Journal of Information Processing Vol.0 No.0 1–6 (??? 1992)

[DOI: 10.2197/ipsjjip.0.1]

Regular Paper

Towards Cloud Bursting for Extreme Scale Supercomputers

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, I/O Bursting Buffer Model

1. Introduction

An increasing number of scientific applications are now running on Supercomputer for high performance computing nodes, large bandwidth and low latency interconnection environment, also a great number of processors for high scalability.

Although supercomputer can offer a high computational cpacity, there are some situations HPC system can not satisfy user's demands even there are still some idle nodes, consider there are 50 nodes available, a user submit a serial application use only one node for an hour, after that, a job using 50 nodes is submitted, and have to wait until previous job finished although there are 49 nodes remainning available. Another problem is the power problem in summer, in order to reduce power consumption, some nodes will be forced to be shut down (peak shift in TSUBAME[1]), reducing numbers of nodes will make the first problem more serous.

One solution is federating supercomputer with a public cloud, moving parts of job and computation to public cloud when there are not enough computing nodes available for user's request, which is known as cloud bursting. Although cloud bursting is used by several companies and already have some sophisticated solutions[2], [3], [4], since there are a significant performance gap between supercomputer nodes and cloud nodes, there will still be several problems when we try to federate a supercomputer with a public cloud, and there are several studies on cloud burst about cost[4], execution time[5], etc.. The biggest problem will be data transfer throughput between two environment, supercomputers usual deal with Gigabyte or even Petabyte input and output of data, low I/O throughput will suffer supercomputer user. This paper focuses on methodology of increasing throughput when we do federation.

In order to increase data transfer throughput, we propose a I/O burst buffer architecture that uses several nodes in each system as a I/O nodes to achieve high throughput concurrent data transferring, and a I/O burst buffer model that uses to switch between I/O burst buffer mode and direct connect mode. Also, public cloud usually charges for nodes usage, using I/O bursting buffer may reduce the computation time, but I/O nodes will be charged for money, we also provide a cost-based model, to reduce the overall cost.

Our contribution can be summary as following:

- An architecture of I/O burst buffer for increasing data transfer throughput between two systems.
- A throughput-based I/O burst buffer model uses to switch between I/O bursting mode and direct connection mode in order to achieve a high data transfer when federating two systems, a cost-based I/O burst buffer model uses to reduce the total cost and a queue model used.
- Evaluating I/O burst buffer architecture and two models by using data obtained from several benchmarks from TSUBAME supercomputer and AMAZON EC2 public cloud.

The remainder of this paper is organized as follow:in section 2, the motivation and background of this study will be introduced, a overview of I/O bursting buffer Architecture including direct connection and I/O bursting buffer will be introduced in section 3, and the model used to switch be-

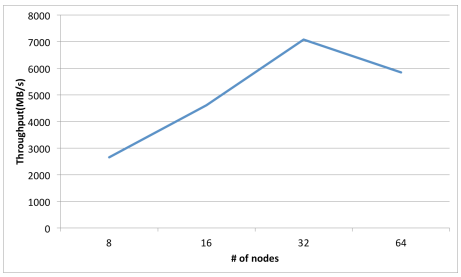

Fig. 1 I/O Throughput to Lustre inside TSUBAME direct mount

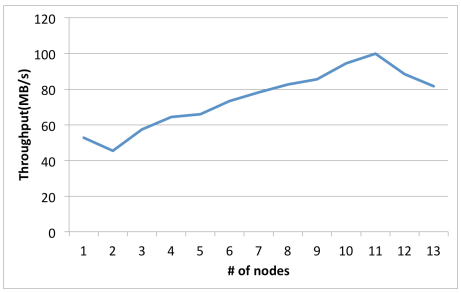

Fig. 2 I/O Throughput from AMAZON EC2 to file system inside our lab using sshfs

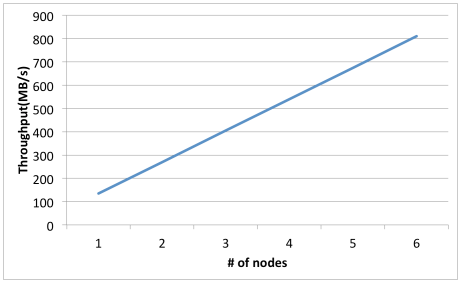

Fig. 3 point to point connection inside AMAZON

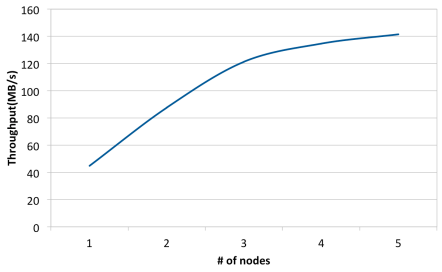

Fig. 4 point to point connection throughput between AMAZON and our lab

tween two modes will be described in section 4, a simulation result of our model will be shown in section 5, and finally, conclusion and related work will be seen in section 6.

