ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

High performance cloud computing

Viktor Mauch *,[1], Marcel Kunze [1], Marius Hillenbrand [2]

Karlsruhe Institute of Technology (KIT), Kaiserstrasse 12, 76131 Karlsruhe, Germany

ARTICLE INFO

Article history:
Received 15 December 2011
Received in revised form
5 March 2012
Accepted 9 March 2012
Available online xxxx

Keywords:
Cloud computing
HPC
IaaS
InfiniBand

ABSTRACT

Today's high performance computing systems are typically managed and operated by individual organizations in private. Computing demand is fluctuating, however, resulting in periods where dedicated resources are either underutilized or overloaded. A cloud-based Infrastructure-as-a-Service (IaaS) approach for high performance computing applications promises cost savings and more flexibility. In this model virtualized and elastic resources are utilized on-demand from large cloud computing service providers to construct virtual clusters exactly matching a customer's specific requirements.

This paper gives an overview on the current state of high performance cloud computing technology and we describe the underlying virtualization techniques and management methods. Furthermore, we present a novel approach to use high speed cluster interconnects like InfiniBand in a high performance cloud computing environment.

1. Introduction

High Performance Computing (HPC) is one of the leading edge disciplines in information technology with a wide range of demanding applications in economy, science, and engineering.

Due to specific requirements regarding performance or compliance it is common to commission and operate HPC resources in private. In this context, the execution environment for tasks is usually well-defined: It depends on specific libraries, applications, job schedulers and operating systems. Users of the services are thus limited to implement certain application scenarios. In such a context, modifications or extensions of the runtime or development environment can only be realized in cooperation with the system management and the corresponding operator teams. Furthermore, access to the HPC resources is very often restricted and jobs have to wait in a queue for execution. Conversely, it may happen that physical machines are under-utilized, depending on the varying demand within the particular institution.

A High Performance Cloud Computing (HPC2)[3] model might offer a solution that overcomes most of the aforementioned limitations. HPC2 applies the concept of cloud computing to contemporary HPC resources, resulting in an Infrastructure-as-a-Service (IaaS) delivery model [1,2]. The cloud computing approach promises increased flexibility and efficiency in terms of cost and energy consumption. Large providers with massive amounts of resources are able to gain an excellent economy of scale leading to affordable prices. In addition, public IaaS resources usually may on average be better utilized than private computing resources within institutes or companies.

Suitable management methods for elastic virtual clusters may provide the capacity and environment exactly matching actual demand and workload without the need to purchase and operate own hardware and software. The use of virtual machines (VM) allows users to securely gain administrative privileges within the guest operating system and customize the runtime environment according to their specific requirements. Due to the continuous availability and scalability of a cloud service, jobs no longer have to wait in a queue for hours or days (or even wait for months for the initial commissioning of a new cluster). Thereby, the wall clock time to get the job done may improve even if the execution speed of a virtual cluster environment is somewhat lower than that of a physical infrastructure.

In the following Section 2, we introduce the HPC2 model and discuss challenges and approaches concerning the realization. Thereby we focus on the integration of the InfiniBand interconnect technology which promises high network performance and a low latency, respectively. Furthermore, InfiniBand is highly prominent in the domain of HPC clusters. In Section 3 we compare current public HPC services to our private cloud approach based on InfiniBand. In this respect we will have a look at Amazon Web Services with its *cluster compute instance* offering. In comparison, we present a comprehensive architecture for an HPC2 model based on the three design goals:

* Corresponding author.
E-mail addresses: viktor.mauch@kit.edu, mauch@kit.edu (V. Mauch), marcel.kunze@kit.edu (M. Kunze), marius.hillenbrand@kit.edu (M. Hillenbrand).
[1] Steinbuch Centre for Computing, http://www.scc.kit.edu/.
[2] System Architecture Group, http://os.ibds.kit.edu/.

[3] High Performance Cloud Computing (HPC2) is a term originally coined by Robert L. Clay of Sandia National Labs. It refers to scalable runtime environments based on core notions from cloud computing applied to high performance machine architectures.

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.03.011

- *multi-tenancy*:
  Isolation enforcement of multiple work loads within the same physical computing and networking infrastructure.
- *dynamic provisioning*:
  Fully automated allocation, deployment and configuration of virtual clusters without any intervention of system administrators.
- *service level agreements*:
  Guaranteed network performance and quality of service (QoS) definition.

Besides an architectural description, we present workload measurements which illustrate the benefits of using a high speed interconnect as compared to 10 GE in virtualized environments. Section 4 deals with potential future prospects of HPC2 like live migration of virtual nodes and checkpointing. The paper concludes with a summary and an outlook in Section 5.

2. Challenges and benefits of an HPC2 model

Despite its promises, the HPC2 model faces several challenges: Very often the data and applications represent intellectual property that a company may not want to move to shared infrastructures. As a solution to this requirement, several cloud providers have started to offer special multi-tenant environments (so-called virtual private clouds) that are hosted by the provider but fully under control of a customer [3]. Some services offerings even go for shared-nothing resources to be exclusively used by a critical application [4].

However, the performance issue in a cloud context remains critical: In contrast to typical server workloads, HPC applications have a much higher demand for resource guarantees and timely delivery of results. It is hard to match these conditions within virtualized environments due to the higher I/O overhead and jitter of CPU cycles. In addition, HPC jobs are typically CPU-bound and often require synchronization of threads during execution. Therefore, both the placement strategy as well as the connectivity of clustered CPU resources are crucial to achieve adequate performance. Several essential requirements must be met to successfully implement an HPC2 model in a computing cluster:

- *Dynamic provisioning* requires automated deployment and managed allocation of virtual machines including the configuration of the cluster network.
- *Multi-tenancy* demands the isolation of each individual user or workload in the cluster interconnect network.
- *Quality of Service* with respect to network performance has to be guaranteed at all times, even when multiple users are sharing specific infrastructure parts at the same time.

It is difficult to meet all requirements in a virtualized environment at the same time and it is the task of a cloud service management to work out the best trade-offs suiting the specific application requirements.

