DOI: 10.14529/jsfi160204

InfiniCloud 2.0: Distributing High Performance Computing across Continents

Jakub Chrzeszczyk[1]***, Andrew Howard***[1]***, Andrzej Chrzeszczyk***[2]***, Ben Swift***[1]***,***
Peter Davis[1] ***Jonathan Low***[3]***, Tin Wee Tan***[45]***, Kenneth Ban***[5]

InfiniCloud 2.0 is the world's first native InfiniBand High Performance Cloud distributed across four continents, spanning Asia, Australia, Europe and North America. The project provides researchers with instant access to computational, storage and network resources distributed around the globe. These resources are then used to build a geographically distributed, virtual supercomputer, complete with globally-accessible parallel file system and job scheduling. This paper describes the high level design and the implementation details of InfiniCloud 2.0. Two example applications types, a gene sequencing pipeline and plasma physics simulation code were chosen to demonstrate the system's capabilities.

Introduction

The original InfiniCloud system, presented at Supercomputing Frontiers Singapore in March 2015, enabled researchers to quickly and efficiently copy large volumes of data between Singapore and Australia, as well as to process that data using two discrete, native InfiniBand High Performance Clouds [8]. It also provided an opportunity to establish a detailed baseline of compute, memory, storage and network performance of native Infiniband High Performance Cloud [11].

While the unique capabilities of InfiniCloud enabled new ways of processing data, it also inspired a whole new range of research questions: Can the entire capacity of the system be aggregated? Do entire data collections need to be copied for processing (even with a 99% effective circuit efficiency delivered using extended InfiniBand), or can data be accessed in place? How does the InfiniCloud design scale to an arbitrary number of sites? How we ensure a consistent state of all InfiniCloud clusters? And finally, can the resources across four continents be joined together using the InfiniCortex fabric to create a Galaxy of Supercomputers [14]?

In this paper we aim to explore these research questions and propose new ways of utilizing distributed computation, storage and network resources, using a variety of novel tools and techniques. We aim to provide a unified interface allowing users to transparently access resources at each and every site using a standardized set of CLI, GUI and API tools. We also take advantage of the expansion and enhancement of the InfiniCortex fabric which took place in 2015 [12], which includes full support for InfiniBand subnets and routing, greater available bandwidth and last but not least the growing number of participating sites. Finally, we demonstrate new and unique capabilities of the system by deploying example scientific applications across geographically distant sites.

[1]The Australian National University, Canberra, Australia
[2]Jan Kochanowski University, Kielce, Poland
[3]A*STAR Computational Resource Centre (ACRC), Singapore
[4]National Supercomputing Centre (NSCC), Singapore
[5]Dept. of Biochemistry, Yong Loo Lin School of Medicine, National University of Singapore, Singapore

1. The Network

The InfiniCortex network, which is crucial to the existence of InfiniCloud, has expanded from an experimental service connecting Australia and Singapore, through a cycle of significant extension, improvement in operational stability and enhancement of fabric isolation throughout 2015 to create for a time an InfiniBand Ring Around the World.

The underlying physical network is constructed using interconnected Advanced Layer 2 Services (AL2S) provided by National Research and Education networks to deliver a set of loss-less 10Gbs Layer 2 channels reaching around the globe. Each presenting a fixed latency and data rate. To date, the network has been instantiated on-demand to support ISC and SC, during 2016 a permanent InfiniCortex core will be created linking Europe, Asia and the Americas.

A significant characteristic of the InfiniCortex network is the optimised presentation of both high bandwidth and fixed latency using InfiniBand as the underlying transport protocol. When compared to traditional IPv4 as the transport protocol the deterministic nature of InfiniBand delivers a direct improvement of large data set transfer data rates. While we have demonstrated this capability at an international scale we believe that it provides greater advantages at a Campus, Metro or Regional scale with lower inter-site latency to scale transparently beyond a single data centre or facility. (Figure 1, Figure 2)

1.1. Connecting to Europe and additional US based facilities

The most significant change to the InfiniCortex in 2015 was the establishment of native InfiniBand connectivity to Europe, thanks to help and support from many National Research and Education Networks including AARNet, Internet2, TEIN*CC, SingAREN and Geant. This enabled InfiniCortex sites to connect to Warsaw and Poznan in Poland as well as Reims in France, allowing the University of Reims to become a new InfiniCloud node (Figure. 2). Connectivity to the East Coast USA has been maintained and also been significantly enhanced, enabling Stony Brook University in New York to also join InfiniCloud project.

1.2. Transition to 100Gbit/s Networks and InfiniBand

The second significant change to the InfiniCortex 2015 was the phase change from 10G and 40G to 100G as an advanced research network service. This allowed additional virtual circuits to be created between Asia and sites located in North America, allowing higher data transfer performance, supporting a greater level of concurrency for the increased volume of network traffic.

We anticipate in 2016 as 100G network capabilities become more prevalent and affordable and the implementation of 100G EDR based InfiniBand fabrics more widespread, that the capabilities of the InfiniBand Extension and Routing devices underlying the InfiniCortex will follow, allowing the removal of the current bandwidth impedance mismatch of a 10G network connecting FDR 56G or EDR 100G InfiniBand fabrics and supporting line rate 100G inter-fabric communication in the near future.

This will provide the opportunity to exploit the deterministic nature of data access in RDMA capable applications operating in this environment. We have begun exploring the use of a distributed file system layer which is tuneable for the InfiniCortex network characteristics. Allowing a change in the conventional paradigm of staging large data set into and out of a remote facility for processing to one of data access in place using the underlying opportunistic caching