2. Motivation and Background

As we know, throughput gap between interconnection network and Internet is extremely large, also numbers of I/O nodes will affect I/O performance. We use Iperf[6], which was developed by NLANR/DAST as a modern alternative for measuring TCP and UDP bandwidth performance, and IOR[7], which is widely used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces.

Fig. 1 shows I/O throughput between TSUBAME V queue nodes, which run on VM on several shared machines 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 [8]. On the other hand, Fig. 2 shows I/O throughput between AMAZON EC2 m3.medium nodes, which have moderate network performance, and a file system machine inside our lab, which has about 1GBit/s Internet access. 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[9],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, about 40-80 times smaller than throughput inside TSUBAME. Also if we compare Fig. 4 with Fig. 2, the maximum throughput If we move some jobs to AMAZON EC2 with input data stored in TSUBAME Lustre, the execution time will increase because of I/O low throughput,for a data sensitive application, it will be devastating, also according to AMAZON's pay-as-you-go policy[10], longer execution time means more cost.

However, if we consider interconnection throughput, as shown in Fig.3, although each node can achieve only 1GB/s, the influence between nodes is extremely small, figure shows a perfect linear line also a strong scalability. Since we can achieve a high interconnection throughput inside a system (HPC system, public cloud), we consider use some nodes as a buffer nodes, using a buffer queue to buffer I/O data and achieve a high throughput.

Although there are some studies about workflow optimization and balance[11], Since data transfer rate is extremely low in Internet compared with interconnection network and data size processed is extremely large, user do not want to settle their nodes across two systems, so in this model, we just consider the situation that all nodes used for the same job are allocated in the same system.

3. I/O Bursting Buffer Overview

An overview of I/O burst buffer architecture and two kinds of connection: direct connection and I/O bursting 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. The main idea is that some of computing nodes serve as a I/O buffer nodes in each system, I/O data will first be buffered in buffer queue in the same system, and then be transferred to final storage system. 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.

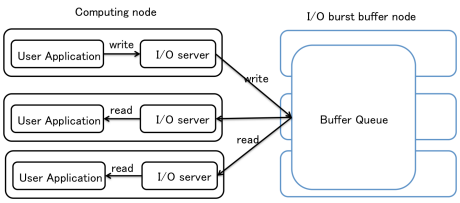

Fig. 5 I/O server and buffer queue

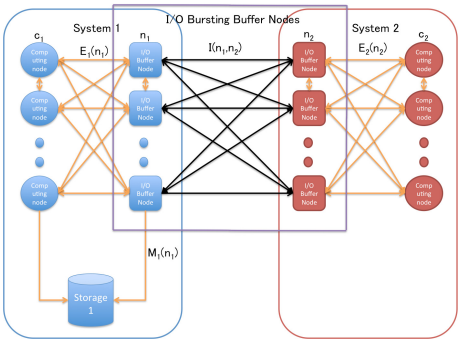

Fig. 6 overall illustrate of I/O Burst Buffer Architecture

3.1 System Environment

First a general system used in following sections is defined as follow:

- 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 are a constant number of public IP addresses can be assigned to some computing nodes.
- 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.

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 buffer nodes*, computing nodes run user's application,which assigned only and I/O burst buffer nodes, which assigned both private IP address.

Fig. 5 is a illustrate of I/O server inside client nodes and buffer queue in I/O burst buffer nodes. In each computing node, there is a I/O server, which is a filesystem client used to buffer I/O data and send to or read from I/O buffer nodes. Among I/O burst buffer nodes, there is a master nodes, which maintain a global buffer queue and a namespace, controll 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.