Former research by Evangelinos and Hill [5] respectively Gupta and Milojicic [6] concerning the execution of HPC applications in contemporary non-HPC IaaS environments has identified network quality of service (QoS) as the primary hindrance to implement HPC2 clusters. Regola and Ducom [7] have found that providing virtual machines with a high-speed cluster interconnect such as InfiniBand [8] may improve performance significantly. Our studies and prototype evaluation show that multi-tenant virtual clusters based on the InfiniBand interconnect can be realized with considerable benefits concerning network latencies as compared to IaaS solutions based on Ethernet.

2.1. Dynamic provisioning

Cloud computing provisioning heavily relies on virtualization techniques, with the main goal to get an abstract view of physical resources retaining their interfaces. Smith and Nair presented an overview of system-level virtualization [9]. In fact, virtualization may offer several benefits for HPC, as has been pointed out in former research by Figueirede et al. [10], Huang et al. [11] and Mergen et al. [12]:

- Virtualization allows to use highly customized environments for HPC jobs, ranging from libraries down to the operating system. The concomitant isolation enables the simultaneous coexistence of different runtime environments on the same physical infrastructure.
- VMs allow to preserve binary-compatible runtime environments for legacy applications, without putting constraints on other applications or on the use of physical resources.
- Customized runtime environments are much easier to manage and faster to start and stop in a VM than on a physical host because hardware re-initialization is avoided.

Further advantages affect isolation, productivity and fault tolerance, amongst others:

- Isolation is ensured by the hypervisor in a protection layer below the guest OS. In that case users can be granted administrative privileges within their VMs, without breaking isolation between separate VMs or compromising host integrity.
- VMs implicitly form a resource principal that covers all the resource utilization inside the guest, explicitly including OS activity.
- Virtual system setups for testing and debugging of parallel applications can easily provide as many virtual nodes to detect scalability problems as required.
- In addition, virtualized resources can be debugged and monitored from outside the VM without integrating debugging code into the OS or application under test.
- Software reliability can be improved with checkpoint/restart schemes for VMs. At the same time, the hypervisor can shield VMs against (some) hardware failures on the system level and thereby reduce the need to implement fault tolerance mechanisms inside a guest OS.

2.1.1. Node virtualization

A virtual HPC2 system is usually instantiated as a collection of VMs grouped into virtual clusters by application of a placement strategy. The hypervisor on the physical nodes manages the allocation of the infrastructure resources such as CPU, memory, network, and disk storage. The most prominent hypervisors in the HPC environment are currently Xen [13] and KVM [14], compatible to Linux respectively the POSIX standard. Ferreira et al. [15] and Petrini et al. [16] found that the two alternatives, timesharing and memory overcommitment, would lead to preemption of HPC jobs, paging activity, and ultimately jitter in the execution speed of HPC workflows. For this reason usually a single physical CPU-core is mapped to a virtual core in order to minimize background activity resulting from the virtualization layer. In addition, it is best practice to match the complete virtual memory assigned to VMs with physical random access memory.

2.1.2. I/O virtualization

HPC systems are usually built around high performance network interconnects and it would be interesting to include these into the virtualized environment. The application of commodity

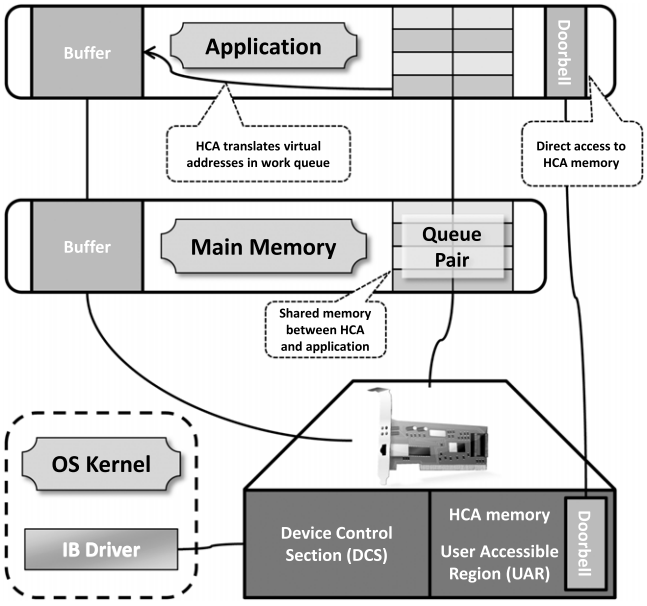

Fig. 1. OS-bypass access to an InfiniBand HCA. An application directly triggers send and receive operations. Transfer buffers are managed as shared memory between device and application (setup by OS to ensure isolation).

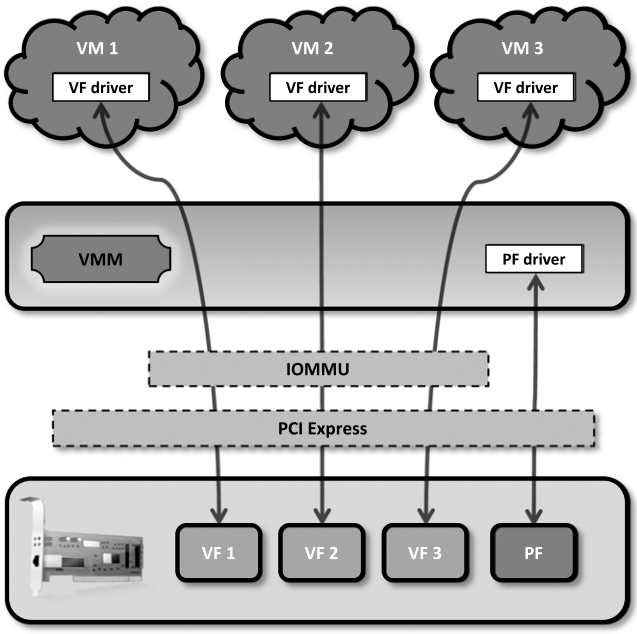

Fig. 2. Virtualized I/O device access with Single-Root I/O Virtualization (SR-IOV). The PCI device supports virtualization and presents itself as several *virtual functions*. Each VM gets access to one virtual function via PCI-passthrough, requiring an IOMMU.

network technologies like Ethernet would reduce performance as all network traffic has to flow through the protocol stack of the guest and host OS. Cluster interconnect technologies like InfiniBand [8] or Myrinet [17] rely on protocol offload and OS-bypass to achieve high bandwidth and low latency.