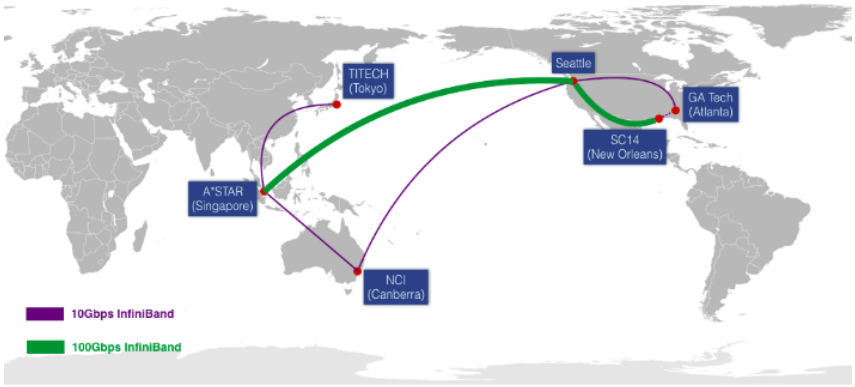

Figure 1. InfiniCortex 2014 network diagram

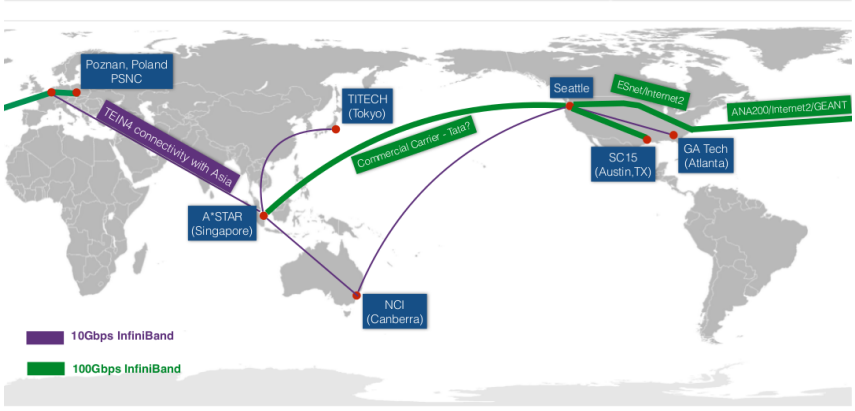

Figure 2. InfiniCortex 2015 network diagram.

capability of the filesystem to optimise inter-facility traffic. Our initial investigation has shown that gene alignment, sequencing and annotation applications with a primarily serial data access pattern perform well in this environment. Analysis and characterisation of the performance of additional classes of applications at both a metro scale and global scale environment are underway.

1.3. Routable InfiniBand

The third key change was transition from a single, centrally managed InfiniBand fabric to a interconnected pool of independent InfiniBand subnets, connecting to each other using the Obsidian Strategics R400 InfiniBand Router. This transition has significantly enhanced InfiniCortex operational stability - by eliminating crosstalk between local subnets, reducing latency in Subnet Manager communication by orders of magnitude and containing any network issues within a single site, minimizing the impact on other sites. All this was achieved without any impact on network performance and the ability to efficiently and securely transfer data. All the key components of the InfiniCortex network implementation are listed in (Table 1)

Progressive enhancements and improvements to the InfiniCortex have laid a foundation for building the next generation of geographically distributed High Performance Computing and Clouds. As the network matures from an experimental platform to a operational service, ongoing performance characterisation is being undertaken and the findings will be presented in a future paper, the main focus of this paper however is on the Cloud infrastructure.

Table 1. InfiniCortex long-range InfiniBand equipment

InfiniBand extenders	Obsidian Strategics Longbow^{TM} E100
InfiniBand routers	Obsidian Strategics Crossbow^{TM} R400
Layer 2 connectivity	Multiple loss-less, QoS-enabled 10GE circuits

1.4. Private, software-defined networks

With the latest versions of the OpenStack and Obsidian products, it is now fully supported to implement a number of private, secure, software-defined InfiniBand networks. Each project described in this paper was contained within a dedicated InfiniBand subnet, with the option of reliable, secure, high-performance external connectivity if required.

2. The Cloud

The founding requirements of InfiniCloud 2.0 were i) simplifying and standardizing access to the resources provided by the distributed system, ii) enabling new ways of using the network capabilities for interacting with data iii) enabling easy scaling to more than two sites.

2.1. InfiniCloud rationale and the existing solutions

The original motivation behind exploring the novel topics of long-range, routable InfiniBand, as well as connecting it to the Cloud was solving one of the ultimate challanges underlying Cloud Computing - the ability to work with large volumes of data in a geographically distributed context. While Public Cloud resources are available on demand and at low cost to users anywhere in the World, the easy access to computational and storage resources becomes much more difficult as soon as we add the requirement of accessing large quantities of data from outside any of the popular Public Clouds.

One issue with Public Cloud data transfer is performance - from our experience, copying a typical genomics sample of 300GB out of a Cloud hosted on the East Coast US to Singapore can take as long as 24 hours. This itself makes it impossible to enable real-time aggregation of compute and storage to achieve HPC-grade performance - where the boundary is previous-generation interconnect performance - 10Gbit/s or 1GB/s. Also significant the cost of data transfer out of Public Clouds is several orders of magnitude higher than the cost of compute within those Clouds.

To provide an example - Amazon EC2 instances compute instances are priced from 0.01$/hr for t2.micro to 1.68$/hr for c3.8xlarge. At the same time, data transfer rates are from 0.02$/GB for inter-region transfers within Amazon to 0.09$/GB for data transfers to the Internet [1]. With these rates, even if we were to hypothetically run transfers at the speed we're aiming for (1GB/s) which we weren't able to achieve, this would amount to 72$/hr for inter-region transfers or to 324$/hr for transfers to the Internet.

In any possible case, the comparison of the cost of compute and network traffic (out by two orders of magnitude) clearly states that the use case discussed in this paper is not possible to implement efficiently using today's Public Cloud model.

The use cases typical to most cloud users do not involve moving such huge volumes of data on a regular basis hence this is not a major focus for the Public Cloud operators. However, from

our experience, organizations such as genomic consortium's and large multi-national companies do have such requirements.

The advantage that InfiniCloud 2.0 is proposing is a mechanism which can enable virtual aggregation of geographically distant data centres - in a secure and energy efficient way utilising the next generation of metro, regional and global networks. This is done by connecting Cloud to a global-range, routable Infiniband fabric for the first time.

As no such capability is currently available from any of the Public Cloud providers, we envision that just like dial-up was replaced by broadband links supporting unlimited data transfer, inefficient and expensive data transfer rates charged per gigabyte will ultimately be replaced with solutions similar to those explored by this collaboration using Research and Education networks and described in this paper.

2.2. Cloud Architecture

In order to fulfill some of these requirements, a significant re-design of the system was required. The underlying model of independently operated, site-contained clusters which leverage the InfiniCortex fabric for high performance, efficient data transfer between cloud instances was replaced with a more tightly coupled, experimental architecture, utilising a central Cloud Controller with Compute Nodes distributed around the globe.

This change implemented the first requirement: a uniform user interface with a single CLI / GUI / API interface accessible to users. It also removed the necessity to manually synchronize state between different clusters which was a requirement in the previous, loosely-coupled design.

With the architecture described above, it is crucial to carefully consider the characteristics of the network, especially in terms of latency, in order to make optimal component placement decisions. It is highly recommended that consideration of placement of the critical components (Cloud Controller, Subnet Manager, and at later stage - also parallel file system instances and the job scheduling instance) occurs in a way to ensure that the latency between all nodes is consistent.

