Vol.2014-HPC-145 No.5
2014/7/28

Towards Cloud Bursting for Extreme Scale Supercomputers (Unrefereed Workshop Manuscript)

Tianqi Xu[1] Kento Sato[1] Satoshi Matsuoka[1]

Abstract:
Extreme-scale HPC systems, which consist of a large number of compute nodes, can provide high computational capacity for multiple users. However, computing nodes in the systems occasionally can not meet the demand due to bursty job requests in short period times. In order to accommodate the bursty requests, we consider federating HPC systems with public clouds, which is known as cloud bursting. Although the federated systems can acquire virtually infinite computational power with cloud bursting, the QoS may not be guaranteed due to a significant performance gap between HPC systems and public clouds. The most critical problem is a gap in I/O performance. In this paper, we propose an I/O acceleration technique using distributed cloud bursting buffers. We also create the I/O performance model to explore the effectiveness. Our model-based simulations, which target the TSUBAME supercomputer for an HPC system, and AMAZON EC2 for a public cloud, show that the distributed cloud busting buffer can improve I/O throughput while reducing the cost.

Keywords: Supercomputer, Cloud, Burst buffer

1. Introduction

Supercomputers provide an increasing number of scientific applications with high computational power by a large number of processors, large bandwidth memory and interconnects. Although supercomputers can also offer high computational capacity, the computational resources are not unlimited. Supercomputers can not meet the demands of users when the demands exceed the limit. For example, on grand challenge projects, computational resources are exclusively used for the scientific discoveries [1], [2], and the system can not provide adequate resources for other jobs. In addition, under power budget constrain [3], [4], a part of compute nodes are required to be shutdown to reduce its power consumption, which also leads inadequate of resources for all users.

One of solutions to provide adequate computational resources under the constrains, is federating the supercomputer with public clouds. By moving a part of jobs to public clouds when there are not enough compute nodes available, we can meet the needs for users' requests even under the constrains, which is known as *cloud bursting* [5], [6]. Although cloud bursting is technically feasible [7], [8], [9], and is employed by several private companies, there are several problems if we apply this technique to supercomputers. One of the problems is a significant performance gap between supercomputers and public clouds especially in I/O performance For example, if we migrate a part of jobs from a supercomputer to a public cloud or run jobs on a public cloud insted of a supercomputer, and the jobs need data located in a parallel file system (PFS), the data need to be transferred between the two system. Because the two systems are usually geographically distributed, and network throughput between the systems is quite low, the low I/O throughput suffers supercomputer users. Thus, improving I/O throughput between two systems, which are geographically distributed each other, is critical in federating supercomputers with public clouds.

In this paper, we propose I/O burst buffer architecture. The I/O burst buffers consists of I/O dedicated staging nodes, which cache hot files in the buffers, and enables asynchronous write back for improving both read and write operations to remote file systems. We also model the I/O burst buffers to optimize configurations of the I/O burst buffers according to dynamically changing environments According to our simulation, we can achieve a 4-20 times higher throughput (depends on data locality and other factors) by using 20 I/O buffer nodes on Amazon EC2 environment, as well as reduce 2-12 times of cost.

Our contributions can be summarized as following:
- A I/O burst buffer architecture for increasing data transfer throughput between two systems;
- A throughput-based I/O burst buffer model, which estimate I/O throughput of systems with I/O burst buffers, and a cost-based I/O burst buffer model, which estimate overall cost give a system configuration;
- Evaluation of the I/O burst buffer architecture based the performance models by using real data obtained from several benchmarks from the TSUBAME supercomputer and Amazon EC2.