Compared to more traditional network technologies, such as Ethernet, InfiniBand gains a performance advantage by aggressive protocol offloading (up to transport layer handled completely in network adapters) and by directly interfacing applications to the network hardware. The OS is only involved in establishing connections and registering memory buffers with the InfiniBand adapter to ensure protection. Then, applications can bypass the OS and trigger actual communication operations and poll for their completion directly by accessing device memory, see Fig. 1. As a result, an application can handle complete send/receive cycles independently and without latency from intervention by the OS.

The following three virtualization concepts for I/O devices provide an acceptable performance within a virtualized environment:

- *PCI Passthrough*.
  This concept grants a VM direct access to a dedicated PCI I/O device. It requires an I/O Memory Mapping Unit (IOMMU) to ensure memory protection between different VMs, as described by Yassour et al. [18]. The number of VMs per host is restricted to the number of I/O devices built in, because each physical adapter card is used exclusively by a single OS. With PCI passthrough, a guest OS may use regular device drivers.
- *Para-Virtualization*.
  This virtualization technology provides a software interface that is similar but not identical to the actual hardware. The OS must be ported to be executed on the virtual machine. However, the porting simplifies the structure of the virtual machine and can achieve higher performance compared to a contemporary software virtualization approach.
  Liu et al. [19] have proposed VMM-bypass I/O, a para-virtualization approach for InfiniBand on Xen [13]. It does not need an IOMMU. However, this solution requires ongoing modifications of drivers (in host and guest) with respect to changes of the underlying hardware and OS. The available source code is outdated and difficult to adapt.
- *Single Root I/O Virtualization (SR-IOV)*.
  SR-IOV [20] is a new specification for virtualization support by hardware devices. It allows a PCI Express device to appear as multiple, separate devices, called *Virtual Functions (VF)*. Each guest may gain exclusive access to one VF by PCI passthrough. The *Physical Function* includes the SR-IOV capability. It is an anchor for creating VFs and reporting errors and events that cannot be attributed to a single VF. Fig. 2 provides an overview of the SR-IOV structure of an I/O PCI device.

Setting up I/O devices via PCI passthrough for VMs works independent of the actual device. The device access is restricted to a single VM (even the host OS is debarred). In contrast, para-virtualization would support multiple VMs, but it is highly device-dependent and requires to adapt and maintain drivers for different OSs and hypervisors. However, SR-IOV depends on manufacturer support to add appropriate functionality to the I/O device hardware and to provide specifications and/or drivers.

With respect to the InfiniBand interconnect, in all three cases an application can circumvent all layers of the system software, OS and hypervisor, when accessing the host channel adapter directly, just as in the native case. Hence, communication operations suffer no virtualization overhead in the command path between user-level application and InfiniBand interface. The SR-IOV concept currently provides the most sustainable virtualization solution for the InfiniBand interconnect technology. InfiniBand vendors such as QLogic [21] and Mellanox [22] have announced first officially supported solutions for productive environments to be available in 2012.

2.2. Multi-tenancy

Multi-tenancy is one of the essential attributes of Cloud Computing to concurrently support different customers and applications in a secure fashion, referring to access policies, application deployment, and data access and protection. The architecture of the cloud system has to be designed in such a way that it allows resources to be apportioned and shared efficiently between multiple tenants. A special emphasis is on networks

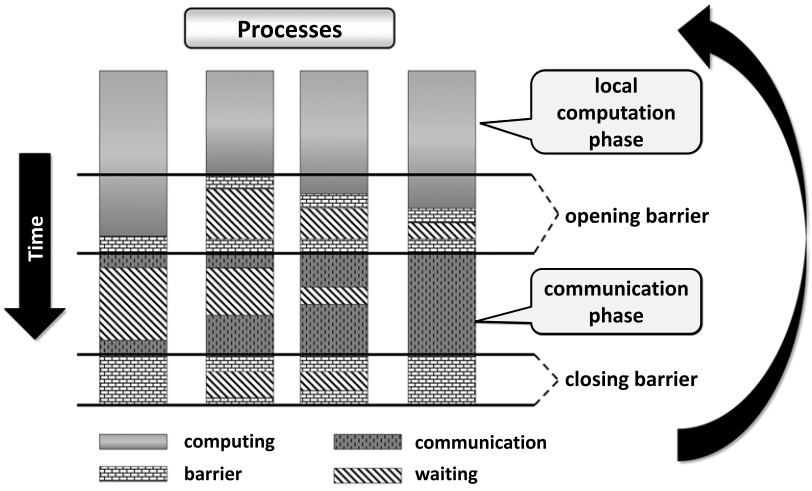

Fig. 3. SPMD Model. Tasks are split and run simultaneously on multiple processors for parallel execution. Each process alternates between computation and synchronization.

where in an Ethernet environment, usually a VLAN technique is employed in the switches and adapters to shield the traffic of each individual customer. High performance networks like InfiniBand on the other hand have specific challenges if they have to be implemented in a multi-tenant architecture.

2.2.1. Architecture

An InfiniBand network consists of a processor and I/O nodes connected through a fabric made up of cascaded switches and routers. I/O units can range in complexity from a single device to a large, memory-rich *redundant array of independent disks (RAID)* subsystem. The network can be subdivided into *subnets* interconnected by routers. A *subnet* consists of end nodes and switches and forms an administrative domain for InfiniBand network management. All transmissions begin or end at a *channel adapter (CA)*. Each processor contains a *host channel adapter (HCA)* and each peripheral has a *target channel adapter (TCA)*. CAs appear as nodes to the fabric and provide one or more ports to which the data is sent.

Send requests and receive buffers are posted to pairs of work queues for send and receive. These so-called *queue pairs (QPs)* form communication endpoints. Each of the ports may include multiple such QPs. From the point of view of data transfer the QP number is part of the address.

InfiniBand supports four transport types: connection- and datagram-oriented, each in a reliable and an unreliable variant. All connection-oriented as well as the reliable datagram transport types form an association between two *queue pairs*, before data transfer can occur. In addition to send and receive primitives, the specific InfiniBand transport protocol supports *remote DMA (RDMA)* operations, thereby providing asynchronous remote shared memory semantics.