In the scenario presented in this paper, Singapore provided the best location for these services. A consistent, 300ms round-trip latency from Canberra, Reims and New York as well as access to 100Gbit/s connectivity made the South-East Asian city-state the optimal location for the Cloud Controller and the Master Subnet Manager.

Two distinct pools of compute nodes were also hosted in Singapore, joined by more nodes hosted in Canberra, New York and Reims, all connected with high performance RDMA and IPoIB connectivity provided by the InfiniCortex.

The last but not the least, given the nature of the system, users needed to be able to have fine-grained control over scheduling of the resources requested, be it compute or storage. Some applications might be suitable to be distributed around the world in a round-robin fashion, while others may require user-defined, role, facility, regional, sovereign territory or geographically based scheduling.

2.3. Cloud implementation

All InfiniCloud systems run the following hardware (Table 3) and software stack (Table 2).

The deployment of the system begins with building the Cloud Controller in Singapore.

Table 2. InfiniCloud 2.0 software stack

Operating System	CentOS 7.1 x86_64
InfiniBand drivers	Mellanox OFED
OpenStack version	Kilo (customized)

Table 3. InfiniCloud 2.0 hardware configuration

CPUs	Intel Haswell (Singapore), Sandy Bridge and Ivy Bridge (other sites)
Memory	64GB-256GB
Interconnect	Mellanox FDR
Local storage	Intel DCS family SSDs (Singapore), SAS HDDs (other sites)

2.3.1. Cloud Controller

Due to the complexity of this component, it is performed in a multi-step process, starting with a kickstart build and then switching to a puppet configuration management solution based on widely adopted PuppetForge OpenStack modules [5] [6] . After the base installation is completed, the custom, out-of-tree InfiniCloud2.0 specific patches are applied, which completes the Cloud Controller installation.

2.3.2. Compute Nodes

InfiniCloud 2.0 Compute nodes are much simpler than the controller node and as such can be kickstart-built. As soon as the freshly-built nodes report to the Cloud controller using the InfiniCortex fabric, their configuration is pushed and they become ready for operation.

Two compute nodes are build in each of the InfiniCloud 2.0 locations. Additional capacity is available in Singapore for infrastructure services, such as a virtual parallel file system.

2.4. Resource scheduling

The default behaviour of the OpenStack scheduler is to allocate virtual instances to compute nodes in a round robin fashion. While this is a reasonable approach for a homogeneous cluster hosted in one location, it is unsuitable for a globally distributed system, where - as discussed in the previous sections - the locations of particular components have to be carefully optimized for consistent latency.

2.4.1. Availability zones

For the above reason, a separate availability zone was created for each location. Each zone (Australia, Asia, Europe, USA) corresponds to a physical location (Canberra, Reims, Singapore, New York) and holds all the compute resources that are hosted in this location.

On top of these, two additional availability zones were added: local and distributed. The local zone is reserved for infrastructure use (such as Sun Grid Engine head node or BeeGFS parallel file system nodes) and instances allocated to this zone will always be launched in Singapore. The distributed zone stripes across all the locations and provides a simple mechanism to distribute workloads across the remote nodes. This is illustrated in (Listing 2.4.1 and Listing 1). **Remark:** For the purpose of this paper and for clarity let us assume that host aggregates and availability zones are synonyms.

```
# availability zone example
[root@ics111 ~(keystone_admin)]# nova aggregate-list
+----+-------------+---------------------+
| Id | Name        | AvailabilityZone    |
+----+-------------+---------------------+
| 1  | local       | -                   |
| 7  | distributed | -                   |
| 8  | singapore   | singapore           |
| 9  | australia   | australia           |
| 10 | europe      | europe              |
| 11 | america     | america             |
+----+-------------+---------------------+
```

Listing 1. InfiniCloud 2.0 availability zones

```
# zone to host mapping example
[root@ics111 ~(keystone_admin)]# nova aggregate-details 7
+----+-------------+-----------------------------------+-------------------+
| Id | Name        | Hosts                             | Metadata          |
+----+-------------+-----------------------------------+-------------------+
| 7  | distributed |'ica120.infinicloud.nci.org.au',   | 'distributed=true'|
|    |             |'ica121.infinicloud.nci.org.au',   |                   |
|    |             |'ics130.infinicloud.a-star.edu.sg', |                   |
|    |             |'ics131.infinicloud.a-star.edu.sg', |                   |
|    |             |'icu143.infinicloud.stonybrook.edu',|                   |
|    |             |'icu144.infinicloud.stonybrook.edu',|                   |
|    |             |'icf157.infinicloud.univ-reims.fr', |                   |
|    |             |'icf158.infinicloud.univ-reims.fr'  |                   |
+----+-------------+-----------------------------------+-------------------+
```

Listing 2. InfiniCloud 2.0 host to availability zone mappings

2.4.2. Instance types

While launching Cloud instances, the user has an ability to explicitly specify the intended availability zone. However, to enable a high degree of automation, it is recommended to create dedicated instance types which are linked to a particular availability zone and/or a scheduling pattern. This can be implemented using instance type metadata. This is illustrated in Listing 5. After the metadata is set, when instance types such as "local.2c.8m.170d" (local storage) or "geo.8c16m20d" (distributed compute) are requested, instances will be scheduled in the corresponding locations. These instance types can be referenced directly in applications such as ElastiCluster which is described in the next section. Examples of creating instances in a specific location or across all locations is demonstrated in Listing 3 and Listing 4.

```
[root@ics111 ~(keystone_admin)]# nova boot --flavor geo.2c2m20d --key-name oskey
--image BioPipeline_v0.8.6.7 --num-instances=4
--availability-zone singapore|australia|america|france singlezone
```

Listing 3. Creating cloud instances on a particular continent

```
[root@ics111 ~(keystone_admin)]# nova boot --flavor geo.2c2m20d --key-name oskey
--image BioPipeline_v0.8.6.7 --num-instances=4 distributed
```

Listing 4. Creating cloud instances across all continents

```
# instance types
[root@ics111 ~(keystone_admin)]# nova flavour-list --all   extra-specs
+---------+---------------+------+----+---------+----+------+---------------+
| ID      | Name          |Memory|Disk|VCPU|Public|extra_specs                |
+---------+---------------+------+----+----+------+--------------------------+
| 0e...db | local.2c8m170d| 8192 | 20 | 2  | True |{u'local': u'true'}       |
| 3a...5e | geo.8c16m20d  |16384 | 20 | 8  | True |{u'distributed': u'true'}|
| 60...7f | local.2c8m120d| 8192 | 20 | 2  | True |{u'local': u'true'}       |
| 83...52 | geo.4c8m20d   | 8192 | 20 | 4  | True |{u'distributed': u'true'}|
+---------+---------------+------+----+----+------+--------------------------+
```

Listing 5. InfiniCloud2.0 host to availability zone mappings

2.5. Communication patterns

There are three main types of network traffic which are relevant to operating a geographically distributed OpenStack cluster: Message queue traffic (AMQP), cloud endpoint traffic (HTTP), and downloading images (HTTP). All these protocols are native to IP, so IPoIB is used to carry the relevant traffic.

