ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 4, No. 2, XX-YY (2019)
www.astesj.com

ASTES Journal
ISSN: 2415-6698

The Chimera and the Cyborg

Hybrid Compute: In vivo HPC, Cloud and Container Implementations

Lev Lafayette[*,1], Bernd Wiebelt[2], Dirk von Suchdoletz[2],
Helena Rasche[3], Michael Janczyk[2], Daniel Tosello[1]

[1]*University of Melbourne, Department of Infrastructure, Melbourne, Australia*

[2]*University of Freiburg, High Performance Computing, Freiburg, Germany*

[3]*University of Freiburg, de.NBI, Freiburg, Germany*

ARTICLEINFO

Article history:
Received: 30-Jan-2019
Accepted: 03-Feb-2019
Online:

Keywords:
Hybrid HPC/Cloud
High Performance Computing
Cloud Computing
High Throughput Computing
Containers

ABSTRACT

High Performance Computing (HPC) systems offer excellent metrics for speed and efficiency when using bare metal hardware, a high speed interconnect, and massively parallel applications. However, this leaves out a significant portion of scientific computational tasks, namely high throughput computing tasks that can be trivially parallelized and scientific workflows that require their own well defined software environments. Cloud computing provides such management and implementation flexibility at the expense of a tolerable fraction of performance. We show two approaches to make HPC resources available in a dynamically reconfigurable hybrid HPC/Cloud architecture. Both can be achieved with few modifications to existing HPC/Cloud environments. The first approach, from the University of Melbourne, generates a consistent compute node operating system image with variation in the virtual hardware specification. The second approach, from the University of Freiburg, deploys a cloud-client on the HPC compute nodes, so the HPC hardware can run Cloud-Workloads using the scheduling and accounting facilities of the HPC system. Extensive use of these production systems provide evidence of the validity of either approach.

1 Motivation

Modern research institutions increasingly employ digital methods and workflows, which require a corresponding increase in the amount of computational resources. As data sets grow the researchers' own machines or large workstations no longer suffice, and they must turn to other resources such as High Performance Compute (HPC) and Cloud resources. Procurement, installation, and operation of these compute resources are demanding tasks which cannot be handled by individual researchers and work groups any more. Centralization of resources and administration can leverage economies of scale, but comes with compromises regarding hardware and software configurations that can cater to the needs of a growing user base. Currently neither HPC nor Cloud computing in isolation can mesh the the demands of users with the needs and resources of the compute centers.

University centers must find new, efficient ways to cater to user desires of tailored infrastructures that meets their computational needs. Any new solution should provide comparable offerings regarding features and pricing while avoiding overstretching existing personnel resources. Often demands for hardware arise with short notice and for project durations well below the normal cost amortization period of five to six years. The typical challenges of university computer centers are rooted in the very diversity of their scientific communities; wide ranging requirements for software, services, workflows, and compute resources.

Virtualization is a key technology from two fronts; it can permit accommodating diverse user requirements with a largely centralized resource, and it can help to isolate the different software environments and hardware configurations. As many resources in research infrastructures are inconsistently utilized, tapping into cloud strategies can help to significantly save on energy, personnel investment, and hardware resources.

[*]Lev Lafayette, University of Melbourne, lev.lafayette@unimelb.edu.au

This paper explores the possibility to overcome the dichotomy of cloud and HPC with a single cohesive system, and how such a setup can provide the performance of an HPC system as well as the flexibility of a cloud compute environment. Furthermore, the paper discusses if such a single system can deliver the best possible result for overall throughput and make better use of computational resources. We examine two practical approaches to make HPC resources available in a dynamically reconfigurable hybrid HPC/Cloud setup, both of which can be achieved with few modifications to existing HPC/Cloud environments. The first approach (University of Melbourne) generates a consistent compute node operating system image with variation in the virtual hardware specification. The second approach (University of Freiburg) deploys a cloud-client on the HPC compute nodes, so the HPC hardware can run Cloud-Workloads for backfilling free compute slots. This paper is an extension of work originally presented in the 2017 IEEE 13th International Conference on e-Science [1], with a new exploration of container implementations in such environments.