2.2.2. Addressing

The InfiniBand architecture distinguishes two scopes for addressing, and thus two types of addresses. *Global identifiers (GID)*, using the format of IPv6 [23] addresses, are used by layer-3 routers at a potentially global scale, while *local identifiers (LID)*, comprising of 16 bits, are used by layer-2 switches. LIDs and GIDs are assigned dynamically by the subnet management software. Each InfiniBand adapter has a static, manufacturer-assigned unique identifier, called a *globally unique ID (GUID)*, that serves to identify nodes independent of the network configuration.

2.2.3. Management

The InfiniBand architecture defines uniform management interfaces and protocols that are used to configure, for example, the isolation based on partitioning and QoS mechanisms. These protocols utilize a special type of network packet, called *management datagrams (MAD)*. Such packets have to be sent from special queue pairs (numbered 0 and 1) to be considered valid. Using these special queue pairs is a privileged operation typically reserved to the (host) OS. All InfiniBand devices (e.g., HCAs, switches, and routers) have to support a basic set of management protocols. The most important one is the *subnet management protocol*, which a *subnet manager* uses to assign node addresses, amongst others. As a consequence, all advanced configuration options of InfiniBand are available using a standardized interface, independent from the hardware actually used. In contrast, advanced mechanisms of Ethernet, such as VLAN support, have to be configured with vendor-specific protocols.

2.3. Quality of service

It is a non-trivial task to construct a virtual cluster on the basis of virtual machines and achieve a well-defined quality of service. Many HPC applications match the bulk-synchronous single-program multiple-data (SPMD) application model, described by Dusseau et al. [24] and Jones et al. [25], and can be seen simplified as cycling through two phases (see Fig. 3): Purely local computation alternates with synchronization and communication. A parallel job consists of several processes (a static number) that run concurrently, each on a separate processor, and execute the same binary program. Processes exchange data and synchronize via message passing. A popular, standardized framework for executing such parallel computing tasks is the message passing interface (MPI) [26]. A lot of HPC applications are using MPI and heavily rely on synchronous and fast communication. Latencies and jitter that are introduced by the virtualization layer may thus hamper operation at a good and continuous performance.

2.3.1. Partitioning and traffic-shaping

The InfiniBand architecture provides mechanisms to configure partitions and shape the network traffic. Groups of nodes called *partitions* are formed by isolation mechanisms with restricted communication relations. Isolation is implemented by the InfiniBand transport layer protocol with partition identifiers in

each packet.[4] Management software configures each node to be a member of one or multiple partitions by storing a *membership table* in each node.[5] Each adapter accepts or drops packets based on that table. Switches can filter packets based on partition identifiers, for each switch port separately, and thereby allow to enforce isolation.

The InfiniBand specifications define a flexible QoS mechanism that is implemented in each outgoing network port (i.e., in adapters as well as switches). QoS mechanisms allow scheduling network bandwidth in a flexible way by differentiating between up to 15 traffic classes, called *service levels*. In each port, packets to be sent are sorted into up to 15 send queues, so-called *virtual lanes*, based on their traffic class. The outgoing link demultiplexes the send queues based on a configurable weighted round-robin schedule.[6] The scheduling policy can be configured at two stages:

1. Packets are sorted into a send queue based on their traffic class and a configurable table. Instead of queuing packets, they can be filtered based on their traffic class.
2. The outgoing link is multiplexed between the send queues based on a configurable weighted round-robin schedule.

2.3.2. Jitter

Operating system processes and handling of asynchronous events constitute performance noise or jitter perceived by an application. Various studies demonstrate the effects of jitter for HPC applications [27,28]. In order to avoid jitter it is common to deploy specialized lightweight operating systems on the compute nodes. This idea could be followed as well for the hypervisors and it would be interesting to deploy Linux systems with lightweight kernels such as Kitten or Palacios. Vendors of commercial virtualization products also work along this line: An example is VMware vCloud where very lean virtualization kernels are implemented on the virtualization nodes that are managed by the central vCenter console process. However, at the time being there are only few studies about OS jitter and its impact on the performance in large virtualized HPC clusters.

3. Architecture and implementation

At the time being there are only a few public cloud services that follow the HPC2 model. These offerings are however all based on Ethernet interconnects. For this reason we have implemented a small testbed to study the functional properties of virtual clusters using a fast InfiniBand network. In the following we compare the properties of public HPC2 services with the private InfiniBand based solution.

3.1. Public cloud

Public cloud providers offer a multitude of services at different performance levels that match different application requirements. The Amazon Web Services for instance currently comprise twelve instance types with variable amount of memory, storage and networking capacity. The relative processing performance is given in Elastic Compute Units (ECU) where one ECU corresponds roughly to the equivalent CPU capacity of a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. The smallest machine can be instantiated with 1.7 GB RAM and Gbit Ethernet and has a single virtual core with a compute power of 1 ECU; the largest machine offers 60 GB RAM, 10 Gbit Ethernet and 88 ECUs, based on the actual dual eight-core Intel Xeon CPU systems. The largest instances are *cluster compute instances (CCI)* that are specifically useful to support HPC applications. In contrast to the smaller instances they are using the hardware virtualization technology that is built into modern CPUs. Furthermore, they are equipped with a high performance Ethernet networking interface that is directly supported by the virtual machine hypervisor. Amazon offers a placement strategy that allows grouping nodes in physical neighborhood and combining them into clusters with low communication latency. Tools like StarHPC [29] or MIT StarCluster [30] provide sophisticated mechanisms for cluster setup on top of these offerings.

Table 1
Single node performance of AWS instance types.

AWS type	ECU/Core/RAM	N/NB/P/Q	GFlops
m1.large	4/2/7.5	4k/128/1/2	9.1
m1.xlarge	8/4/15	8k/128/2/2	27.9
m2.2Xlarge	13/4/34.2	16k/128/2/2	37.2
cc1.4Xlarge	33.5/8/23	16k/128/2/4	75.9
cc1.8Xlarge	88/16/60.5	16k/128/4/4	227.8

In order to get an impression of the single node virtual machine performance, the High Performance Linpack (HPL) [31] benchmark has been deployed on various virtual machine instances. The tests have been performed using the Intel C++ compiler, the Intel MKL library, and the Intel MPI implementation. The results look very promising and are given in Table 1. It seems that the performance penalty introduced by the virtualization layer is quite bearable in single node operation. When the nodes are combined into virtual clusters, additional communication latencies might play a role.