2.5.1. Bandwidth and latency considerations

Both message queue and endpoint traffic is only minimally bandwidth intensive and tolerant to large latency, so no tuning was required. In the case of images, while underlying HTTP protocol is robust and built-in auto-tuning mechanisms are useful, it is not able to fully utilize the available bandwidth over a 10Gbit/s connection with a 300ms round-trip latency. While the default configuration is usable, initial image caching can take several minutes (in case of the processing method described in this paper, this is a one off operation, so this delay is acceptable). This could be addressed with additional network tuning or the use of a custom image transport mechanism, ideally using native RDMA communications. Implementing such a mechanism, however is outside of the scope of this paper.
The amount of bandwidth consumed by the Cloud itself is minimal, and nearly all the bandwidth can be used by the applications running in virtual instances.

3. The Applications

This section will cover a selection of example applications that can take full advantage of the geographically distributed, High Performance Cloud: BeeGFS, Geopipeline and Extempore.

3.1. BeeGFS

One of the key advantages of high bandwidth, RDMA-capable interconnect spanning across continents is providing the ability to efficiently work with data. In our previous work, this mainly meant high speed data transfers and/or data synchronization. However, in many cases it might be more efficient to access datasets in-place, without the need to move parts or even the entirety of the data set. In this experiment, we use a BeeGFS parallel filesystem cluster hosted in Singapore to export data to consumers in Australia, Europe and the US [2].

The BeeGFS cluster used in this experiment consisted of three nodes. The first node was running management *(fhgfs-mgmtd)*, metadata *(fhgfs-meta)* and storage *(fhgfs-storage)* processes. The other two nodes were running solely the *fhgfs-storage* process. The storage space was presented as a qcow2 300GB ephemeral drive which was stored on a a single SSD drive. BeeGFS performance is typically determined by backing store read/write bandwidth - in our case 3xSSD drives, each capable of 500MB/s read/write, provided a 1500MB/s theoretical maximum I/O capability and given some overhead in the parallel filesystem layer, 1000MB/s is a typical value delivered. The network bandwidth needs to match or exceed the storage bandwidth. In our case, 56Gbit/s is available to local clients and 10Gbit/s is available to remote clients so this requirement is met. CPU performance and memory allocation have less impact on BeeGFS performance, however BeeGFS cluster under load manifests medium CPU utilization and sufficient memory is essential to support high-latency of an intercontinental link due to data buffering. We used 2 CPU cores and 8GB of RAM per server. All data transfer communications use native RDMA.
While mounting a file system across a link with such a large Bandwidth Delay Product (BDP) brings a number of technical challenges, we were able to derive the tuning parameters optimized for such scenario.

3.1.1. BeeGFS configuration for high bandwidth-delay products

This cluster was then specifically tuned for a large Bandwidth Delay Product inherent in high latency links. The key configuration value are included in (Listing 3.1.1). An important remark is that this results in an increased memory utilization, so the virtual instances need to be created with sufficient memory allocation to support this requirement. The default client time-out values were also increased by an order of magnitude.

```
> connRDMABufSize                 = 8192
> connRDMABufNum                  = 128

< connRDMABufSize                 = 65536
< connRDMABufNum                  = 260
```

Listing 6. BeeGFS tuning parameters

3.1.2. Optimizing data access patterns

In order to be able to obtain performance near or matching the local performance, the I/O patterns in use also need to be optimized. We aim to optimize data access methods to ensure as much data remains in flight as possible and to ensure that data is striped across all available network links and servers. In order to achieve this, we used:

- multiple clients,
- multiple threads (10-20),
- large block sizes (10MB and more)

With the sufficient degree of parallelism and sufficient block size, we were able to achieve satisfactory performance, nearly matching what is possible to achieve locally: Write operations reached 1GB/s, saturating 10Gbit/s link (Figure 3). Read performance was lower, but still very good, reaching 700MB/s.
The authors consulted BeeGFS support team about write:read disparity. This behavior is often observed, due to the characteristic that results in write operations being cached and reorganised

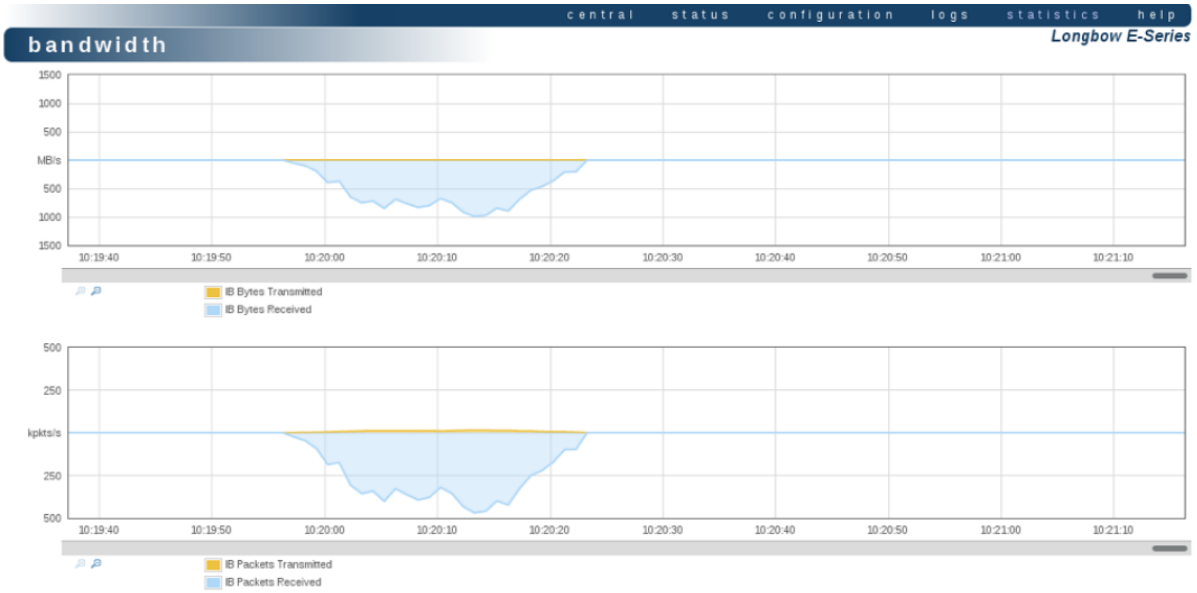

Figure 3. Network utilization diagram for Canberra-Singapore link during BeeGFS data access

more efficiently than read operations. Often, this allows the storage back-end to aggregate write operations in a more efficient manner. When reading data, it is sent out to the clients as quickly as possible, and more data is streamed read from the disks as it is needed. This doesn't allow as much room for optimization as the case of write operations. Such performance profile is not unique to BeeGFS and is often observed with other parallel filesystems, such as Lustre [4] [15].