2 The HPC/Cloud Conflict

HPC systems running massively parallel jobs require a fairly static software environment, running on bare metal hardware, with a high speed interconnect in order to reach their full potential. Even then, they only offer linear performance scaling for cleverly designed applications. Massively parallel workloads need to be synchronized. Fitting several of these jobs into the individual nodes' schedules will necessarily leave gaps, since jobs have to wait until a sufficient number of nodes become available. The scheduler is "playing Tetris" with large, incompatible pieces, which can lead to under-utilization of the whole system. In contrast, cloud workloads typically consist of small tasks, each only using fractions of the available compute resources, but often in large quantities. Cloud computing offers flexible operating system and software environments which can be optimally leveraged in cases of embarrassingly parallel workloads that do not suffer from slow inter-node connections. A cloud environment offers more flexibility at the expense of the virtualization overhead and the loss of efficient, multi-node high-speed low-latency communication.

Large scale projects, e.g. the Large Hadron Collider from the particle physicist community, have a tendency to develop extremely large and complex software stacks. There exists a well-founded lay-assumption that a computational experiment that deals with objective values ought to be reproducible by other researchers. However this is often not the case as researchers are unaware of the details of their operating environment (operating system, application, and dependencies etc) [2]. One potential solution to the reproducibility issue is to be found in containerisation technology, which permits encapsulation of the entire operating environment. Unfortunately this approach has a couple costs; there is a modest performance penalty, and it requires that the container owner manages be the entire environment.

In addition to the more general problems of HPC and Cloud workloads, finding a truly optimal hybrid system should also consider resource allocation issues. An often encountered instance of this problem is when one parallel application on a HPC system requests 51% of the available resources while another application needs 50% of said resources. It is impossible to co-locate these workloads, which unfortunately leave a large portion of nodes idle if those are the only two tasks scheduled. Another resource allocation problem arises in the scheduling of jobs which different behaviours. In the case where two types of jobs (single-node, short term vs. multi-node, longer term) are submitted to a job scheduler with backfilling to maximise resource utilisation, the overall system can still experience sub-optimal utilization. The single node jobs, request fewer resources and will have priority over the multinode jobs, thereby reducing the cost-effectiveness of the expensive high-speed interconnect and resulting in idle cluster resources.

When looking for a hybrid system which offers flexibility and performance in the face of such problems, it is also worth noting that applications and their datasets usually have varied and characteristic computational workflows that are more or less appropriate for different computational architectures. It is certainly preferable from a user's perspective that a single system is capable of adapting to these diverse requirements, rather than requiring the user to migrate data between different systems depending on the task. A hybrid system must be found in order to address these sub-optimal resource utilization issues.

3 Hybrid Architectures

Both Melbourne and Freiburg Universities have been operating various compute clusters for more than two decades. While the variation in hardware architecture has decreased, new possibilities have arisen from X86 hardware virtualization becoming ubiquitous. It has allowed both sites to host multiple operating systems on a single server and to strictly separate their complete operating environments. Hardware and software stacks are decoupled. While widespread in computer center operation, virtualization is still underutilized in HPC environments. Projects like hybrid cluster approaches and the use of Virtualized Research Environments (VRE) drastically changes the landscape.