A self-made virtual cluster of 17.024 AWS CCI cores entered the TOP500 list of the world's fastest supercomputers [32] in November 2011 at rank 42 with a HPL performance of 240.09 TFlops [33], reaching 68% of its theoretical peak performance of 340.10 TFlops. Other physical clusters in the list that are equipped with 10 Gbit Ethernet interfaces are reaching around 70% of peak performance, whereas clusters with standard Gbit Ethernet are at 50% only. This indicates that the AWS cluster is doing quite well as compared to its physical competitors despite the fact that it is based on virtualization technology. It is interesting to see that clusters of similar performance at neighboring entries to the AWS machine based on physical nodes with the much faster InfiniBand QDR interconnect are yielding 75% and 85% of peak performance, respectively. As the single virtual node performance of virtual clusters looks quite reasonable and also the virtual networking interfaces seem to work quite well this indicates that by use of a faster network interconnect the efficiency of virtual clusters might be significantly increased.

3.2. Private cloud

Former studies concerning the performance figures of HPC applications in compute clouds by Iosup et al. [34], Juve et al. [35] and Montero et al. [36] are concentrating on Ethernet based cloud computing infrastructures. We believe that an cloud computing IaaS solution based on the InfiniBand interconnect could provide an added value for HPC customers. However, in contrast to Ethernet the InfiniBand technology is much more complex and is not directly supported by current hypervisors. This might explain why at the time being public cloud providers do not offer InfiniBand as an alternative interconnect technology. We have set up a testbed to study the use of InfiniBand in virtual clusters with

[4] Partition identifiers comprise 16 bits and allow to distinguish 65 535 partitions. The identifier *0xff* is reserved and means *any partition*.

[5] Full access of an InfiniBand adapter allows to modify the local partition membership table. However, switch partition filtering will prevent eavesdropping from foreign partitions.

[6] Two round-robin schedules are used, one prioritized strictly higher than the other (with possible exceptions that we omit for brevity).

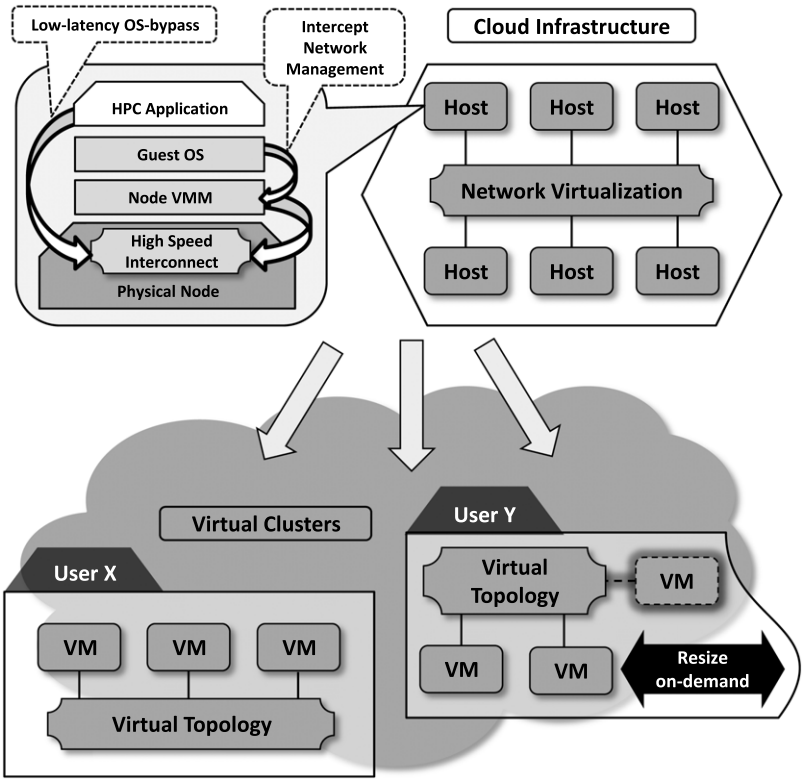

Fig. 4. The virtualization of host and the cluster interconnect is controlled by a cloud framework to provide elastic virtual clusters.

the long-term goal to provide self-service elastic virtual compute clusters for HPC workloads.

In contrast to physical clusters, we provide *virtual clusters* that support dynamic scaling according to customer demands. The architecture comprises three aspects: Each node of a physical cluster is virtualized, the cluster interconnect is virtualized, and we orchestrate node and network virtualization with an HPC cloud management framework. Fig. 4 illustrates our basic architecture. A detailed description of the underlying architectural approach has been worked out by Hillenbrand [37].

We propose to employ GNU/Linux and the Kernel-based Virtual Machine (KVM) hypervisor in combination with the existing InfiniBand driver stack to set up a virtualization fundament for our HPC cloud architecture. Currently, we turn our attention to the InfiniBand technology, however we expect that our findings and concepts are transferable to other cluster interconnects or future Ethernet generations.

The underlying infrastructure to operate our prototype HPC cloud is a 4-node cluster with an InfiniBand interconnect. Computing nodes of type *HP ProLiant BL460c G6 Blade Server* are in use within a *HP BladeSystem c7000 Enclousure*. They are each equipped with an Intel Xeon E5520 quadcore CPU (with Hyper-Threading, clocked at 2.26 GHz) and 6 GB of DDR2 DRAM. The test cluster InfiniBand interconnect comprises Mellanox ConnectX-2 InfiniBand HCAs built into each server, and an InfiniBand DDR switch integrated within the blade chassis. The physical links provide a bandwidth of 20 Gbit/s limited by the switch. We use an additional Gbit Ethernet connection for remote management access via SSH. Since SR-IOV InfiniBand drivers are not yet officially available, we revert to using PCI passthrough and dedicate each InfiniBand HCA to a single VM.

3.2.1. HPC cloud management prototype

As a proof of concept, we extended the cloud management framework OpenNebula with new abstraction layers and control

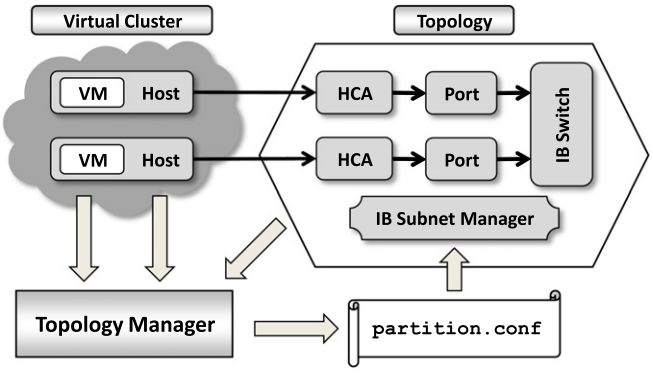

Fig. 5. Allocation workflow of virtual clusters in the extended OpenNebula framework.

mechanisms. They allow to group virtual machines into clusters and automatically derive the partition configuration for the *opensm* subnet manager to establish network isolation within the InfiniBand fabric. The modified deployment procedure is illustrated in Fig. 5.