[1] Tokyo Institute of Technology, Japan, Tokyo, Meguro, okayama 2-12-1

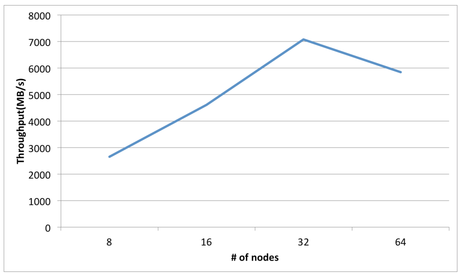

Fig. 1 I/O Throughput to Lustre inside TSUBAME direct mount

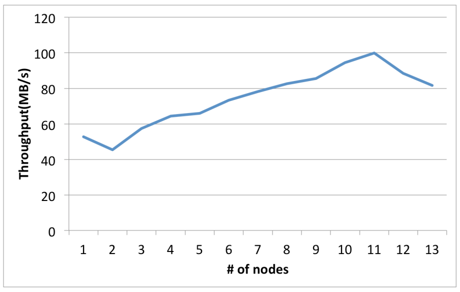

Fig. 2 I/O Throughput from AMAZON EC2 to file system inside our lab using sshfs

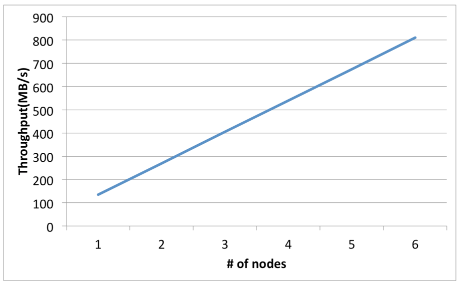

Fig. 3 point to point connection inside AMAZON

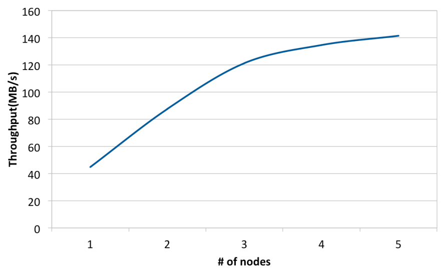

Fig. 4 point to point connection throughput between AMAZON and our lab

The rest of this paper is organized as follows. In Section 2, we clarify the motivation and the background. We introduce an overview of the I/O bursting buffer architecture in Section 3, and show the performance models to evaluate the I/O bursting buffer architecture in Section 4. In Section 5, we present our experimental results based of our performance models. Finally, we detail related work in Section 6, and conclusion in Section 7.

2. Background and Motivation

2.1 Public Cloud

Public cloud is one in which the services and infrastructure such as applications and storage are provided off-site over the Internet, the main benefit of using a public cloud service are: easy fast and inexpensive set-up because hardware, application and bandwidth are covered by the provider, easy to scale to meet needs, no wasted resources because you pay for what you use. There are many companies provide public cloud solution like google, IBM, Microsoft etc., among them, Amazon Elastic Compute Cloud (Amazon EC2)[10] is one of the most famous public cloud, like other public cloud, Amazon EC2 provides Virtualize computing, and owns computing and data centers in several geographic regions. Amazon EC2 provide a large amount of instance types optimized to fit different use cases comprise varying combinations of CPU, memory, storage, and networking capacity. In this study, we used general purpose m3.xlarge instance which has 4 vCPUs, 15GiB memory, 2*80GB SSD storage and high level interconnection network network condition in Tokyo region, we run Amazon Linux AMI 2014.03.2 (HVM), which is based on Linux 3.10 on these instances. Amazon EC2 charges for nodes usage, Amazon use a pay-as-you-go pricing policy[10], which means you pay only for what you use, there is no minimum fee and you will pay for compute capacity by the hour with no long-term commitments.

Table 1 TSUBAME thin node specification

CPU	Intel Xeon 2.93GHz CPU (4 cores)*2
Memory	54GB (16 GB)
SSD	120GB
GPU	NVIDIA Tesla K20X *3
Network	QDR InfiniBand *2 (80Gbps)

2.2 Federation between supercomputers and clouds

When we try to federate supercomputer with public clouds, there are need to have a similar performance and environment between these two systems. So we virtualize the supercomputer(like TSUBAME U queue) and federate VMs running on supercomputers with VMs provided by public cloud. By using VMs, we can run the same image on both supercomputer and public cloud as well as set both the same specification, and make it looks the same to user.

2.3 Challenges in Federation with clouds

Although we virtualize supercomputer and obtain a similar environment on both supercomputers and clouds, there are still several problems remains, for example, input data of applications running on supercomputer is usually stored in shared storages in the same system and can be read from and wrote to these storages with a extremely high throughput.

We use Iperf[11], which was developed by NLANR/DAST as a modern alternative for measuring TCP and UDP bandwidth performance, and IOR[12], which is widely used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces. Fig. 1 shows read and write throughput from TSUBAME V queue nodes (VM running on TSUBAME Thin node, which specification is shown in **Table 1**) and TSBUAME Lustre file system, which is mounted by using lustre client, it is a interconnection throughput inside TSUBAME supercomputer, we can see that I/O throughput growing as numbers of nodes growing, and the aggregate read and write throughput reach 6-8GB/s with 64 nodes, the same result also can be seen in [13]. However, Fig. 2 shows I/O throughput between AMAZON EC2 nodes and a file system machine inside our lab, since TSUBAME Lustre can not be accessed outside of TSUBAME because of security problem, instead of TSUBAME Lustre file system, we used a file server inside our lab, which has 1GBit/s internet bandwidth, also because of security problem, we used sshfs[14],which is a filesystem client based on SSH File Transfer Protocol, to mount this file system from Amazon. We can see that the I/O throughput also grows as number of nodes grows but the aggregate throughput is only 100-140 MB/s, which is limited by Internet bandwidth, about 40-80 times smaller than throughput inside TSUBAME. For data-sensitive application, low I/O throughput is devastating, especially for application running on supercomputers, furthermore, lower I/O throughput leads a longer execution time, according to Amazon pay-as-you-go policy, longer execution time means more cost.