3.1 HPC with Compute Nodes as Cloud VMs

The University of Melbourne approach consists of a traditional HPC cluster with a high speed interconnect in one partition (or queue), and an alternative partition providing virtual machines (VM) managed through OpenStack as compute nodes. Using the Slurm Work-

load Manager multiple partitions are accessible; the bare metal partition in place for a traditional HPC architecture), while the National eResearch Collaboration Tools and Resources project (NeCTAR) research cloud provides generic virtual machines). Additionally there are private departmental partitions (the "water" and "ashley" partitions) , a specialist proteomics partition ("punim0095"), a general GPU partition, a large general-purpose graphics processing (GPGPU) partition for recipients of a specific Linkage, Infrastructure, Equipment, and Facilities (LIEF) grant from the Australian Research Council, and others. In total there are 24 partitions, including those used for debugging and testing purposes, and nodes can belong to multiple partitions simultaneously. Despite their heterogenous high level use cases, the general hardware specifications can be summarised as follows (Fig. 1):

- Physical partition: 12 core, Intel Xeon CPU E5-2643 v3, 3.40GHz, 256GB RAM.
- Cloud partition: 8 core, Intel Haswell, 2.3GHz, 64GB RAM.
- GPGPU partition: 24 core, Intel Xeon E5-2650 v4, 2.20GHz, 128GB RAM, 4 Tesla P100 cards.
- Ceph filesystem: 4.5PB for /home, /project, and /scratch storage.
- Network varies by partition; cloud partition on 10GbE with Cisco Nexus and physical partition with Mellanox ConnectX4 cards and SN2100 switch.

The VMs on the cloud partition use a common image just like the traditional HPC compute nodes, but with differing virtual hardware based on the results of job profiling and user requests. Each of these have an nodelist and are generated by VM images. Deployment of compute node according to partition is carried out with a simple script which invokes the OpenStack Nova service to deploy specific images. These can be deployed as either a static definition, or dynamically using Slurm's cloud-bursting capabilities. In addition the login and management nodes are also deployed as virtual machines.

Jobs are submitted to the Slurm workload manager, specifying which partition that they wish to operate on. The collection of virtual machines comes from the Melbourne share of the Australian-wide NeCTAR research cloud [3]. The VMs, with different virtual hardware, can be configured flexibly into different partitions in accordance with user needs. However, unlike a lot of VM deployments, overcommitment of resources is used. Early testing indicated that whilst there is good boundary separation through the virtualisation model, overcommit usage resulted in unexpected time mismatch errors on concurrent compute tasks. As a result the entire virtual machine architecture operates with a 1:1 ratio with physical machines. While this removes the normal advantages of overcommit for scheduling low utilization processes, it instead simplifies deployment and offers significant flexibility in extending or reducing the size of partition as required.

Of particular importance is assuring that the HPC "physical" partition has a high-speed interconnect. Mellanox 2100 switches with 16 x 100Gb ports with a mixture of 25/50/100Gb, maximum of 64 x 25Gb connections with RDMA over Ethernet and Cumulus Linux OS. An Message Passing Interface (MPI) "ping-pong" test was conducted between two compute nodes on separate infrastructure, with 40GbE RDMA over ethernet receiving better latency results than 56Gb Infiniband FDR14 on a comparable system [4].

3.2 HPC with Cloud VMs on Compute Nodes

The University of Freiburg runs an HPC cluster for Tier-3 users coming from different scientific communities and fields of interest. After a primary installation of 752 nodes in 2016 the cluster now commands more than 1000 compute nodes equipped with dual socket 10 core plus HT Intel Xeon E5-2650 v4, 2.20GHz CPUs and 128GB of memory each. The local operating system is remotely booted over the 1Gb Ethernet interface which also handles most of the user traffic. The cluster is equipped with 100Gb OmniPath offering 44 node islands which are aggregated by 4 100 Gb links. This high speed low latency network provides both MPI functionality for HPC jobs and access to the BeeGFS parallel filesystem.

Standard bare metal jobs are scheduled by Adaptive Moab. On top of the standard HPC jobs (bare metal jobs), it enables users to run virtual machines as standard compute jobs (VM jobs). In order to run VMs on a compute node, a virtualization hypervisor is installed on every compute node using the standard Linux Kernel-based Virtual Machine (KVM) hypervisor. This architecture enables users to run both bare metal and VM jobs on the same hardware, through the same resource manager, without partitioning the cluster into isolated parts. Users that require a special software environment and do not need direct access to hardware can use the VM jobs which provide Virtual Research Environments (VRE). A VRE in this instance is a container image with a complete software stack installed and configured by the user [5]. Thus, the file system of a virtual machine or VRE is a disk image presented as a single file. From the operator's perspective this image is a "black box" requiring no involvement, provisioning, updating, nor any other management effort. From the researcher's perspective the VRE is an individual virtual node whose operating system, applications and configurations as well as certain hardware-level parameters, e.g. CPU and RAM, can be configured fully autonomously by the researcher.