3.2. Geopipeline

Geopipeline is a set of tools implementing computational genomics codes on a set of geographically distributed hardware. It consists of two main components: ElastiCluster and Biopipeline - and also relies on a parallel file system - in our case BeeGFS

3.2.1. ElastiCluster

ElastiCluster is a software suite implementing on-demand HPC Cluster capability in the Cloud. By default it uses Ansible configuration management, NFS file sharing and Sun Grid Engine job scheduler [3]. All these components can be customized as required.
We configure OpenStack and ElastiCluster in a way which supports running a BeeGFS storage cluster in a central location and then a geographically distributed HPC cluster spanning multiple locations which connect to the central storage cluster.
After the relevant configuration has occured as detailed in Section 2.2 and Section 2.3, the geographical location of the compute instances becomes irrelevant. When ElastiCluster is instantiated, it deploys a head node in the central location and a number of virtual compute nodes spread across continents. The compute nodes then mount the remote BeeGFS parallel file system and report to the Sun Grid Engine scheduler running on the head node and are ready to run jobs. Physical distance between the nodes running virtual instances is abstracted away and the applications run and behave exactly in the same way as if it were running on a local cluster hosted in a single datacentre rack. (Figure 4) and (Figure 5) represent a state of the virtual cluster after launch and during processing.

Figure 4. Ganglia monitoring interface after starting a geographically distributed cluster

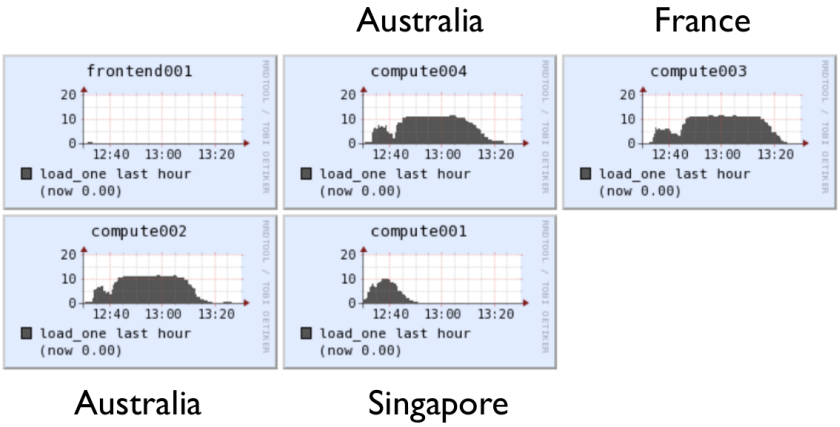

Figure 5. Resource utilization across the distributed cluster members

3.2.2. Implementation of variant calling genome analysis pipeline

Next, we demonstrate the on-the-fly provisioning and setup of a virtual machine which can be used to parallelize a genomic analysis workflow. We selected a clinically relevant workflow, called variant calling, which takes genomic sequences from cancer samples and detects mutations in genes that could be used to determine the prognosis of a patient, or to identify potential chemotherapy drugs that could be used for treatment. Because each cancer sample can be analysed separately, the workflow is amenable to simple asynchronous parallelization without any interprocess communication. Each compute node is configured with 12 CPU cores, 24GB RAM, 20GB system disk and 100GB ephemeral scratch space.

In this workflow, genomic sequences are processed in a pipeline through a series of steps using different applications to identify and annotate mutations (Figure. 9). We use a pipeline appli-

cation to orchestrate the steps in processing and to distribution the processing to the compute nodes using the SGE scheduler.

(Figure. 6) and (Figure.7) illustrate typical local CPU and memory utilization on one node during example run of the above pipeline. These metrics were recorded using Ganglia. Figure 8 captures typical aggregate I/O usage patterns measured against the parallel file system using sysstat package. These graphs clearly show that these applications are primarily CPU bound and in certain steps (particularly alignment) storage intensive, both in terms of read and write. All applications are configured to run a number of threads matching the number of available cores. Memory utlization is moderate. Network utilization is high at times of heavy storage utilization and very low at all other times. Native RDMA is used for data movement. TCP over IPoIB is used for SGE control streams and monitoring - all these components combined use only minimal network bandwidth, measured in kilobytes a second.

