trinity ranked the 7th most powerful supercomputer in the world. In the year of 2017, Trinity has gone through Stage II. By the time of this paper is written, Trinity just finished the merge, which combines two halves of the system together. One half contains 9,384 compute nodes with Intel Haswell processors and the other has 9,984 Intel Knights Landing (KNL) processors. After the merge, Trinity now has $>$1,900 compute nodes, 576 burst buffer (Datawarp) nodes, which will probably let Trinity get a higher rank in TOP500[6] supercomputer ranking.

The contributions of this paper are as follows:

- We added burst buffer simulation to the CODES dragonfly network model. The current dragonfly network simulation in CODES only models two entities: compute node and router. Since burst buffer nodes

are placed inside the same dragonfly network with compute nodes and routers, we must add this new entity into the model. The reason that we are not able to reuse a compute node to represent a burst buffer node is a burst buffer node has different parameters and behaviors from a compute node. To ensure the simulation accuracy, the burst buffer node model should be distinguished from the existing compute node model.

- We validated our simulator with IOR test results of Trinity at LANL. Before we can carry on further studies with our simulator, we must validate the simulator to make sure of its accuracy. The validation process is usually hard and time-consuming. So is the validation in our work. We developed an IOR workload generator and validated our simulator with IOR benchmarking results collected from Trinity after the merge. The simulation error rate we got is less than 15
- At the end, we used some Darshan IO traces to show IO interference when burst buffers are shared across applications. On Trinity at LANL, researchers have observed a strong network congestion when more than one applications access burst buffer nodes. In order to further study the issue, we need to generate a similar situation in our simulator. Therefore, we modified the Darshan IO workload generator in CODES to support Darshan IO logs above version 3.0. Then we fed the simulator with Darshan IO traces collected from Cori, a supercomputer with similar architecture at NERSC. From the results, we can observe an obvious IO interference between applications.

The rest of this paper is organized as follows: In Section II, some background information and motivation of this work are given. In Section III, we present our modeling and simulation details. Experiment results and conclusions are given in Section IV. Section V is related work and our future work. Section VI is related work.

II. Background & Motivation

This section provides some background knowledge and our motivation of this work. The background information is mostly related to Trinity.

A. Trinity system overview

In this paper, our simulating target is Trinity, a Cray XC40 architecture supercomputer at LANL. The overview of whole system is shown in Figure. 1. Trinity uses dragonfly network to connect all compute nodes.The most significant feature of Dragonfly network is the all-to-all pattern interconnections among routers. In order to connect all routers manageably and efficiently, links among routers are grouped into 3 levels: system level (group level), cabinet level and chassis level. On system level, 57 two-cabinet groups are connected with optical links. This optical link is called inter-group link. On cabinet level, each cabinet contains 6 chassis. All chassis in the same cabinet are connected with backplane electrical links which can be called inter-chassis link. The same type of interconnection is also used to connect 16 Aries routers in a chassis on the Chassis level, which is called intra-chassis link. Each Aries

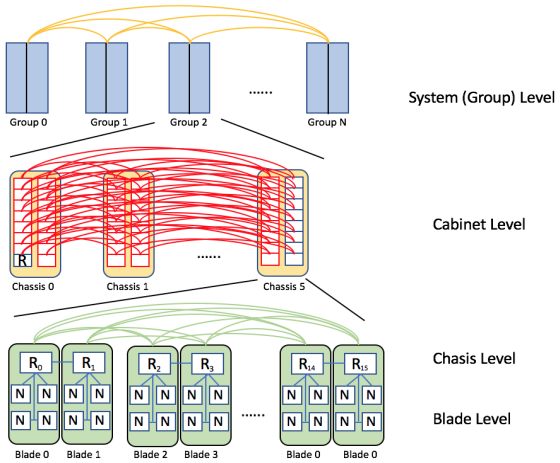

Fig. 1: Trinity system overview

router device and different kinds of nodes are placed onto one blade. One the Blade level, PCIe 3.0 connections are used to link Aries router and nodes. This on-blade link is called intra-blade link.