However, when we look at interconnection network throughput inside Amazon as Fig.3 shows, although we only show the result up to 8 pairs of nodes, each node achieved only 135MB/s (1GBit/s), the influence between nodes is extremely small, figure shows a perfect linear line also a strong scalability. since when we running the benchmark, many other users were also running applications on Amazon, so we can assume that highest throughput 1GB/s (8Gbit/s) shown in Fig. 3 is not the maximum bandwidth of interconnection network in Amazon EC2 . Comparing Fig. 3 with Fig. 2 and Fig. 4, interconnection network throughput is much higher than Internet, it shows that our solution, by using I/O buffer nodes can achieve a higher interconnection network throughput than Internet.

To solve this problem, we propose our I/O burst buffer architecture, since usually Internet throughput is the bottleneck, by using parallel I/O and cacheing file in I/O buffer nodes, we can fully utilize the Internet bandwidth as well as avoid frequently transferring file between two systems, hence achieve a high I/O throughput. Although using I/O buffer can increase I/O throughput, reduce the execution time and cost, I/O buffer nodes will be charged for usage, also a better instance will cost more than a normal one, there will be a trade-off between I/O performance and monetary cost, we introduce a throughput-based model and a cost-based model to predict the throughput and overall cost.

3. I/O Burst Buffer Overview

An overview of I/O burst buffer architecture and two kinds of buffer model: one-side buffer and two-side buffer are described in this section. As we mentioned in the previous section, our model takes advantage of high throughput inside a system, and use buffer queue system in order to increase throughput between two systems. Two kinds of buffer are used in our I/O burst buffer architecture, the first one is in client computing node, first buffer user I/O in the same node, another one is in I/O buffer nodes. The main idea is that some of computing nodes serve as a I/O buffer nodes in each system, for write I/O data, if buffer queue in I/O buffer nodes is not full, data can first be buffered in buffer queue in the same system, and then be transferred to final storage system, and for reading, if that file is stored in the buffer queue, computing nodes can read from I/O nodes through interconnection network. In other cases (buffer queue is full when issue a write request or requested file is not stored in buffer queue when issue a read request we call it cache miss), an read from or write back operation described below will be executed.

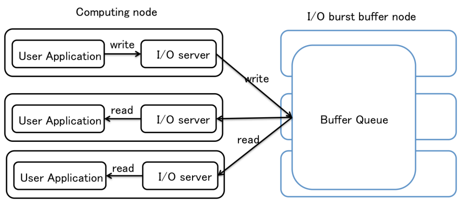

Fig. 5 I/O server and buffer queue

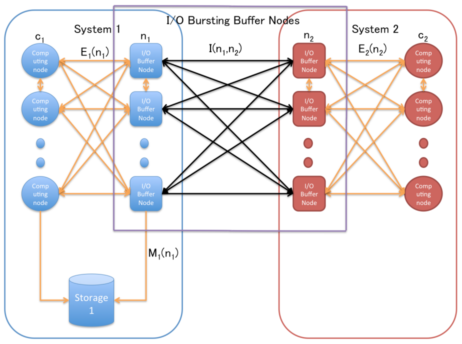

Fig. 6 overall illustrate of I/O Burst Buffer Architecture

3.1 Assumptions

First, we make several assumptions for the federated environments as follows.

- All computing nodes are connected by large bandwidth and interconnection network, note network topology maybe different in each system, so topology is not specified here, interconnection network performance is measured by throughput.
- There is a shared storage for date sharing inside system, all computing nodes are connected with shared storage, also the filesystem of shared storage is not specified and performance is measured by throughput.
- All nodes used by the same job are allocated in the same system, since the Internet bandwidth is low and unstable, user do not want to make communication between nodes via Internet.

3.2 I/O Burst Buffer Architecture

There are two kinds of nodes in our I/O burst buffer architecture: *computing nodes* and *I/O burst nodes*, computing nodes run user's application and I/O burst nodes. Among I/O burst nodes, there is a master nodes, which maintain a global buffer queue and a namespace, control buffer read and write, and manage I/O buffer nodes, buffer data is distributed to all I/O buffer nodes in order to enable concurrent read and write.

Fig. 5 is a illustrate of I/O server inside client nodes and buffer queue in I/O burst nodes. In each computing node, there is a I/O server, which is a filesystem client used to buffer I/O data, communicate with I/O buffer nodes, including send I/O request and send or receive I/O data.