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 data will be transferred to I/O burst buffer. First, I/O server sends the size of I/O data to master I/O burst buffer node, master node return a list of several I/O

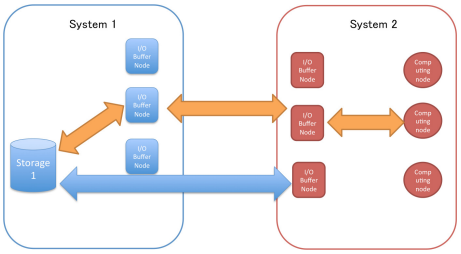

Fig. 7 read and write operation

buffer nodes, then I/O server split I/O data into pieces, and sends each pieces to a I/O buffer node, after data transfer finished, that file will be added to buffer queue and namespace, and this file can be seen by all computing nodes, since we assume that all nodes used by the same job are allocated in the same system, so data consistency can be guaranteed since that. Buffer queue operation including data write back and operation when queue is full will be described in following section.

When user issue a read request, there are two conditions: 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 operations when file is not in I/O buffer nodes or file needs to be write back to storage in another system. 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 achieve 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 rea-

son. 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 use 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 required one more data transfer (I/O nodes to storage in the same system), if storage data transfer throuhput become a bottleneck, two-side buffer will cost more time than one-side buffer. Also, 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.

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 tranferring 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.

4. I/O Burst Buffer Model

Throughput-based Model, Cost-based Model and buffer queue write back model will be described in this section. Throughput-based Model compare one-side buffer throughput with two-side buffer, cost-based model compare one-side buffer cost with two-side buffer, and buffer queue write back model. We make definitions shows in **Table 1** in order to describe our model:

Table 1 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 only I/O nodes in each system have Internet connection.
$I(n_1, n_2)$	Internet throughput using n_1, n_2 I/O buffer 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.
$C_1\_Money(T)$,$C_2\_Money(T)$	Cost for standard node in system 1 and 2 for T time usage
$C_1\_High\_Money(T)$,$C_2\_High\_Money(T)$	Cost for I/O nodes in system 1 and 2 for T time usage, since I/O nodes may use a better network condition, we assume it will cost more than normal nodes.

4.1 Throughput-based Model

In the case of one-side buffer, there are two data transfers: computation nodes to I/O buffer nodes in system 2, I/O buffer nodes in system 2 to storage in system 1, so throughput will be:

$$\text{throughput}_{\text{one-side}} = \min\{D_1(n_2), E_2(n_2)\} \tag{1}$$

In the case of two-side buffer, there are 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, throughput will be:

$$\text{throughput}_{\text{two-side}} = \min\{M_1(n_1), I(n_1, n_2), E_2(n_2)\} \tag{2}$$

In this throughput-based model, these two throughputs are evaluated, and a switch is based on these two values:

$$\begin{cases} \text{throughput}_{\text{one-side}} \geq \text{throughput}_{\text{two-side}} & \text{use one-side} \\ \text{throughput}_{\text{one-side}} < \text{throughput}_{\text{two-side}} & \text{use two-side} \end{cases} \tag{3}$$

4.2 Cost-based Model

In the case of cost-based model, we consider using 1,and 2 total time for transferring unit size of data can be compute as:

$$\begin{cases} T_1 = \frac{1}{\min\{D_1(n_2), E_2(n_2)\}} & \text{one-side} \\ T_2 = \frac{1}{\min\{M_1(n_1), I(n_1, n_2), E_2(n_2)\}} & \text{two-side} \end{cases} \tag{4}$$

here we compute cost by using T_1, T_2:

$$\text{cost}_{\text{one-side}} = c_2 \times C_2\_Money(T_1) + n_2 \times C_2\_High\_Money(T_1) \tag{5}$$

$$\begin{aligned} \text{cost}_{\text{two-side}} = {} & c_2 \times C_2\_Money(T_2) \\ & + n_1 \times C_1\_High\_Money(T_2) \\ & + n_2 \times C_2\_High\_Money(T_2) \end{aligned} \tag{6}$$

In this throughput-based model, these two throughput are evaluated, and a switch is based on these two values:

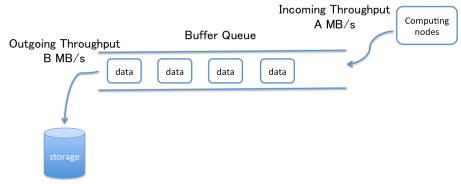

Fig. 8 buffer queue

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

Fig. 10 cost comparison with and without I/O burst buffer

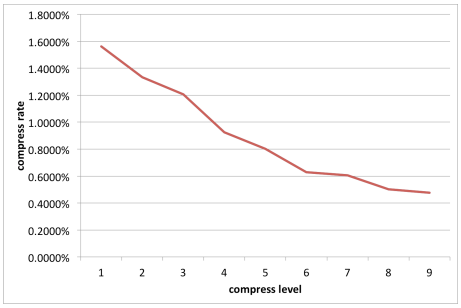

Fig. 11 compress rate

$$\begin{cases} \text{cost}_{\text{one-side}} \geq \text{cost}_{\text{two-side}} & \text{use one-side} \\ \text{cost}_{\text{one-side}} < \text{cost}_{\text{two-side}} & \text{use two-side} \end{cases} \tag{7}$$

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 burst. However 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.

$$T = \frac{m_2 \times n_2}{A - B}$$

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. In this case, jobs running on public cloud have to be moved back to original system until buffer queue is empty.