B. Cray Aries router

Aries router is a key component in CrayXC40's interconnection network. The inside layout of an Aries router is modeled in Figure. 2. In an Aries router, there are totally 48 tiles or ports used to switch flits. These 48 tiles are categorized to 4 groups. 10 tiles are used for inter-group optical links, which are colored as blue. Maximum bidirectional speed of each these tiles is 4.7GB/s. Another 15 green tiles are used to provide intra-chassis connection ports for the rest 15 routers in the same chassis. . The maximum bandwidth is 5.25GB/s in both directions. Another 15 black tiles link the rest 5 chassis inside the same cabinet. To achieve a higher bandwidth, each inter-chassis link occupies 3 this kind of tiles which sums up 15.75GB/s in each direction. The rest 10 tiles are used for connection with nodes on the same blade with the router.

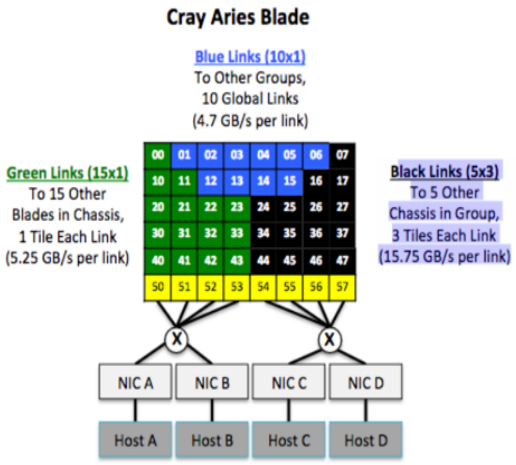

Fig. 2: Aries router layout

C. Cray Datawarp node

Datawarp[7] is Cray's implementation of burst buffers in CrayXC series supercomputers. Burst buffers, as well as Parallel file system IO nodes, are integrated into Datawarp nodes. Datawarp architecture can be presented in Figure. 3. Before launching a job on compute nodes, the workload manager or scheduler on a server node configures both compute nodes and Datawarp nodes. When an application issues IO requests, requests are forwarded to DVS server process on a Datawarp node. DVS, aka. data virtualization service is Cray's IO processing software. After receiving IO requests from DVS server, Datawarp service process checks Datawarp configuration of the job. Then Datawarp service process decides if the requests should be forwarded to parallel filesystem or to local burst buffers.

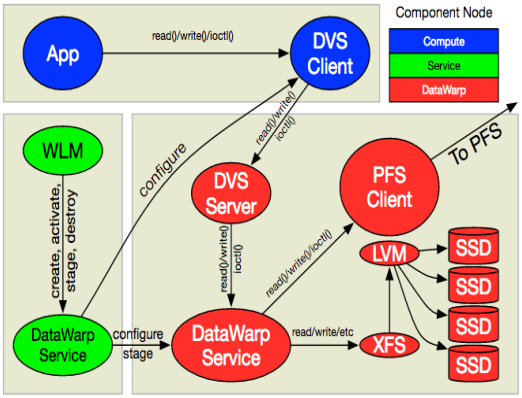

Figure 2. DataWarp software architecture component diagram.

Fig. 3: Aries router layout

D. CODES/ROSS simulation framework

CODES is a network simulation framework developed by Argonne National Laboratory (ANL) based on ROSS. ROSS is a massive parallel event driven simulator developed by Rensselaer Polytechnic Institute (RPI). In CODES, multiple network typologies are modeled in model-net layer. Network routing algorithms are also implemented in the model-net layer. Each node in a network is modeled by a logical process (LP). Messages between LPs are passed in form of events with MPI. All LPs and events are managed by ROSS's optimistic scheduler. In addition to simulation frame implementation, CODES provides some different workload generators. For example, network traffic workload generator could parse DUMPI traces to generate workload simulating applications' MPI traffic. A Darshan IO workload generator is also implemented which can read applications' IO traces generated by Darshan instrumentation.

E. Motivation

Though burst buffer architecture has been already adopted in production systems, how to achieve best performance from burst buffers still remains a critical issue. One problem observed on Trinity is interference between applications when accessing Datawarp nodes. In order to better understand and solve the issue, we propose burst buffer simulation in dragonfly network. After validating the simulator, we can use applications' Darshan IO traces to replay this issue and try to find solutions.

III. Simulation Design

In this section, we present our simulation design from following aspects: group modeling, network configurations and CODES application layer.