On typical HPC clusters the resource management is orchestrated through a scheduler. Since researchers using a hybrid compute resource are allowed both to submit jobs to the scheduler for bare metal computa-

tion and initiate VRE jobs, it is necessary that the scheduler is also responsible for the resources requested by virtual machines. If VMs are executed on the cluster without the knowledge of the scheduler, resources could get overbooked. A special workflow was developed to submit a request for a VM as a normal cluster job and let the scheduler handle the request for a new virtual machine. Thus, the hybrid HPC cluster concept becomes flexible and independent of the actual scheduler deployed.

The hybrid cluster setup is not without additional costs. For the orchestration of the virtual machines on the cluster the OpenStack framework is installed. If a compute job is designed to run in a virtual environment (VM) the Moab scheduler is configured to instantiate this VM through the OpenStack API. OpenStack is selected for its modular architecture to allow to choose the components needed for the cluster virtualization, such as network, image handling or a web interface and omit the rest. While introducing additional management and operational overheads, this allows building a virtualization environment for the cluster that meets the specific objectives. Four infrastructure nodes are dedicated to running the OpenStack services and preparing VM images. These nodes are not part of the compute resources available to the user.

3.3 Containerization

A further elaboration of the hybrid architecture is the use of containers in either aforementioned model. Whilst virtual machines simulate the hardware environment (even in a 1:1 environment), a container virtualizes at the operating system level. As containers can be imported with a exact level of consistency this is seen as solution to reproducibility issues in computation.

The most well-known application for containerisation is Docker. However Docker is not a good fit for the common tasks in research computing [6]. Docker is primarily designed for micro-services on an enterprise level, or during software development on local systems. Additionally, the Docker daemon runs as root, and the group with root privileges, which has security implications, e.g. when mounting file systems. These scaling and security issues leads HPC system administrators to be resistant to the installation of Docker. As an alternative, Singularity can be deployed without the issues of Docker, to obtain the benefits of containerisation [7]. Singularity is used at the University of Melbourne as it has good integration with the Slurm Workload Manager and MPI. Security issues are mitigated by ensuring that user privileges inside the system are the same as the privileges outside the system (i.e., as a user on an HPC) and that there are no root-owned daemons or root-level privileges for the group. Another container technology that also is considered appropriate for high performance compute environments is Shifter [8]. In either case, these containers can run on HPC nodes, whether they are bare-metal or virtualized, leading to the interesting situation of a virtualized operating environment (singularity) on a VM with its own operating environment, which is running on real hardware with an additional operating environment. The saying "that there is no cloud, there is just somebody else's computer" is doubly applicable for containers.

The following is a simple example to illustrate the use of the container within a Slurm job script that makes use of an existing container.

```
#!/bin/bash
#SBATCH --partition cloud
module load Singularity/2.4-GCC-6.2.0
singularity \
 exec vsoch-hello-world-master.simg \
 echo "Hello from inside my container!" \
 > output.txt
```