When a user application issue a write request, I/O data will be buffered in that node by I/O server, when user close the file, call flush function or I/O data size exceed I/O server buffer size, I/O

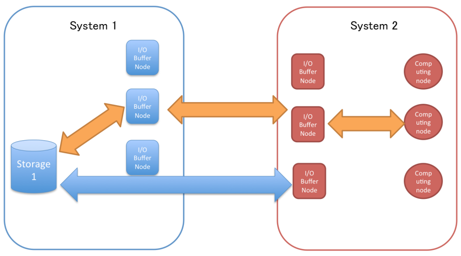

Fig. 7 read and write operation

server will try to transfer I/O data to buffer queue in I/O burst buffer, if buffer queue in not full, I/O data will be sent to I/O burst buffer via interconnection network and can be seen among computing nodes in the same system after that. However if buffer queue is full, I/O server will block user application and wait until buffer queue is ready to receieve new I/O data, causing a low I/O throughput.

Similarly when user issue a read request, there are two conditions: required file is buffered in buffer queue in I/O burst buffer, or file is stored only in storage in another system. In the first cases, file can be transferred to computing nodes from buffer queue directly, and can achieve a high throughput. If data is not in buffer queue, then a read operation described below will be executed, since data must be transferred from storage in another system, in this case, throughput will depend on Internet condition and it is hard to achieve a high throughput.

Fig. 6 shows a overall illustrate of I/O burst buffer architecture.

3.3 Read and Write

In this section we describe read and write operations when cache miss happens. We propose two kinds of model: one-side buffer and two-side buffer.

3.3.1 One-side Buffer

As Fig. 7 blue arrow shows, one-side buffer is a connection between I/O buffer nodes in system 2 and storage in system 1 directly. When I/O nodes need to write data back to storage in another system, since data is already spread among I/O nodes, a parallel write can be used to fully utilize the Internet bandwidth.

Similarly, when I/O nodes need to read data from storage in another system, several I/O nodes will be assigned equal size of data, and then these nodes read assigned piece of data from the storage concurrently.

There will be two problems in this solution:

- First storage should be opened to Internet, in order that I/O nodes can read from or write to it. However storage system in supercomputer store Petabyte of research data, open the access of storage system means that put these research data under risk of attack.
- As we mentioned before, throughput of one side communication is lower than two side communication, since we use SSH protocol based File system for security reason. also two side communication can achieve a higher throughput with fewer nodes, according to pay-as-you-go policy, reduce number of nodes can reduce cost.

3.3.2 Two-side Buffer

On the contrary, two-side buffer solution uses I/O buffer nodes in both two systems as Fig. 7 orange arrows show. I/O data will be split twice, transferred throughput two sides of I/O buffer nodes and then reached the destination.

Consider I/O nodes in system 2 issue a write back operation, also, data is already spread among I/O nodes in system 2, each I/O node will find a pair among I/O nodes in system 1, then transfer their piece of data to system 1 concurrently. After I/O node in system 1 received data, they write data back to storage in system 1.

Compared with one-side buffer, two-side buffer require both systems allocate I/O buffer nodes, if node usage is charge in both system, total cost may be larger than one-side buffer.

However since two-side buffer does not read data from or write data to storage directly, it means we can compress data before transfer it, though it will take some time to compress and decompress data, when the Internet throughput is extremely low, compress and decompress time will be far smaller than transferring time, and compress data can make more data buffered in buffer queue. Also, unlike one-side buffer, two-side buffer do not require storage to be opened to Internet, only required several I/O nodes have access to Internet.

For these reasons, in the following section, we use two-side buffer model as our read from and write back model in I/O burst buffer.

4. I/O Burst Buffer Model

Since the instance type choice, interconnection network condition, Internet condition will affect the actual I/O throughput between two system, we introduce a throughput-based model, cost-based model and buffer queue write back model used to predict throughput, overall cost and in buffer rate when use I/O burst buffer and direct I/O, and using these predict model to make decision like which type of instance should use, and how many I/O nodes is needed in a particular cases.

The throughput-based model compares throughput with and without I/O burst buffer, cost-based model compare cost with and without I/O burst buffer, and buffer queue write back model shows the situation that I/O buffer queue will be full. We make definitions shows in Table 2 in order to describe our model.

4.1 Throughput-based Model

First we consider direct I/O, computing nodes read from and write back to storage in another system directly. In the case of direct I/O, computing nodes connect to storage in another system directly, there is only one data transfer, so both read and write throughput will be:

$$\mathrm{thr}_{\mathrm{direct}} = D_1(c_2) \qquad (1)$$

When using I/O burst buffer, there will be three data transfers: computation nodes to I/O buffer nodes in system 2, I/O buffer nodes in system 2 to I/O buffer nodes in system 1, I/O buffer nodes in system 1 to storage in system 1, for reading, if required file is buffered in buffer queue, computing node can read data from buffer queue directly, then throughput will be:$E_2(n_2)$, similarly, if buffer queue still have enough space for I/O data when issue a write request, computing nodes can write data to buffer