A. Trinity group modeling

A group is defined as a 2-cabinet rack in Trinity which can be modeled as Figure.4. There are 4 types of entities in a Trinity node group. LN nodes represent Lustre Network Routers. These routers are connected to the compute network with Aries router as well as to the external Lustre parallel filesystem. LN nodes play the role of forwarding compute nodes' IO requests to PFS. 2 LN nodes are connected to one Aries device. SN nodes are service nodes which provide management service, like workload manager and meta server. CN nodes are compute nodes. In Trinity final phase, there are 2 types of compute nodes. Intel Xeon (Haswell) nodes and Intel Xeon Phi (Knights Landing- KNL) nodes. In our simulation, we do not differentiate Haswell nodes and KNL nodes because currently we only focus on the burst buffer and network performance. DW nodes are Datawarp nodes. Datawarp nodes are Cray's implementation of burst buffers in Cray XC40. 2 Intel DC P3608 SSDs are installed to each Datawarp node. For different nodes, the numbers of nodes attached to an Aries router are different. The ratios between nodes and an Aries router are: 4: 1 for compute nodes, 2: 1 for Datawarp nodes and 2:1 for LNET nodes.

Our simulation models 3 entities in each group: compute node, router and burst buffer node. The reason of ignoring LNET node and PFS is that currently we are studying the I/O inside the dragonfly network that only involves with burst buffer nodes. Since services nodes only provide some services, their I/O traffic is neglectable . According to the configuration of Trinity, there are 96 routers and 384 nodes in each group. Based on the ratios between routers and different nodes, we simulate 360 compute nodes, 10 Datawarp nodes and 4 LNET nodes. We leave 10 nodes unused because of the unclear number of service nodes. For Trinity final phase, we simulate 57 groups which sum up to 20,520 compute nodes and 570 Datawarp nodes.

B. Trinity network configurations

1) Network bandwidths: In this section, bandwidth numbers of each type of links are given and explained. For the inter-group link, the bidirectional bandwidth is 18.75GB/s. As shown in Figure.2, there are 10 tiles used for global link. If all 10 tiles are used for inter-group links with 1-to-1 ratio, it means a router can be connected to up to 960 other groups. The number is far more larger than current need, therefore 2 routers are combined together to share 5 optical connectors so that each inter-group link bandwidth is crease by 4 times of 4.7GB/s. For intra-chassis links, since only 1 tile is used for each intra-chassis link, the bandwidth is 5.25GB/s. For inter-chassis links, 3 tiles are combined to be used by 1 inter-chassis link, thus inter-chassis link bandwidth is 3 times of 5.25GB/s,

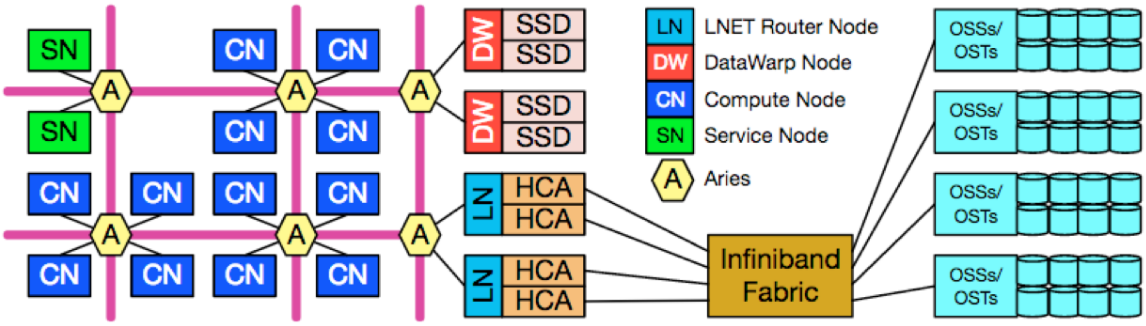

Fig. 4: Trinity group overview

which is 15.75GB/s. For intra-blade links, the bandwidth is fixed by the bandwidth of PCIe 3.0 interface which is 8GB/s.

2) Inter-group link arrangement: Generally, there are two issues regarding the arrangement. The first is which two groups should be connected. The second is which two routers in these two groups should be connected. To solve these two issues, we found a paper [11] focusing on global link arrangement in Dragonfly network. There are mainly 3 arrangement patterns, shown in Figure 6: Absolute, Relative and Circulant-based. The detailed mapping algorithms can be found in the paper mentioned above.