The simple deployment policy of virtual machines in OpenNebula [38] was adapted and complemented with a more sophisticated logic with respect to the underlying InfiniBand interconnect. At startup, the OpenNebula core queries all hosts for specific information about their InfiniBand HCAs. The abstract topology is determined by the globally unique identifier (GUID) of the HCAs. The topology manager merges the associated VMs to virtual clusters and sets up the corresponding partitions within the InfiniBand fabric. The enforcement of the network isolation is realized by a cyclic re-configuration of the *opensm* subnet manager. Because of the current operation mode which allows only one single virtual machine with direct access to a dedicated HCA, the deployment process does not have to manage resource sharing of physical hosts

amongst different tenants. An operative SR-IOV configuration will become more sophisticated in the future.

Although OpenNebula does not support device assignment via PCI pass-through, it is possible to include raw configuration statements for the hypervisor management in a VM definition file. Such statements are not interpreted by OpenNebula, but are just passed on to the libvirt API unmodified. In addition to the InfiniBand interconnect access, the CPU affinity for VMs is also managed in the same way. Each logical core is mapped to a separate physical core to enforce dedicated resource allocation which is the preferred mode to support HPC tasks.

3.2.2. Network isolation

In order to guarantee network isolation for customers, the HPC cloud provider has to keep the configuration of the physical network switches under exclusive control. Therefore, everyone besides the cloud provider must be locked out of the InfiniBand management interfaces. In our architecture of an HPC cloud, we employ the protection key mechanism[7] to protect the network configuration in case a user accidentally or forcefully gained access to a HCA, and therewith had the possibility to send management datagrams.

As a result, we achieve a protection of the management interface and thereby also hide the physical network topology. As long as the protection keys are kept secret, we can be sure that:

- Communication is restricted to within a virtual cluster only.
- No additional subnet manager operated by users can modify the network configuration.
- The topology of the whole physical network cannot be determined (e.g., using the OFED utility *ibnetdiscover*).
- Protections keys and isolation settings (partition keys) cannot be readout, because the get requests will be discarded without the correct key.

Practical experiments have verified that these properties within virtual environments with full access to the InfiniBand HCA are useful. The upcoming SR-IOV implementation for the InfiniBand interconnect promises even more configurability concerning restriction policies.

3.2.3. HPC workload in virtualized environments

SKaMPI [39] is a synthetic MPI benchmark that we used to examine the functional properties of our test setup. The benchmark sends and receives data packets of varying size between the cluster nodes and measures the communication latency. All measurements have been performed between two VMs for different communication setups.

With regard to our prototype, we compare the native InfiniBand DDR (20 Gbit/s) network with the introduced virtual cluster environment based on virtualized InfiniBand by using PCI passthrough. Furthermore we are including the corresponding figures for Amazon EC2 CCI of type *cc1.4XLarge* that is based on 10 Gbit Ethernet. To compare the EC2 CCI communication latency with our prototype network, we restrict the corresponding virtualized InfiniBand link speed to 10 Gbit/s for an additional measurement series. The results are given in Table 2. It turns out that the latency increases only slightly by introduction of the virtual layer, and the performance penalty is constantly in the order of 0.1–0.2 μs only, independent of the message size. It is obvious that the communication time of Ethernet as compared to Infiniband lasts an order of magnitude longer.

Table 2
Measured communication latencies in μs.

Message size (B)	EC2 CCI ETH 10 Gb	Virt. IB 10 Gb	Virt. IB 20 Gb	Native IB 20 Gb
4	77.5	3.4	2.2	2.0
16	77.1	3.5	2.3	2.1
64	78.3	4.7	2.5	2.3
256	80.1	4.1	3.8	3.6
1024	82.3	5.6	4.8	4.6
4096	103.8	9.9	7.7	7.6

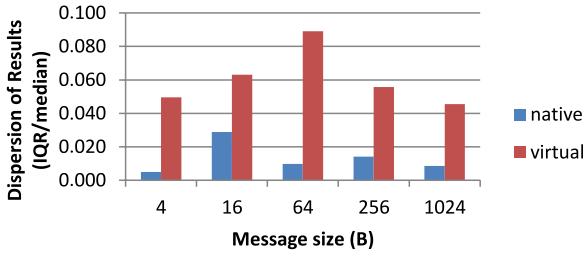

Fig. 6. Jitter vs. message size for InfiniBand (native and virtual).

The main causes for this difference are the overheads introduced by network interconnects and software interfaces. In an EC2 CCI, data transfers have to pass several layers, starting from the user applications through the TCP/IP stack of the guest OS, via a para-virtualized Ethernet device through the hypervisor, and finally through the OS in the Xen driver domain to the physical Ethernet network interface adapter. In contrast, communication over InfiniBand uses OS-bypass access to the HCA (see Section 2.1.2) and thereby circumvents all layers of system software for communication operations.

Communication-bound parallel applications require low latency for an acceptable scalability. For such jobs, an HPC2 model based on a dedicated cluster interconnect like InfiniBand provides a clearly better network performance compared to a solution using Ethernet.

We made, however, the observation that the virtualization layer obviously introduces an additional source of OS noise, as is indicated by the measurements of the dispersion of the results (see Fig. 6). It seems that there is a need to work on the optimization of the hypervisor in order to reduce jitter.

4. Future perspectives

The concept of virtualization plays a key role for the implementation of the HPC2 model. Besides the construction of virtual on-demand clusters there are further benefits that could be expected, such as live migration and checkpointing.

4.1. Live migration

Live migration is a concept that allows to shift virtual machines between physical hosts during operation, described by Clark et al. [40]. The advantage is that maintenance work becomes possible without interruption of running processes and programs. The live migration feature could be very interesting for large HPC systems with thousands of nodes as it would bring high availability and also gives a new degree of freedom for load-balancing. In the ideal case, the migration is transparent to the migrated VM as well as to all peers that communicate with the migrated VM.