Table 2 Definition of parameters

c_1, c_2	Numbers of computing nodes in system 1 and system 2
n_1, n_2	Numbers of I/O buffer nodes in system 1 and system 2.
m_1, m_2	Available memory size for each I/O node in system 1 and 2, also the maximum buffer size will be $n_1 \times m_1$ and $n_2 \times m_2$
$D_1(n_2), D_2(n_2)$	Throughput when $n_2(n_1)$ I/O nodes in system 1 (system 2) connect to storage in the other system directly, here we assume I/O nodes and computing nodes in each system have the same Internet connection speed.
$I(n_1, n_2)$	Internet throughput using n_1, n_2 nodes respectively, since overall Internet throughput is affected by number of nodes involved in connection.
$E_1(n_1), E_2(n_2)$	Interconnection network throughput in system 1 and 2, although interconnection throughput is also affected by numbers of I/O nodes and computing nodes, numbers of users will running application on different number of computing nodes.
$M_1(n_1), M_2(n_2)$	Throughput of connection between storage and n_1 I/O nodes in system 1 and storage and n_2 I/O nodes in system 2.
$C_iM(T)$	Cost for standard node in system i for T time usage
$C_iHM(T)$	I/O nodes in system i for T time usage, since I/O nodes may use a better network condition, we assume it will cost more than normal nodes.

queue, and throughput also will be:$E_2(n_2)$.

If required file is not buffered in buffer queue when issue a read request, or when issue a write request buffer queue is full, the throughput will be:min$\{M_1(n_1), I(n_1, n_2), E_2(n_2)$, since for read request, file need to be read from storage in another system Internet, and for write request, it need to wait until buffer buffer write buffered file back the storage. throughput

$$\text{thr}_{\text{I/O buffer}} = \begin{cases} E_2(n_2) \\ \text{buffer queue available} \\ \min\{M_1(n_1), I(n_1, n_2), E_2(n_2)\} \\ \text{buffer queue full or file miss} \end{cases} \quad (2)$$

Although we have two cases for each read and write request (in buffer or not in buffer), it is very difficult to predict the in buffer rate, since it will be affected by I/O size, Internet throughput, buffer queue size and so on.

4.2 Cost-based Model

Although we may use I/O burst buffer to achieve a high I/O throughput and leading a decrease of execution time, and according to public cloud pay for usage policy, the cost for computing nodes may reduce, the relation between overall cost and I/O throughput can not be such simple, since I/O nodes will still charged for usage, and if we use a instance with high interconnection network condition in order to get a higher throughput, we may be charged for more money.

Here we provide a cost-based model in order to predict overall cost for using I/O burst buffer and direct I/O. Like throughput-based model, cost will be affected by in buffer rate, but it depends on application and buffer queue size, and is very difficult to predict. According to 1,and 2 total time for transferring unit size of

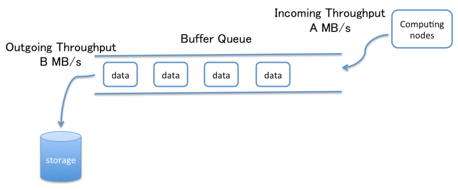

Fig. 8 buffer queue

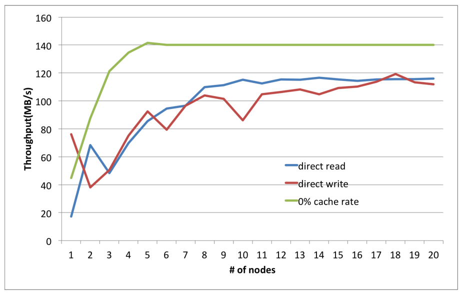

Fig. 9 throughput comparison with and without I/O burst buffer

data can be compute as:

$$\begin{cases} T_1 = \frac{Data\ size}{D_1(c_2)} & \text{direct} \\ T_2 = \frac{Data\ size}{\min\{M_1(n_1), I(n_1,n_2), E_2(n_2)\}} & \text{buffer queue full or file miss} \\ T_3 = \frac{Data\ size}{E_2(n_2)} & \text{buffer queue available} \end{cases} \quad (3)$$

here we compute cost by using T_1, T_2, T_3:

$$\text{cost}_{\text{direct}} = c_2 \times C_2M(T_1) + n_2 \times C_2HM(T_1) \quad (4)$$

$$\begin{cases} c_2 \times C_2M(T_2) + n_1 \times C_1HM(T_2) + n_2 \times C_2HM(T_2) \\ \text{buffer queue available} \\ c_2 \times C_2M(T_3) + n_1 \times C_1HM(T_3) + n_2 \times C_2HM(T_3) \\ \text{buffer queue full} \end{cases} \quad (5)$$

4.3 Buffer Queue Write Back Model

If the buffer size is unlimited. then we can buffer all data in the I/O buffer nodes, and achieve a high throughput in cloud burstHowever buffer size can not be unlimited, we can not buffer all data in the I/O buffer nodes, data in buffer nodes have to be write back to storage in another system. The problem is which data should be written back to storage, like cache in cpu, if we can reduce cache miss in this situation, we can increase throughput. According to data locality, we use a priority queue to determine which data should be write back.