Currently, we are using the Absolute pattern. CODES is using the same pattern but it is not easy to understand and somehow buggy with my code, so we re-implemented it. The question is what kind of group cabling pattern is Trinity is using in reality? Because in the paper, the authors benchmarked these three patterns and conclude that there are differences under different application situation, and group link arrangement is a critical part of the dragonfly network.

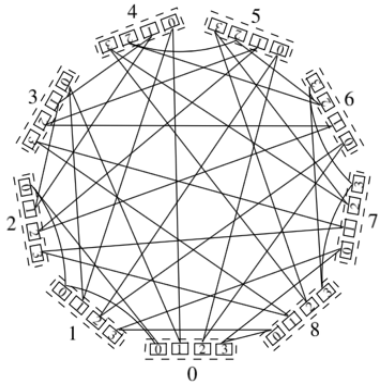

Fig. 5: Direct inter-group link arrangement pattern

3) CODES application layer design: The basic working mechanism of our simulator is shown in Figure. 6. There are mainly 2 layers: Application Layer and Model-net Layer. The application layer processes (ALP) provide different user defined functions such as generating workload on compute nodes, simulating IO operations on burst buffer nodes. The model-net layer processes (MLP) compose the dragonfly network infrastructure which is fixed and cannot be modified by Application Layer behaviors. Network packets are simulated with messages between processes. An ALP generates messages and sends it to a corresponding MLP, and then the messages are converted to model-net network packets. After being routed through the network, packets are delivered to the destination MLP and sent to the corresponding destination ALP. The time between ALP sends a message and receives an ACK from the message receiver is recorded to calculate the throughput.

We also implemented an IOR benchmark workload generator. IOR benchmark is designed to test system IO performance. By setting up parameters like N-to-N flag, xfer size and Read/Write flag to, a user can collect IO performance data under different scenarios. The benchmark workload generator is also implemented in the application layer. By accepting parameters, the synthetic workload generator instructs ALPs representing computing nodes to send events to Datawarp node ALPs. Additionally, we modified the default Darshan IO workload generator in CODES. The current Darshan IO workload generator of CODES only supports Darshan 2.6 version traces. After some modifications, our Darshan IO workload generator now support most recent Darshan 3.x traces.

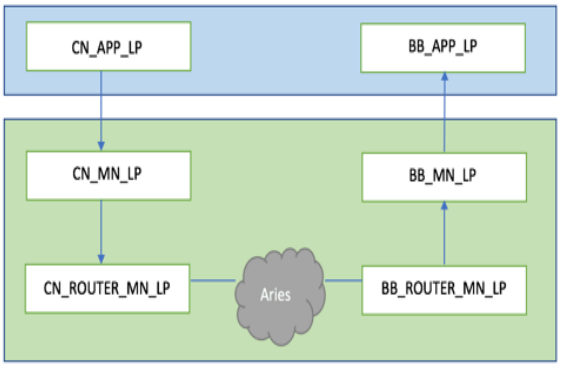

Fig. 6: CODES application layer

IV. Experiments and Results

In this section, first we give details of our testbed. Then we show the validation results against IOR benchmark[8] data, validation against Darshan IO traces.

A. Experiment environment

All experiments are conducted on 8 Chameleon baremetal compute node instances. Chameleon[10] is a cluster setup in both TACC and University of Chicago. The processor model is Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz.

B. IOR validation

The IOR data validation includes two parts. One is Trinity Phase II results. The other is latest data collected on Trinity form the merge. For all following experiments, the xfer size is 1MB and sequential IO.

1) Trinity Phase II: We validated our simulator against the IOR test result of Trinity Phase II. Figure.7 shows the N-N write test results of IOR and our simulation. Since there are two sets of IOR result collected on different dates, we put both of them into the figure. The result can be divided into 3 stages:

- 1-256 nodes: In this stage, each compute node can keep a 2.0 GB write bandwidth. The bandwidth is limited by compute node injection rat.
- 256-512 nodes: When compute nodes are all in one group and write to burst buffers, the bottleneck will be the local links, both intra-chassis and inter-chassis links. Because all nodes are sending traffic towards the routers that have global links. The congestion on the local side of these routers limits full utilization of global links.
- 512-8192: In this stage, IO bandwidth is capped by burst buffers maximum IO bandwidth. Totally, 230 burst buffers can reach 1200GB/s bandwidth.