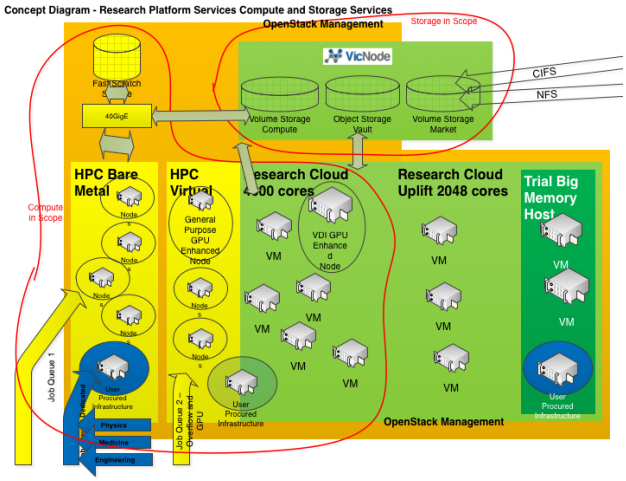

Figure 1: Concept diagram of the hybrid research platform services, combining bare metal and OpenStack resources at the University of Melbourne depicting the various partitions.

4 Workflow Design

Given the two competing primary compute resource variants, HPC scheduling and Cloud scheduling, it is necessary to orchestrate both systems through a mediator. In a hybrid approach one needs to define a primary scheduler which controls which jobs should run on which worker node and instruct the other scheduler to run within the boundaries of the other scheduler, especially respecting the scheduling decision of the other. The mediator could be a special service daemon which waits for requests from users and administrators and translates it to the OpenStack API and vice versa.

The University of Melbourne uses a traditional HPC workflow where job submission with Slurm Workload Manager occurs on different partitions according to whether they are based on physical or cloud architectures. At the University of Freiburg three workflows are present; job submission via Moab scheduler without running a resource manager client in the VM, job Submission via Moab scheduler with a resource manager client, The Terascale Open-source Resource and QUEue Manager (TORQUE) running in the VM, and job submission via OpenStack Dashboard/API.

4.1 Job Submission with Slurm Scheduler for different Partitions

A standard batch job submission workflow is a key part of the architecture. This follows the well-known path in HPC for job submission. As a unique feature, the placement of the virtual machine on the cluster nodes is scheduled by Moab and the job lifetime is coupled to the lifetime of the VM. This allows for a seamless integration with the jobs sent by other user groups and honors the fairshare policies of the cluster. A batch script makes resource requests on a particular partition. Based on the resource request, the information received from resource manager daemons, and the fairshare policies in place, the scheduler will allocate a time when the job will run which may change if other jobs finish early etc. At the appropriate time, the job will launch on one or more compute nodes, run, and write the relevant processed data and output to the directories as specified. The developed thin integration layer between OpenStack and Moab can be adapted to other batch servers and virtualization systems, making the concept also applicable for other cluster operators.

In the University of Melbourne model, the Slurm workload manager acts as the job scheduler and resource manager. Different partitions refer to the queues which a user may submit jobs and are varied by the physical or virtual hardware that have been provisioned (e.g., `#SBATCH --partition=cloud` or `#SBATCH --partition=physical`). For single node jobs, whether single or multi-core, a low speed (Ethernet) network and virtual machines are suitable, whereas for multinode jobs the physical partition with a high-speed interconnect is used. Thus, the physical architecture is optimised for the type of computational task required, increasing overall throughput and more efficiently allocating resource.

4.2 Job Submission via Moab Scheduler without running a Resource Manager Client in the VM

At the University of Freiburg, one use case is where users provide their VM images themselves. These VM images cannot be trusted and therefore they are not allowed to access cluster resources like parallel storage or user home directories. It is expected that the user is working with an external resource manager or is using the cloud-init procedure to start compute jobs within the VM. In this case the user's workflow is to submit a cluster job through the scheduler msub/qsub and, if resources are available, the job launches a virtual machine via the OpenStack API. When the VM boots data and compute instructions have to be injected by an external resource manager or through cloud-init into the virtual machine (possibly injecting the aforementioned cluster job script).