Consider Fig. 8, assume average incoming throughput is A MB/s and average outgoing throughput is B MB/s, if A always larger than B, after T time buffer queue will full.

$$T = \frac{m_2 \times n_2}{A - B} \quad (6)$$

After that, since buffer queue is full, I/O server can not send more data to I/O buffer nodes, have to block any read and write request since that.

5. Evaluation

To evaluate how much federated systems with I/O burst buffers

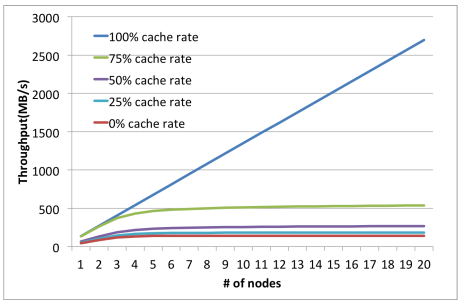

Fig. 10 throughput comparison between each cache rate

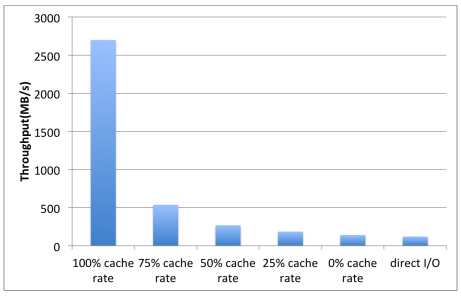

Fig. 11 maximum avaliable throughput

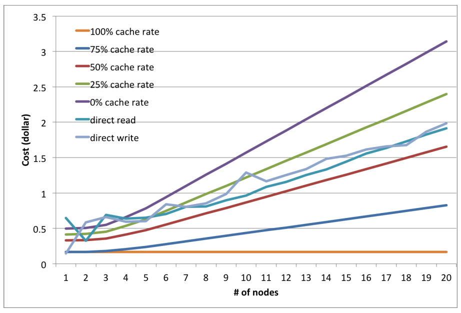

Fig. 12 cost comparison

can improve I/O performance and the cost, we conduct several simulations based on preliminary performance data taken from several benchmarks on both our in-house cluster and Amazon EC2 in a Tokyo region in Section 2.In the simulations, we assume that computing nodes running on one system read or write files on storage of the other system with the same I/O throughput for both read and write operations. We also assume an instance type of both the computing nodes and I/O burst buffer nodes is m3.xlarge provided by Amazon EC2. We compare I/O throughput and overall costs of a system using direct I/O with a system using I/O burst buffer by scaling out up to 20 computing nodes under different cahch hit rate. For read operations, the cache rate means percentage of data, which is read from buffer queues, to total read size. For write operations, the cache rate means percentage of data, which is written to buffer queues before queue becomes full, to total write size. If read and write are completed via caches on I/O burst buffers without accessing to remote storage, read and write throughput become identical to throughput between a compute node and I/O burst buffer nodes.

First we compare I/O throughput of direct connection with the throughput of I/O burst buffer under different cache rates assuming that TSUBAME is federated with Amazon EC2. In this simulation, the number of I/O buffer nodes is equal to the number of computing nodes. This configuration can achieve a maximum I/O throughput. In practice, an I/O buffer node may have higher network throghput than computing nodes, and the throughput can not fully utilized. But we assume there are many users running their applications on large amount of computing nodes, which concurrently read data from or write data to one I/O bust buffer nodes, the network throughput to I/O buffer is fully utilized.

Fig. 9 and Fig. 10 shows the throughput comparisons between systems with and without I/O burst buffers. For direct I/O, each computing nodes read from and write data to storage directly and concurrently. When the number of nodes is small, throughput increases as the number of nodes increses. However, after throughput reach the peak bandwidth between two systems, (120MB/s when number of nodes equals to 10 in our case), the throughput becomes flat.

In 0% cache rate cases, since all data should be read from or write back to storage, the throughput show a similar trend.

The large difference is shown in cache 100%(Fig. 10). In the 100% cases, all reading files are buffered in buffer queue, also buffer queue is not full for writes. Thus, computing nodes can read from and write to buffer queue each time without accessing to remote storage, and the throughput becomens identical to network throughput between computing nodes and I/O buffer nodes. As we mentioned, network throughput between two instances within Amazon EC2 shows the scalablity even if we increase the pairs of two instances. I/O throughput can finally achieve around 2700MB/s, which is about 20 times faster than one of direct I/O.