N-1 simulation result is not matching either set of IOR result. But from another point of view, the figure can tell that the Datawarp software is still not consistent in Trinity Phase II and Cray is still updating it. In the N-N read test, we had a problem that when the experiment scales up to 1K nodes the simulation crashes. So we cannot get further results beyond that point.

2) Trinity from the merge: On 17th July 2017, Trinity finished the merge, which combines two halves of the system together. One half contains 9,384 compute nodes with Intel Haswell processors and the other has 9,984 Intel Knights Landing (KNL) processors. After the merge, Trinity now has >1,900 compute nodes, 576 burst buffer (Datawarp) nodes. We also got some IOR data from the merge and we validated it with the simulator. N-N write result is given in Figure 10: Some clarification needs to be given. Now Trinity has more than 19,000 compute nodes and 576 burst buffers. Our simulation is modeling about 20,000 compute nodes and 570 burst buffer nodes. The reason why the test is still using 8K as the maximum scale is that the test was conducted on the 8K Hasewell node half and on the other 8K Knights Landing node half in each time. We used the higher one on different node scales as the contrast bandwidth. As shown in the Fig 14, there is a gap in 4K and 8K experiments, but before that, the accuracy is fine. Accuracy of N-1 write simulation is similar to N-N write. But another gap appears when the simulation scales from 256 to 512 nodes which is also from one group to two groups. Though we solve the crash problem that exists