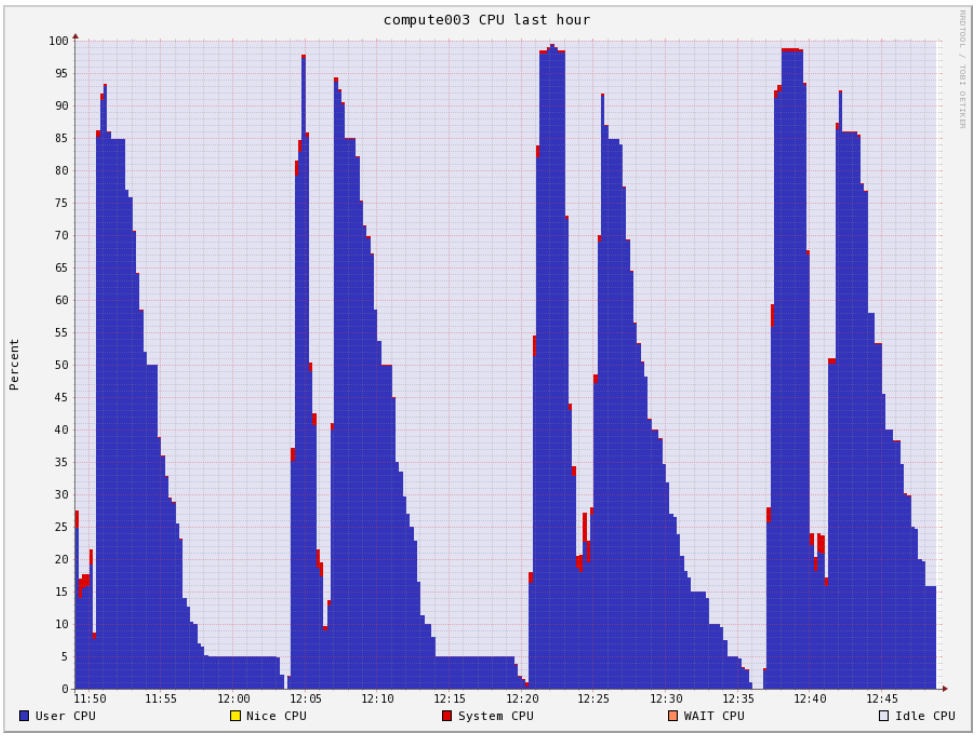

Figure 6. CPU utilization on a node across many test iterations

1. Genomic sequences from each cancer sample are processed with an aligner - an application that compares the sequences to a human reference genome sequence and identifies the position and alignment of each sequence from the cancer samples.
2. The files from each cancer sample are processed by a variant caller program, which compares the aligned sequences to the human reference genome sequence to identify variations (substitutions, insertions, deletions) in the cancer samples.
3. The variant files from each cancer sample are annotated. A specialized application compares each variation to multiple databases to identify what potential effects of each mutation have on regions in the genome.

The applications are pre-installed in the VM images together with their dependencies to enable portability. An example output is shown on (Figure 3.2.2). The reference datasets required by the aligner, variant caller, and annotation tool, are located on remotely mounted BeeGFS. Depending on the expected data access pattern, the datasets can be accessed in place or staged