However, the 100% cache rate is an ideal condition, and is not practical in real applications, so we also estimate the throughput under 75%,50% and 25% cache rates in Fig. 10. If the cache rate is low, most of reading and writing data need to be transferred via a network between the two systems, i.e, TSUBAME and Amazon EC2. If applications can use the cache for read or write, I/O throughput can be improved depending on the cache rates. To estimate the throughput under different cache rates, we use the below model.

$$throughput(n) = \frac{1}{MAX\{\frac{cache\_rate}{E_2(n)}, \frac{1-cache\_rate}{I(n,n)} + \frac{1-cache\_rate}{E_2(n)}\}}$$

we denote $I(n,n)$ as two-side network throughput shown in Fig. 4, $E_2(n)$ network throughput within Amazon EC2 shown in Fig. 3.

As shown in Fig. 10, we can also see that I/O throughput in 75% cache rate are much lower than ones of 100% cache rate. This is because of a larege gap between network throughput within Amazon EC2 and between the two systems, applications must wait until reading or writing data, which is not in caches, are transferred from or to remote storage.

In this simulation we only consider one job use up to 20 computing nodes, but in practice, a large number of computing nodes issue read and write requests concurrently, Fig. 11 shows the maximum throughput of different cache rates and direct I/O. We can see that write can achieve an extremely high throughput using caches on I/O burst buffer nodes while read can not achieve that high throughput unless target files are in caches.

Then we compare overall cost between systems with and without I/O burst buffers. As we mentioned, the cost is determined by both the number of nodes and the execution time. As we showed, by using I/O burst buffers, we can achieve a high I/O throughput, and reduce both the execution time and cost. However, I/O buffer nodes also is charged, hence we use following formular to evaluate the cost.

$$cost(n) = \frac{Data\ size}{throughput(n)} \times C_2HM \times n$$

According to the Amazon pricing policy, m3.xlarge instance is charged by \$0.405 par hour in a Asia Pacific region (Tokyo). For direct I/O, we denotes n as the total number of instances used at Amazon EC2. Therefore, n becomes the total number of compute nodes in a system with direct I/O, while n is equal to the total number of both computing nodes and I/O buffer nodes in a system with I/O nodes. Since we assume that the number of I/O nodes is equal to the number of computing nodes, n for a systen with I/O nodes becomes 2 times higher than n for a system with direct I/O.

Fig. 12 shows cost comparison between each cache rate and direct I/O, which read or write 100GB data. As shown in the figure, we can see that cost grows as the number of nodes grows while there is an exception for 100% cache, which throughput is proportional to nodes number. As shown in Fig. 12, if cache rate is higher than 50%, the overall cost using I/O burst buffers become lower than direct I/O, however, if cache rate is lower than 50%, using I/O burst buffer cost more.

From above simulations, we can see that if we can achieve a high cache rate, we can achieve a high throughput up to 4-20 times faster than direct I/O as well as a low cost up to 2-12 times lower than direct I/O for 20 client nodes. It is may be difficult for read data cache rate to be higher than 50% (can be possible according to data locality), but it is reasonable for write data can be buffered in buffer queue, since public clouds usually provide instance with large size of memory, and it is easy to achieve a Tera size of buffer queue.

6. Related Work

Cloud computing is becoming a topic of much interest in recent years, not only famous Internet companies like google, Amazon, IBM, Oracle, Microsoft provide public cloud, many companies start to build or have built private cloud for internal computation. Recently hybrid cloud, which is a composition of two or more clouds (private or public) is becoming a hot topic since it allows a business to take advantage of the scalability and cost-effectiveness that a public cloud offers without exposing critical application and data to third-party vulnerabilities, also by using cloud bursting, a small private cloud can easily burst into a large cloud to deal with temporarily request peak. Several research works have been done on hybrid cloud and cloud bursting, Tekin Bicer *el at.*[15] considered a software framework to enable data-intensive computing with cloud bursting, which use a combination of compute resources from local cluster and a public cloud to processing on a geographically distributed data set. Their framework assume computation nodes allocated in both local cluster and public cloud, and data set is geographically distributed, However in our study, data set is stored in local system and in order to obtain a high communication throughput between computation nodes, we assume that nodes used for the same job allocated in the same system. Another cloud bursting application can be seen in Tian Guo*el at.*[9], they introduced a system called Seagull, designed to facilitate cloud bursting by determining which applications should be transitioned into the cloud and automating the movement process at the proper time. Their work focused on determine which applications should be moved to public cloud, and our work focus on the methodology of filling the I/O throughput in cloud bursting.

There are also several paper about data I/O throughput, transferring and data processing in hybrid cloud. Chiba Tatsuhiro *el at.*[16] proposed two high performance multicast algorithms used for transferring large amounts of data stored in Amazon S3 to multiple Amazon EC2 nodes. Tekin Bicer *el at.*[17] also described a middleware that allows the specification of data processing using a high-level API in Amazon S3.