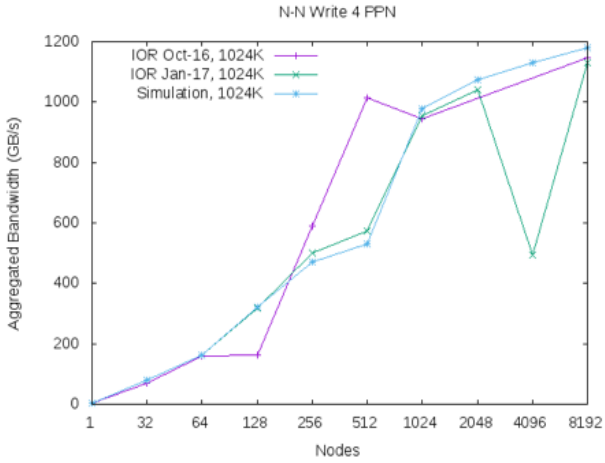

Fig. 7: Trinity Phase II N-N Write

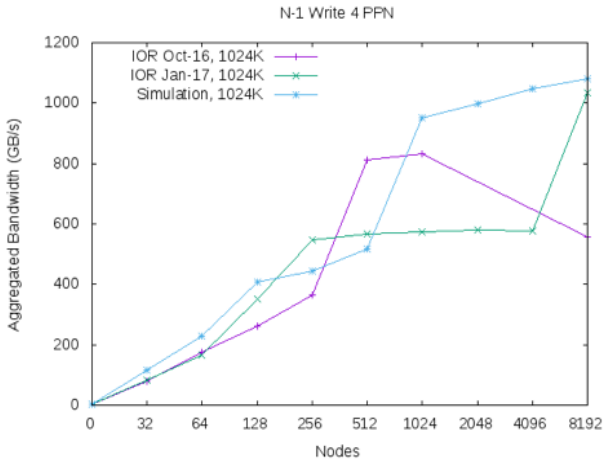

Fig. 8: Trinity Phase II N-1 Write

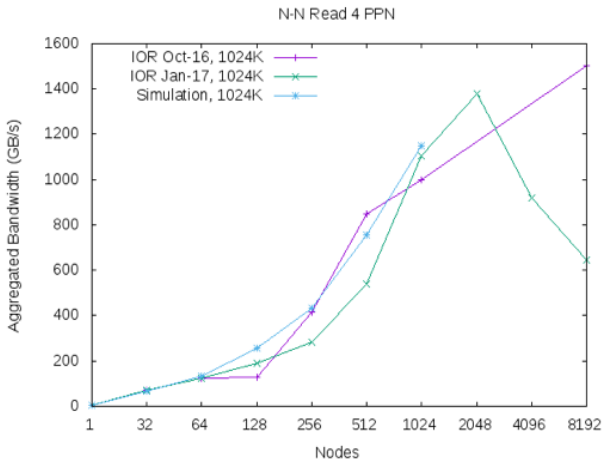

Fig. 9: Trinity Phase II N-N Read

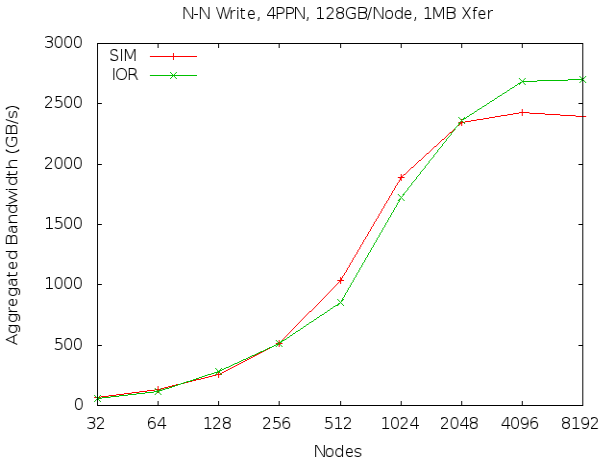

Fig. 10: Trinity merge N-N Write

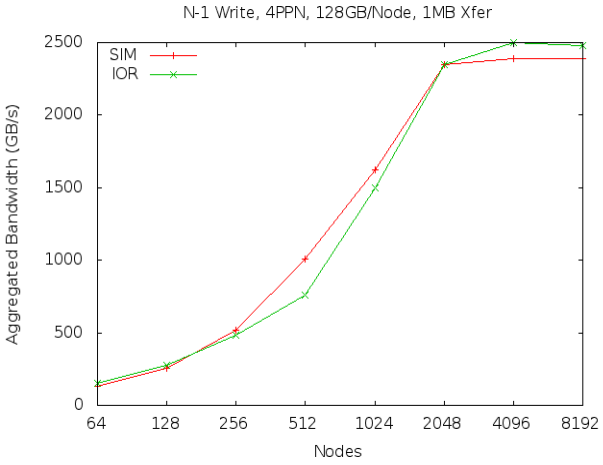

Fig. 11: Trinity merge N-1 Write

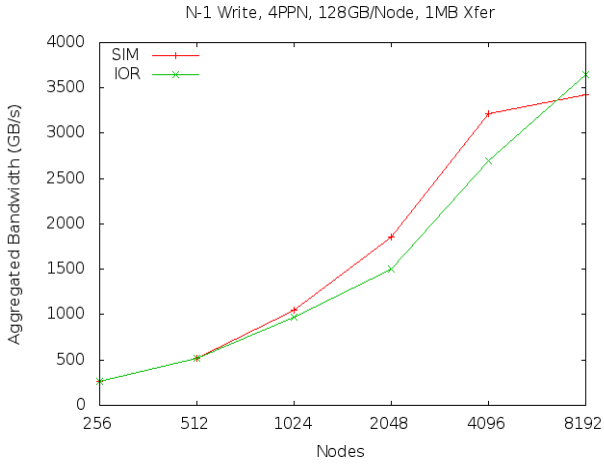

Fig. 12: Trinity merge N-N Read

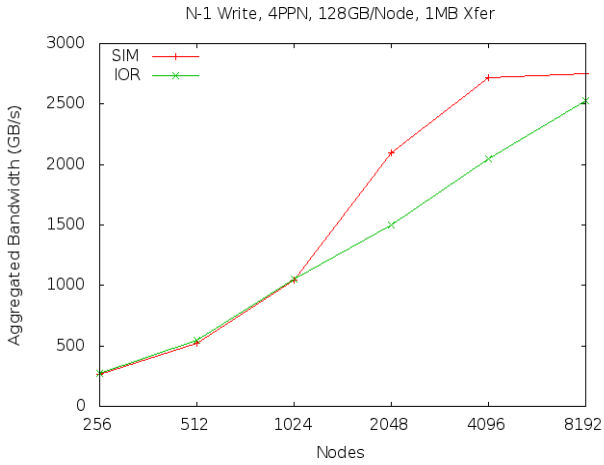

Fig. 13: Trinity merge N-1 Read

in Trinity Phase II simulation, read still remains a problem. There are two issues in N-N read in Figure. 12: First, the error rate is bigger than 15% in 2K and 4K test. Second, in real IOR benchmark, the bandwidth is still increase at 8K scale while our simulation reaches a cap. The issue still remains to be confirmed. In Figure.13, the N-1 read result. The accuracy problem is still obvious at 2K and 4K scale.