5. Evaluation

In this section, a simulation will be introduced based on data taking from several benchmarks on both TSUBAME V queue and AMAZON EC2 public cloud Tokyo region, using m3.xlarge instance, which has a high Ethernet condition and 8 vCPUs with 30GiB memory, run Amazon Linux AMI 2014.03.2(HVM) (Fig. 1, Fig. 2. Fig. 3, Fig. 4).

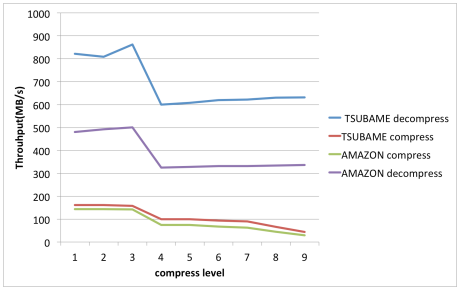

Fig. 12 compress time on TSUBAME V queue and AMAZON EC2

Fig. 13 throughput comparison with and without compress

Fig. 14 cache hit and miss throughput comparison

For throughput between two systems and inside system, from benchmark data, it shows it is hard to achieve a high throughput with only one nodes, and also there is a limit on maximum throughput between two systems and inside system. Although incresing nodes can increse throughput before reach the maximum, throughput achieved by each nodes decrease because of conflict. For these reason, we use following formular for throughput between two systems and inside system.

$$throughput = -Ax^2 + Bx + C \quad A, B > 0 \tag{8}$$

we use following equations to determine A, B, C

$$\begin{cases} -A + B + C = throughput_{one} \\ \frac{B}{2A} = n_{max} \\ -An_{max}^2 + Bn_{max} + C = throughput_{max} \end{cases}$$

First, we compare throughput with and without I/O buffer nodes. Since it is hard for one node to fully utilize Internet and Ethernet bandwidth, according to Fig. 2, we assume that one node can achieve half of maximum bandwidth. We can see from Fig. 9, when interconnection throughput is larger than Internet throughput, our I/O buffer can achieve a higher throughput.

Then, we compare overall cost by using I/O burst buffer,

we compare throughput with and without compression, Fig. 11 shows compress and Fig. 12 shows compress and decompress time on TSUBAME V queue and AMAZON EC2 with a different compress level by using zlib[12]. We can achieve a high compress rate, but the throughput is low, up to 160MB/s, depends on compress level, it is hard to find a compress library can compress data faster and smaller, so the compress throughput may become a bottleneck. On the other hand, the compress rate is high, since the buffer size is limited, if we can make data smaller, it means we can increase the buffer hit rate, Fig. 14 shows a throughput comparison between cache miss and hit.

Fig. 13

6. Conclusion and Future Work

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 can increase I/O throughput when Internet throughput is far smaller than interconnection throughput as well as reducing the overall cost.

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

References

[1] : TSUBAME, gsic (online), available from ⟨http://tsubame.gsic.titech.ac.jp/⟩ (accessed 2014-05-20).

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

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

[4] Guo, T., Sharma, U., Shenoy, P., Wood, T. and Sahu, S.: Cost-Aware Cloud Bursting for Enterprise Applications, USENIX 2012 Annual Technical Conference (2012).

[5] Bicer, T., Chiu, D. and Agrawal, G.: Time and Cost Sensitive Data-Intensive Computing on Hybrid Clouds, CCGRID '12 Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) (2012).

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

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

[8] Sato, K., Maruyama, N., Moody, K. M. A., Gamblin, T., de Supinski, B. R. and Matsuoka, S.: Design and modeling of a non-blocking checkpointing system, SC '12 (2012).

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

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

[11] Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B. and Turck, F. D.: Network-aware impact determination algorithms for service workflow deployment in hybrid clouds, CNSM '12 (2012).

[12] : zlib, private (online), available from ⟨http://www.zlib.net/⟩ (accessed 2014-05-20).