7. Conclusion

In this paper, we propose a I/O burst buffer architecture to burst I/O throughput, provide throughput-based, cost-based and queue write back model, and provide a simulation based on several benchmarks on TSUBAME supercomputer and AMAZON EC2 public cloud. we use several nodes in each system as a I/O buffer nodes, and use data buffering to hide the low throughput between two systems connected by Internet.

From simulation result, we showed that our I/O burst buffer solution can burst I/O throughput about 4-20 times depends on cache hit rate as well as reduce overall cost about 2-12 times when Internet throughput is far smaller than interconnection throughput.

For future work, we plan to implement this I/O burst buffer architecture.

References

[1] Brown, D. L. and Messina, P.: Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale, *The Crosscutting workshop* (2010).

[2] Messina, P. and Bishop, A.: Scientific Grand Challenges in National Security: the Role of Computing at the Extreme Scale, *The National Security workshop* (2009).

[3] Patki, T., Lowenthal, D. K., Rountree, B., Schulz, M. and de Supinski, B. R.: Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing, *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, New York, NY, USA, ACM, pp. 173–182 (online), DOI: 10.1145/2464996.2465009 (2013).

[4] Toshio, E., Satoshi, M., Akira, N., Masamichi, N. and Tadayasu, Y.: Operation of TSUBAME 2.0 Green Supercomputer dealing with Power Crisis, *IPSJ SIG Notes*, Vol. 2011, No. 12, pp. 1–9 (online), available from ⟨http://ci.nii.ac.jp/naid/110008713482/en/⟩ (2011).

[5] Kailasam, S., Gnanasambandam, N., Dharanipragada, J. and Sharma, N.: Optimizing Service Level Agreements for Autonomic Cloud Bursting Schedulers, *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pp. 285–294 (online), DOI: 10.1109/ICPPW.2010.54 (2010).

[6] Bicer, T., Chiu, D. and Agrawal, G.: A Framework for Data-Intensive Computing with Cloud Bursting, *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pp. 169–177 (online), DOI: 10.1109/CLUSTER.2011.21 (2011).

[7] : Eucalyptus, Eucalyptus (online), available from ⟨https://www.eucalyptus.com/⟩ (accessed 2014-05-10).

[8] : Stratos, Apache (online), available from ⟨http://stratos.apache.org/⟩ (accessed 2014-05-10).

[9] Guo, T., Sharma, U., Shenoy, P., Wood, T. and Sahu, S.: Cost-Aware Cloud Bursting for Enterprise Applications, *ACM Trans. Internet Technol.*, Vol. 13, No. 3, pp. 10:1–10:24 (online), DOI: 10.1145/2602571 (2014).

[10] : AMAZON AWS, AMAZON (online), available from ⟨http://aws.amazon.com/⟩ (accessed 2014-06-20).

[11] : Iperf, The Board of Trustees of the University of Illinois (online), available from ⟨http://iperf.fr/⟩ (accessed 2014-06-20).

[12] rklundt: IOR HPC Benchmark, GNU (online), available from ⟨http://sourceforge.net/projects/ior-sio/⟩ (accessed 2014-06-20).

[13] Sato, K., Maruyama, N., Mohror, K., Moody, A., Gamblin, T., de Supinski, B. R. and Matsuoka, S.: Design and Modeling of a Non-blocking Checkpointing System, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC ’12, Los Alamitos, CA, USA, IEEE Computer Society Press, pp. 19:1–19:10 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2388996.2389022⟩ (2012).

[14] Szeredi, M.: SSH Filesystem, GNU (online), available from ⟨http://fuse.sourceforge.net/sshfs.html⟩ (accessed 2014-06-20).

[15] Bicer, T., Chiu, D. and Agrawal, G.: Time and Cost Sensitive Data-Intensive Computing on Hybrid Clouds, *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID ’12, Washington, DC, USA, IEEE Computer Society, pp. 636–643 (online), DOI: 10.1109/CCGrid.2012.95 (2012).

[16] Chiba, T., den Burger, M., Kielmann, T. and Matsuoka, S.: Dynamic Load-Balanced Multicast for Data-Intensive Applications on Clouds, *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID ’10, Washington, DC, USA, IEEE Computer Society, pp. 5–14 (online), DOI: 10.1109/CCGRID.2010.63 (2010).

[17] Bicer, T., Chiu, D. and Agrawal, G.: MATE-EC2: A Middleware for Processing Data with AWS, *Proceedings of the 2011 ACM International Workshop on Many Task Computing on Grids and Supercomputers*, MTAGS ’11, New York, NY, USA, ACM, pp. 59–68 (online), DOI: 10.1145/2132876.2132889 (2011).