3) Darshan IO trace validation: We are using Darshan 3.10 I/O logs collected on Cori[9] at NERSC to validate our simulator. Figure. 15 is a sample validation result. We use Darshan I/O trace logs with different rank counts (x-axis) to generate workloads, and compare the average throughput (y-axis MB/s) calculated from simulation with that from original log files. The result shows a 50% error rate. Though the error rate is too high to validate the simulator, it does show some similar trend. Afterwards, we realized that the traces we used were all Lustre IO traces rather than burst buffer IO traces. Since burst buffers provide higher bandwidth than Lustre does, it accounts for the high error rates we got. In fact, only 2 out of more than 30,000 traces are burst buffer IO traces. One has 60 ranks and the other has 160 ranks either of which is

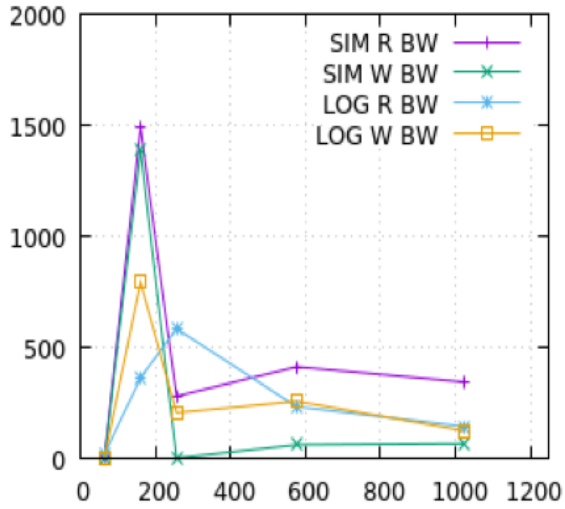

Fig. 14: Average R/W throughput from simulation and logs

enough for validation. Besides, they were collected on Cori whose configuration is quite different from Trinity. So we just cannot use them for application validation. So far we have not got any application's Darshan traces from Trinity, so we have to suspend the application validation at this time.

V. Future work

There are three parts of our next step work:

- More simulation tuning, especially in the Read part.
- After validating the fidelity, we will run experiments with different configurations to see the bottleneck so that we can have enough data to put in the IPDPS paper.
- Then we can use applications Darshan IO traces to further study topics like job placement, network isolation and different modes of burst buffers.

References

[1] Bob Alverson, Edwin Froese, et al, Cray Inc. Cray XC Series Network

[2] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, Modeling a million-node dragonfly network using massively parallel discrete-event simulation, in High Performance Comput., Networking, Storage and Anal. (SCC) SC Companion, 2012, pp. 366376.

[3] Carothers, Christopher D., David Bauer, and Shawn Pearce. "ROSS: A high-performance, low-memory, modular Time Warp system." Journal of Parallel and Distributed Computing 62.11 (2002): 1648-1669.

[4] Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. 24/7 characterization of petascale I/O workloads. In Proceedings of 2009 Workshop on Interfaces and Architectures for Scientific Data Storage, September 2009

[5] M Vigil ."Trinity Advanced Technology System Overview".

[6] "Top500", https://www.top500.org/

[7] Cray DataWarp Applications I/O Accelerator, Cray Inc, http://www.cray.com/datawarp

[8] Shan, Hongzhang, and John Shalf. "Using IOR to analyze the I/O performance for HPC platforms." Lawrence Berkeley National Laboratory (2007).

[9] NERSC, Cori, https://www.nersc.gov/users/computational-systems/cori/.

[10] A configurable experimental environment for large-scale cloud research, https://www.chameleoncloud.org/

[11] Hastings, Emily, et al. "Comparing global link arrangements for Dragonfly networks." Cluster Computing (CLUSTER), 2015 IEEE International Conference on. IEEE, 2015.