4.3 Job Submission via Moab Scheduler with a Resource Manager Client (TORQUE) running in the VM

A second use case at the University of Freiburg is when the user submits classic compute jobs to a different software environment on the cluster (Fig. 2, left branch). The software environment is represented by a VM in this case. This makes it necessary to install and run a TORQUE client in the virtual machine. In this use case the workflow begins with the user submitting a job through the scheduler (msub/qsub) with a specified image and job-script (e.g., msub -l image=¡id¿ job-script.sh). The job is then scheduled like any other bare metal job. If and when resources are available this job will trigger the start of a new virtual machine environment through the OpenStack API. When the virtual machine is booted the TORQUE client connects to the TORQUE server and receives the job which then is started within the VM. If this software environment is provided by an external source (user virtual machine) all necessary data has to be injected as well. This could be achieved for example with the stagein and stageout options from TORQUE.

4.4 Job Submission via OpenStack Dashboard/API

The third use case from the University of Freiburg is when the user submits compute jobs simply by creating a VM via the OpenStack web interface (Horizon) or OpenStack API (Fig. 2, right branch). These virtual machines then should be represented as a compute job in the Moab scheduler. The compute job script is injected via cloud-init into the virtual machine during boot and is executed in the virtual machine after the boot process is finished. In this use case the workflow is initiated by the user starting a virtual machine instance on the compute partition. OpenStack schedules this VM as any other bare metal job and, when and if resources are available, OpenStack will start up the virtual machine on the resources previously assigned by the Moab scheduler. When the virtual machine boots data and compute instructions have to be injected by an external resource manager or through cloud-init into the virtual machine.

4.5 Script and Data Injection

In the University of Melbourne case the injection of scripts and data to the virtual machine is not a significant issue, as the virtual machines are simply another node on the HPC system. The issues faced here are the same as other HPC environments. Primarily this means the need to firmly encourage users to manage their data so that it is physically close to the site of computation. Grace Hopper's famous reminder [9] to "mind your nanoseconds" is applicable even to many research scientists who ask whether remote data can have a mount-point on a HPC system in order to speed

up their computational time, and even across all compute nodes which are normally do not have access outside of the system network for security reasons. A more genuine issue is the access speed of shared mount points across a system, and education on the need for staging actions on local (e.g., `/var/local/tmp`) disk or faster shared storage (e.g., `/scratch`) before copying data to slower shared directories.

For Freiburg's architecture the situation is more complex. There are there possibilities to inject job scripts into a VM: cloud-init, TORQUE client and an external resource manager client.

- cloud-init: This is the easiest way to inject job scripts into a VM. A job script has to be specified when instantiating the VM. After booting the VM the script gets executed automatically.
- TORQUE Resource Manager Client: Since the Cluster uses the TORQUE resource manager new "virtual" clients can be added dynamically and so the job script will be injected directly through the resource manager to the client. The only challenge is to signal the TORQUE Client running the bare metal job not to execute the job script.
- External Resource Manager: If external scheduling and resource managers are implemented by a research group they can be used to inject job scripts into the VM. Once the virtual resource pops up after starting a VM on the bare metal cluster it registers at its external resource manager and then can be used as every other resource of this Resource Manager.

The nature of the VM image defines the difficulty of injecting data into the booted virtual environment. Usually virtual research environments are built by research groups containing the configuration and software which is necessary to solve the problems of that specific research groups. Since these VM images are externally built, the instantiated environments can't be trusted. In such environments mounting home and work directories is not allowed, so other methods for data injection and result retrieval from these environments must be available. There are two possibilities to do so:

- The job script can provide information on how to transfer data by copying or streaming it through the network
- The resource manager client features can be used to stage data into the VM and stage out the results. TORQUE supports staging data by specifying the options "stagein" and "stageout", the job output is copied to the remote site automatically if not specified otherwise in the TORQUE client config ($usecp).

For trusted virtual environments provided by cluster administrators, these issues are not present. For example replicating a modules software environment for experimental particle physicists. These virtual environments are fully controlled by the same people running the bare metal environment and so can be trusted. In these environments, home and work directories can be mounted, and users can be identified and authorized through the same mechanisms as in the underlying bare metal environment. They can be simply leveraged by users as special software environments with no difference to the bare metal system (ignoring the fact, that it wouldn't support multi-node usage).