[7] In this context, we encountered a malfunction of the current *opensm* subnet manager implementation (version 3.3.12). We had to modify the source code to actually enable the protection mechanism, because the delivered version failed to do so. Further, *opensm* currently supports only one protection key for all nodes. This policy allows a malicious user who is able to extract the protection key of his HCA to immediately get access to the configuration of all other nodes. An assignment of unique keys would circumvent this attack strategy.

Please cite this article in press as: V. Mauch, et al., High performance cloud computing, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.03.011

Whereas live migration is commonly found in virtual infrastructures based on Ethernet, there is currently no way known to transparently migrate virtual machines with active connections over InfiniBand. With current InfiniBand HCAs protocol state of open connections is part of the internal state of an HCA and cannot simply be transferred to other adapters together with the VM. Migration of VMs which share a single HCA is even more difficult, because an InfiniBand node address is associated with a (physical) port instead of with a VM. Thus, a VM will have a different node address before and after migration. These node addresses are also used at the InfiniBand user interface, so user-level software has to be modified to handle node addresses changing during migration.

Live migration of VMs with active InfiniBand connections is possible, when the migration process involves user-level applications. Huang has proposed the *nomad* framework in his Ph.D. Thesis [41], an extension to the InfiniBand para-virtualization approach in [19], to accomplish live migration of VMs. The *nomad* framework is based on the fact that most HPC applications do not use InfiniBand directly, but employ a communication library such as MPI. This approach conceals migration from HPC applications by modifying the MPI library to close all connections before and re-establish them after migration. Thus, this technique depends on software modifications inside the VM and cannot be considered transparent. Therefore, transparent live migration of VMs with InfiniBand access is a worthwhile topic for future work.

In contrast, transparent live migration works with TCP/IP over Ethernet, because connection state is kept in the guest OS's protocol stack and thereby is migrated with the memory image of the VM. Usually, each VM has its own distinct MAC address and packet routing automatically adapts to a VM's new location in the network topology after migration. Even when MAC addresses are assigned to host systems and therefore not migrated with a VM, the indirection of using IP addresses in applications avoids changes in user-space.

4.2. Checkpointing

Checkpointing of HPC jobs is an important topic as it leads to a better fault-tolerance. This is especially interesting for actual HPC systems that comprise 10,000 s or even 100,000 s of processing cores. At such large counts of cores, faults are becoming commonplace and checkpointing could lead to more robust operation.

Virtualization technology does enable implicit system-level fault tolerance without modifying existing OS or applications and allows to the record virtual machine memory and an I/O state that may be stored as a snapshot on disk. Thus the state of a complete HPC application could be conserved at any time for later restarts, independent of the actual MPI activity. This could even be accomplished on a different physical infrastructure at an alternate location, thus enabling "beaming" of virtual clusters from one place to another.

Regarding the state-of-the-art, it seems feasible to conserve the machine state of Ethernet based clusters with shared network disks. However, there is currently no solution to conserve the state of the switch and HCAs in InfiniBand based clusters together with the state of the virtual machines. In addition there is currently no method available to conserve the state of parallel file systems.

4.3. Spot market

Cloud providers like Amazon start to establish spot markets on which they sell excess capacity. Getting HPC resources from the spot market is pretty cheap most of the time and works marvelous for development, testing and smaller campaigns. However, there is no guarantee for continuous operation: If the market price exceeds the maximum bid, a virtual machine is stopped. As only a few HPC applications have been designed to work under such conditions, system-level checkpointing could be of importance to generally overcome this difficulty.

On the other hand there are specific market services that allow that customers and clients meet to make a special deal (e.g. SpotCloud market place [42]). SpotCloud has implemented an environment to buy and sell computing capacity globally based on price, location, and quality on a fast and secure platform. Here any datacenter can offer spare resources and sell them for use in a virtual environment. At the same time customers can get very affordable computing power that is sometimes even cheaper than the energy cost to operate the same resources at home. While this currently only works for individual virtual machines, an extension to the HPC2 model seems to be a quite natural step in the near future.

5. Summary

High Performance Cloud Computing (HPC2) is a model that offers a lot of interesting perspectives. The provisioning of elastic virtual clusters as on-demand pay-as-you-go resources avoids initial cost for physically owned hardware and allows great flexibility and scalability for customers and their applications.

Current cloud computing offerings with virtual machines are based on communication via Ethernet. InfiniBand as an interconnect technology would be a better choice due to its superior performance. We present a novel architecture for HPC IaaS clouds supporting InfiniBand that allows an individual network configuration with QoS mechanisms. Customers are able to allocate virtual clusters which are isolated from other tenants in the physical network.

Future plans foresee the provisioning of InfiniBand access within virtual machines based on SR-IOV, that is the most promising and sustainable virtualization technology for high-speed interconnects. The first officially supported SR-IOV firmware releases and drivers will soon be available

Currently, live migration of virtual machines in an InfiniBand based configuration is still an unsolved problem because of the machine specific protocol offloading in the adapter hardware. An additional challenge is the efficient usage of the limited QoS resources which may restrict the maximum number of isolated virtual clusters in the same physical context. In this case, sophisticated scheduling algorithms could help to improve the situation.

References

[1] C. Baun, M. Kunze, T. Kurze, V. Mauch, High performance computing as a service, in: I. Foster, W. Gentzsch, L. Grandinetti, G.R. Joubert (Eds.), High Performance Computing: From Grids and Clouds to Exascale, IOS Press, 2011.
[2] C. Baun, M. Kunze, J. Nimis, S. Tai, Cloud Computing—Web-Based Dynamic IT Services, Springer, 2011.
[3] Amazon Virtual Private Cloud. http://aws.amazon.com/vpc/.
[4] Vcodyne cluster on demand. http://www.vcodyne.com/.
[5] C. Evangelinos, C. Hill, Cloud computing for parallel scientific hpc applications: feasibility of running coupled atmosphere–ocean climate models on Amazonís EC2., ratio 2 (2.40) (2008) 2–34.
[6] A. Gupta, D. Milojicic, Evaluation of HPC applications on cloud, Tech. Rep. HPL-2011-132, HP Laboratories, 2011.
[7] N. Regola, J.-C. Ducom, Recommendations for virtualization technologies in high performance computing, in: Second International Conference on Cloud Computing Technology and Science, IEEE, 2010.
[8] InfiniBand Architecture specification volume 1, Release 1.2.1, InfiniBand Trade Association, 2007.
[9] J.E. Smith, R. Nair, The architecture of virtual machines, IEEE Computer 38 (5) (2005) 32–38.
[10] R. Figueiredo, P. Dinda, J. Fortes, A case for grid computing on virtual machines, in: Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on, IEEE, 2003, pp. 550–559.
[11] W. Huang, J. Liu, B. Abali, D. Panda, A case for high performance computing with virtual machines, in: Proceedings of the 20th Annual International Conference on Supercomputing, ACM, 2006, pp. 125–134.