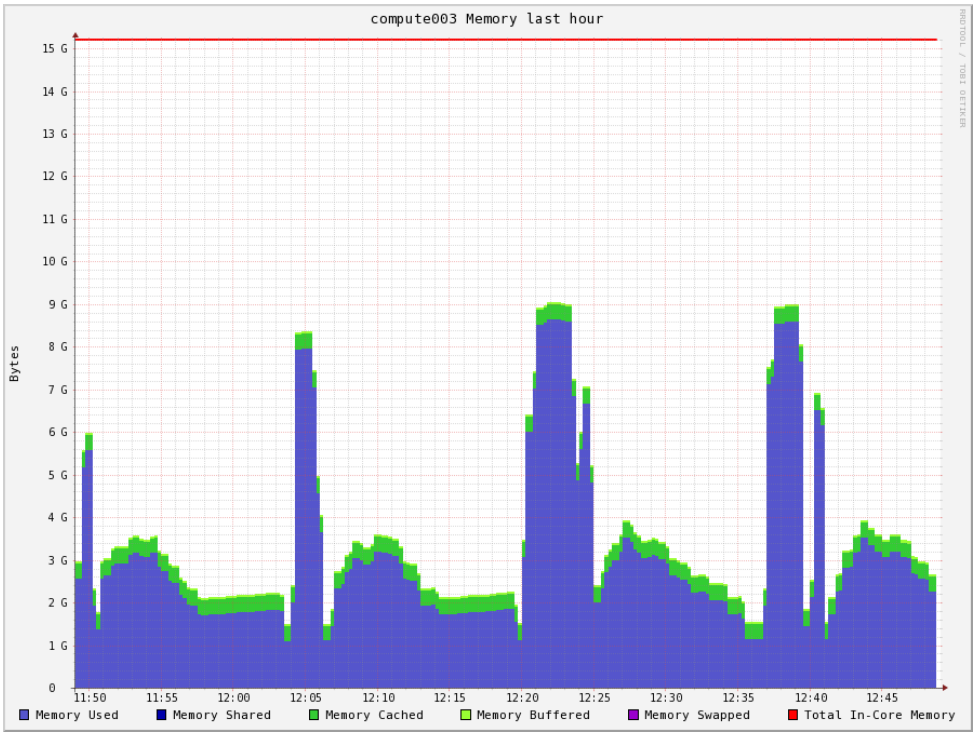

Figure 7. Memory utilization on a node across many test iterations

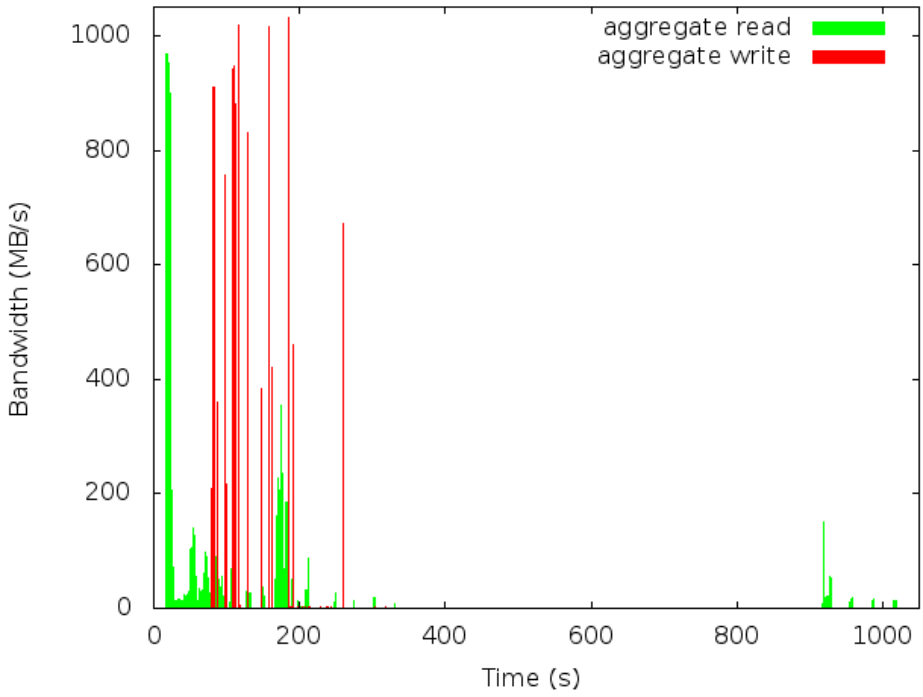

Figure 8. Parallel filesystem read/write bandwidth across single test iteration

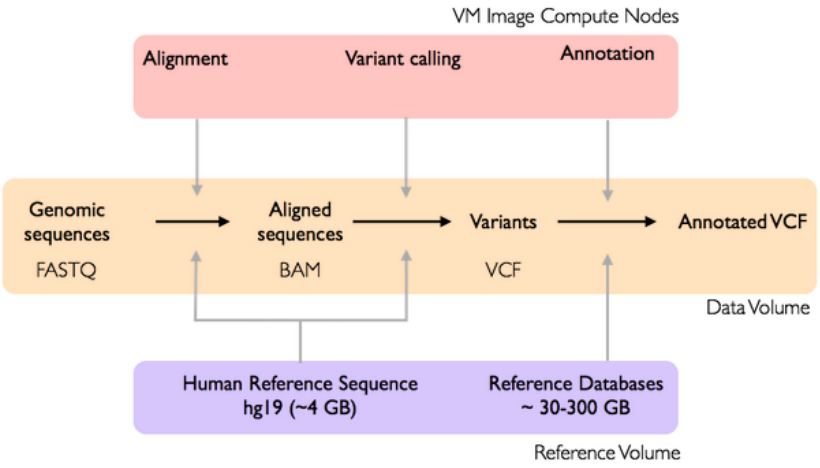

Figure 9. Workflow for variant calling of genomic data from cancer samples

into local scratch. After the processing is complete, the output is also saved on the BeeGFS parallel file system.

```
================================================================================
                    Starting Pipeline at 2016-03-15 21:04
================================================================================

================================= Stage align [AD365_S3_L001]==================
...
============================== Stage variant [AD363_S2_L001] ==================
...
============================ Stage annotate [AD363_S2_L001] ==================
...
======================================== Pipeline Finished ====================
21:48:07 MSG:  Finished at Tue Mar 15 21:48:07 UTC 2016
21:48:07 MSG:  Outputs are:
                annotate/AD407_S5_L001.avinput
                annotate/AD407_S5_L001.hg19_multianno.csv
                annotate/AD363_S2_L001.hg19_multianno.csv
                annotate/AD422_S6_L001.avinput
                annotate/AD422_S6_L001.hg19_multianno.csv
                ... 8 more ...
```

Listing 7. Example output from the pipeline

3.2.3. Geopipeline performance analysis

Performance metrics gathered during the test show that the execution time is determined mostly by CPU performance and, while average I/O bandwidth is low, it can be very streaming read and write intensive in short burts. Given highly parallel nature of the workload, performance can be further optimized by increasing the number of cores available for processing, increasing per-core performance and at later stage increasing storage performance.
The CPU utilization metrics gathered throughout the run show a large variation between completion times across different sites. This is due to heterogenous hardware setup, particularly:

- difference in per-core CPU performance (up to 50% higher per-core performance between Haswell and Sandy Bridge)

- faster local data staging (Singapore nodes don't have to compete for WAN bandwidth with other sites)
- local storage performance (SSD-equipped machines can achieve 50% higher storage bandwidth)

The main focus of this paper is functionality more than performance. The goal of this research is demonstrating the ability to aggregate geographically distant compute resources and enabling the users to efficiently work with large amounts of data over large distances, hence we don't see performance disparity as a problem. For this reasons we haven't performed detailed analysis of performance data or attempts to optimize the pipeline further. This can be a subject of further research. More information on performance analysis and comparison between different types of Public and Private Clouds can be found in [7] and [11].

3.3. Extempore

In contrast to the high bandwidth genome analysis pipeline, we also explored the potential of the InfiniCortex network for interactive latency-sensitive applications using the Extempore "live" interactive HPC programming environment [13]. Extempore allows the HPC programmer to hot-swap running code on-the-fly. It supports seamless integration with C/Fortran (C-ABI compatible) in an interactive "live programming" workflow for Exploratory HPC. For the purpose of this demonstration, we ran four Extempore instances with 8 CPU cores, 64GB RAM and 20GB local SSD. All communications used MPI running over IPoIB. We ran 32 MPI processes in total and the code was mostly CPU and communications intensive, with little memory and storage utilization. The emphasis of this experiment was to test the feasibility of real-time interaction with running computations in a geographically distributed environment like ours, so no detailed resource utilization or execution time measurements were taken.
For the Supercomputing '15 event, we ran an interactive plasma physics simulation (Figure 10) from the SC show floor to demonstrate the possibility of interactive steering and simulation across the globe. We used a particle-in-cell distributed (MPI) plasma physics code, based on codes by Viktor Decyk (UCLA) [9]. All aspects of the simulation could be modified:

- re-definition of constants (time step, electric/magnetic fields) and even subroutines (change boundary conditions), via code updates sent from a laptop on the SC show floor
- summary data was streamed back to the laptop on the show floor, running real-time visualisation and sonification of the computation in progress
- updates of the code from the show floor were immediately reflected in the visualisation in real time, so that visitors to the infinicortex booth could see the simulation being "steered" in real-time

The interactive show floor demo was successful, but not without challenges. The InfiniCortex network was reliable and running the MPI codes over multiple locations (e.g. between Singapore and Australia) required no code changes. However, link latency was a significant challenge for this inherently data-parallel application. For this reason, best results were achieved by dividing the computations into sub-problems that were contained within a single site, and then handling result aggregation in an extra tier.
Thanks to the ability of on-demand, interactive access to high performance computing resources distributed around the globe, InfiniCloud2.0 provided a valuable testing environment for Extem-

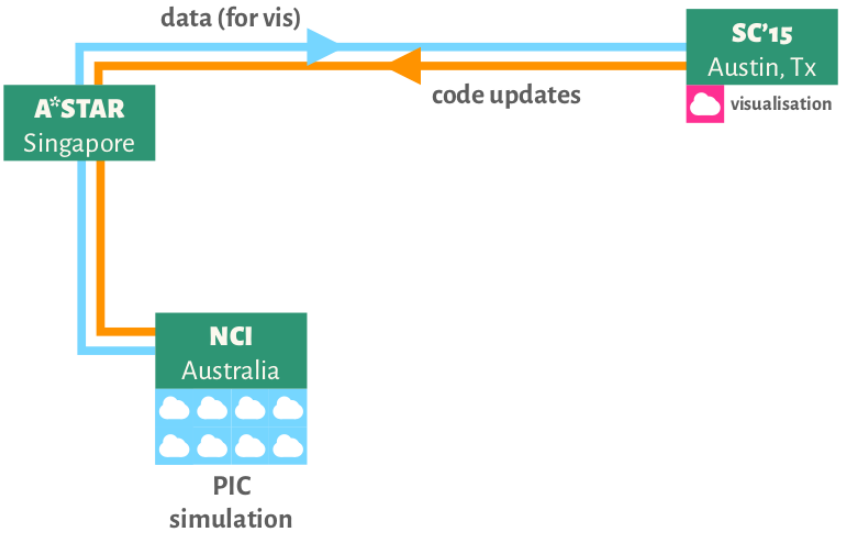

Figure 10. Extempore workflowh

pore, allowing scientific programmers to explore the response of the codes to different parametrisations and workloads.
Future optimization work could involve porting the simulation to reconfigure MPI running over IPoIB to native RDMA which is easier to tune appropriately to the characteristics of a long-distance link. The application could also be enhanced by adding topology and scheduling awareness, which would be very helpful in making sure that each component of the simulation runs in an optimal location, based on latency and bandwidth available. This could then become a foundation for a tiered model for Extempore, where problem sizes are devided into tightly coupled and loosely coupled parts which then can be efficiently scheduled into local and remote worker nodes.

Conclusions

In (Section 1) and (Section 2) we demonstrated the concept, design and implementation of a geographically distributed, High Performance Cloud system, capable of aggregating high performance computing resources available across four continents.
These resources can then be accessed through a uniform set of CLI, GUI and API interfaces, allowing users to create on-demand virtual supercomputers and storage systems, all connected with native InfiniBand.
This clearly demonstrates that it is indeed possible to fully aggregate capacity of a globally distributed pool of computational and storage resources. This is the first, small scale, implementation of a Galaxy of Supercomputers [14].
The InfiniCloud 2.0 platform has proved efficient and resilient. OpenStack components were able to seamlessly communicate over the IPoIB links presented by the InfiniCortex and apart from the image caching overhead, we did not observe any impact of the distance on Cloud operations. Centralized architecture allowed easy scaling to the growing number of InfiniCortex sites, without multiplying the management overhead. Such design enforces consistency through its simplicity - the central site provides a single source of truth.
In this paper, we run four distinct sites and we envision this number can grow to up to ten sites which would allow to provide a good global coverage in data transfer infrastructure for scientific

computations. Hence we are confident to state that our current design scales sufficiently for its purpose.

It is worth noting that this design made the Singapore site a single point of failure. While this is an acceptable risk for a prototype, semi-production environment, a full-production, large scale system would require a slightly different approach providing greater operational resiliency, however providing such solution is beyond the scope of this paper.

In (Section 3.1) we present BeeGFS storage optimized for high-latency, high-bandwidth links proved to be a very powerful tool, supplementing data processing toolkit with the ability to access data over long distance. This means that data collections no longer need to be copied for processing and can be accessed in place.

ElastiCluster and Biopipeline can very easily adapt to operate in a globally-distributed system. After some I/O optimizations, we were able to transparently distribute computational genomics jobs across four continents, aggregating the entire available capacity and creating a fully functional, global HPC cluster, realising the vision of a Galaxy of Supercomputers.

The distributed plasma physics simulation in Extempore demonstrated the ability of the Infinicloud2.0 network to support real-time bidirectional data streaming and real-time code hot-swapping. As dynamic cloud HPC contexts become more popular (e.g. [10] and [16]) the issues of on-demand interactivity and real-time feedback are active areas of research.

We believe a novel approach to High Performance Computing and Cloud Computing proposed in this paper can enable new ways of utilizing computational resources, joining resources available in multiple locations and providing new, more efficient ways of interacting with data collections.

*This work was supported by the A*STAR Computational Resource Centre through the use of its high performance computing facilities.*

This research was undertaken with the assistance of resources from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

The authors wish to thank Universite de Reims Champagne-Ardenne for providing the equipment essential for the experiments described in this paper.

The authors wish to thank Stony Brook University, New York for providing the equipment essential for the experiments described in this paper.

The authors wish to thank Fraunhofer Institute and in particular Bernd Lietzow, Sven Breuner, Frank Kautz and Christian Mohrbacher for their contribution and generous support for our BeeGFS work.

References

1. Amazon EC2 pricing. `http://aws.amazon.com/ec2/pricing/`, urldate = 2016-05-20.

2. BeeGFS parallel filesystem. `http://www.beegfs.com`.

3. ElastiCluster. https://github.com/gc3-uzh-ch/elasticluster.

4. Lustre File System, Operations Manual - Version 2.0. http://wiki.old.lustre.org/manual/LustreManual20_HTML/, urldate = 2016-05-20.

5. OpenStack Cloud Computing Platform. http://www.openstack.org.

6. PuppetForge OpenStack Puppet Modules. https://forge.puppet.com/puppetlabs/openstack.

7. Joseph Antony, Jakub Chrzeszczyk, Dongyang Li, Matthew Sanderson, Andrzej Chrzeszczyk, and Ben Evans. An Initial Microbenchmark Performance Study for Assessing the Suitability of Scientific Workloads Using Virtualized Resources from a Federated Australian Academic Cloud & EC2. Presented at HPC in Asia poster session at International Supercomputing Conference 2014, Leipzig, Germany.

8. Kenneth Ban, Jakub Chrzeszczyk, Andrew Howard, Dongyang Li, and Tin Wee Tan. InfiniCloud: Leveraging Global InfiniCortex Fabric and OpenStack Cloud for Borderless High Performance Computing of Genomic Data and Beyond. *Supercomputing Frontiers and Innovations 2015, Vol2.*

9. V. Decyk. Skeleton Particle-in-Cell Codes on Emerging Computer Architectures. *Computing in Science Engineering*, PP(99):47–52, 2015.

10. Marius Hillenbrand, Viktor Mauch, Jan Stoess, Konrad Miller, and Frank Bellosa. Virtual InfiniBand clusters for HPC clouds. In *Proceedings of the 2nd International Workshop on Cloud*. ACM, 2012.

11. Jonathan Low, Jakub Chrzeszczyk, Andrew Howard, and Andrzej Chrzeszczyk. Performance Assessment of Infiniband HPC Cloud Instances on Intel Haswell and Intel Sandy Bridge Architectures. *Supercomputing Frontiers and Innovations 2015, Vol2.*

12. Gabriel Noaje and Marek Michalewicz. Around The Globe Towards Exascale: InfiniCortex Past and Present. Presented at Supercomputing Frontiers Conference, Singapore, 2016.

13. Ben Swift, Andrew Sorensen, Henry Gardner, Peter Davis, and Viktor K. Decyk. Live Programming in Scientific Simulation. 2(4):4–15.

14. Tin Wee Tan, Dominic S.H. Chien, Yuefan Deng, Seng Lim, Sing-Wu Liou, Jonathan Low, Marek Michalewicz, Gabriel Noaje, Yves Poppe, and Geok Lian Tan. InfiniCortex: A path to reach Exascale concurrent supercomputing across the globe utilising trans-continental InfiniBand and Galaxy of Supercomputers. Supercomputing Frontiers Conference 2015, Singapore.

15. Calleja Paul Turek, Wojciech. Technical bulletin: High performance lustre filesystems using dell powervault md storage. http://i.dell.com/sites/content/business/solutions/hpcc/en/Documents/lustre-hpc-technical\%20bulletin-dell-cambridge-03022011.pdf.

16. Jerome Vienne, Wasi-ur Rahman, Nusrat Sharmin Islam, Hari Subramoni, and D.K. Panda. Performance Analysis and Evaluation of InfiniBand FDR and 40GigE RoCE on HPC and Cloud Computing Systems.