Please cite this article in press as: V. Mauch, et al., High performance cloud computing, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.03.011

[12] M. Mergen, V. Uhlig, O. Krieger, J. Xenidis, Virtualization for high-performance computing, ACM SIGOPS Operating Systems Review 40 (2) (2006) 8–11.
[13] P. Barham, et al., Xen and the art of virtualization, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ACM, 2003, pp. 164–177.
[14] Kernel-based virtual machine. http://www.linux-kvm.org/.
[15] K. Ferreira, P. Bridges, R. Brightwell, Characterizing application sensitivity to os interference using kernel-level noise injection, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE Press, 2008, p. 19.
[16] F. Petrini, D.J. Kerbyson, S. Pakin, The case of the missing supercomputer performance: achieving optimal performance on the 8192 processors of ASCI Q, in: ACM/IEEE SC2003 Conference on High Performance Networking and Computing, 2003, p. 55.
[17] Myrinet. http://www.myricom.com/scs/myrinet/overview/.
[18] B.-A. Yassour, M. Ben-Yehuda, O. Wasserman, Direct device assignment for untrusted fully-virtualized virtual machines, Tech. Rep., IBM Research, 2008.
[19] J. Liu, W. Huang, B. Abali, D. Panda, High performance VMM-bypass I/O in virtual machines, in: Proceedings of the Annual Conference on USENIX, vol. 6, 2006.
[20] PCI-SIG single-root I/O virtualization specification. http://www.pcisig.com/specifications/iov/single_root/.
[21] QLogic. http://www.qlogic.com/.
[22] Mellanox Technologies. http://www.mellanox.com/.
[23] S. Deering, Internet protocol, version 6 (ipv6) specification.
[24] A. Dusseau, R. Arpaci, D. Culler, Effective distributed scheduling of parallel workloads, in: Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, ACM, 1996, pp. 25–36.
[25] T. Jones, S. Dawson, R. Neely, W. Tuel, L. Brenner, J. Fier, R. Blackmore, P. Caffrey, B. Maskell, P. Tomlinson, et al. Improving the scalability of parallel jobs by adding parallel awareness to the operating system.
[26] Message Passing Interface Forum. MPI: a message-passing interface standard, 1994.
[27] B. Tsafrir, Y. Etsion, D.G. Feitelson, S. Kirkpatrick, System noise, os clock ticks, and fine-grained parallel applications, in: Proceedings of the 19th ACM International Conference on Supercomputing, ACM, 2005.
[28] B. Beckman, K. Iskra, K. Yoshii, S. Coghlan, The influence of operating systems on the performance of collective operations at extreme scale, in: Proceedings of the IEEE International Conference on Cluster Computing, 2006.
[29] C. Ivica, J.T. Riley, C. Shubert, Starhpc—teaching parallel programming within elastic compute cloud, in: ITI, 2009, pp. 353–356.
[30] StarCluster. http://web.mit.edu/star/cluster/.
[31] J. Dongarra, P. Luszczek, A. Petitet, The linpack benchmark: past, present and future, Concurrency and Computation: Practice and Experience 15 (9) (2003) 803–820.
[32] H. Meuer, The top500 project: looking back over 15 years of supercomputing experience, Informatik-Spektrum 31 (3) (2008) 203–222.
[33] Top500 (11/2011)—Amazon EC2 cluster compute instances. http://www.top500.org/system/177457 (accessed 15 November 2011).
[34] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, IEEE Transactions on Parallel and Distributed Systems 22 (6) (2011) 931–945.
[35] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. Berman, P. Maechling, Scientific workflow applications on Amazon EC2, in: E-Science Workshops, 2009 5th IEEE International Conference on, IEEE, 2009, pp. 59–66.
[36] R.S. Montero, R. Moreno-Vozmediano, I.M. Llorente, An elasticity model for high throughput computing clusters, Journal of Parallel and Distributed Computing 71 (6) (2011) 750–757.
[37] M. Hillenbrand, Towards virtual infiniband clusters with network and performance isolation, June 2011. http://os.ibds.kit.edu/.
[38] OpenNebula—The open source toolkit for data center virtualization. http://opennebula.org/.
[39] R. Reussner, et al., Skampi: a detailed, accurate MPI benchmark, in: V. Alexandrov, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer, 1999.
[40] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation—Volume 2, USENIX Association, 2005, pp. 273–286.
[41] Wei Huang, High performance network I/O in virtual machines over modern interconnects, Ph.D. Thesis, Ohio State University, 2008.
[42] SpotCloud market place. http://spotcloud.com/.

Viktor Mauch is a Ph.D. student at the Steinbuch Centre for Computing (SCC) of the Karlsruhe Institute of Technology (KIT). His research interests include cloud computing and high performance computing with focus on the InfiniBand interconnect. He received his Diploma degree in Physics from the KIT in 2008.

Marcel Kunze leads the Research Group Cloud Computing (RG-CC) in the Steinbuch Centre for Computing (SCC) and a cloud computing research laboratory in the Engineering Mathematics and Computing Lab (EMCL) at the Karlsruhe Institute of Technology (KIT).

Marius Hillenbrand is a Ph.D. student at the System Architecture Group of the Karlsruhe Institute of Technology (KIT). His research interests include virtualization and the operating system aspects of cloud computing on HPC infrastructure with low-latency networks, currently with a focus on InfiniBand clusters. He received his Diploma degree in Computer Science from the KIT in 2011.

Please cite this article in press as: V. Mauch, et al., High performance cloud computing, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.03.011