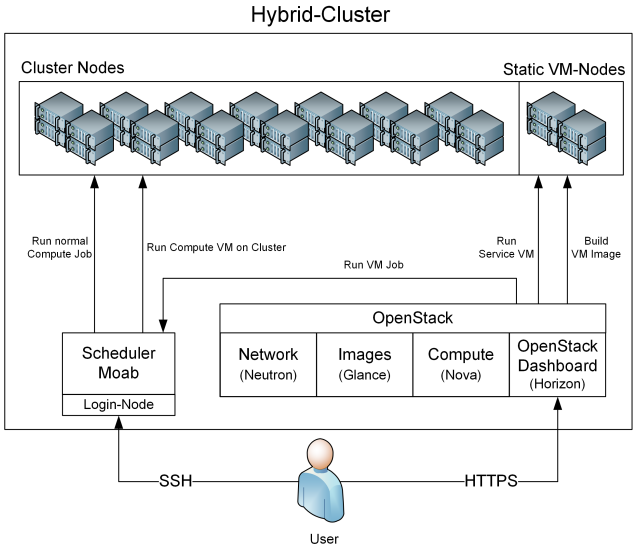

Figure 2: The Freiburg HPC cluster user has multiple options to submit compute jobs, either via the traditional way submitting it to the scheduler (left branch) or interactively by starting an appropriate virtual machine containing the relevant workflow (right branch).

5 Conclusions

Hybrid clusters make research groups more independent from the base software environment defined by the cluster operators. Operating VREs brings additional advantages like scientific reproducibility but may introduce caveats like lost cycles or a layered job scheduling. However, the advantages of this approach makes HPC resources more easily available for broader scientific communities. Both models are meant to be easily extensible by additional resources brought in by further research groups.

The two models - HPC with Cloud VMs on Compute Nodes, and HPC with Compute Nodes as Cloud VMs - represent different hybrid systems to solve different problems. In effect, the University of Freiburg model provides a "cyborg", where the HPC compute nodes are replaced with cloud virtual machines, whereas the University of Melbourne model provides a "chimera", a multi-headed beast where the virtual machines have become new cloud nodes. In the former case there was a desire to make existing compute nodes available to researchers for their particular configurations. In the latter case there was a desire to make virtual machines accessible to an HPC system to increase cost efficiency and improve throughput.

The two approaches illustrate the importance of HPC-Cloud hybrids in general purpose research computing.

Migrating complex and diverse software stacks to new HPC systems every few years constitutes a major effort in terms of human resources. By virtualization of these complex software environments, one sacrifices a fraction of performance, but one gains the possibility to run these complex software environments in cloud systems and thus literally anytime, anywhere and on any scale. This flexibility, in many cases, outweighs the loss in performance [10–12].

For future developments, the University of Melbourne plans to extend their model to provide the ability to include cloud bursting to external providers (e.g., Amazon, Azure), and hosting highly varied X86 configurations on the same system, although this requires equivalent real or virtual hardware, partial replication of the operating environment, and accounting for latency between the local and remote sites. A further option of HPC cloud-bursting that will be subject to further investigation is Moab/NODUS cloud-bursting by Adaptive Computing. The principles used by NODUS are similar to the exploration here; a workload queue has an elastic trigger that uses an API key to an template image which then invokes nodes, deploys upon them and potentially cluster nodes simultaneously, completes the job, and transfers the data as necessary. The ability for fine-grained task parallel workloads across local HPC and remote cloud providers suggests an obvious difficulty.

The University of Freiburg model is rather complex and took a while to mature. It is in production since the official start of the cluster in mid 2016. The HPC/cloud group hopes to improve their model, mapping Moab commands to OpenStack commands allowing to pause/hibernate and resume the virtual machine for preemption or maintenance instead of killing a job. In addition the possibility of mapping the live migration of virtual machine to Moab during runtime will give the opportunity to migrate compute jobs during runtime to optimize the overall cluster utilization. For the next generation HPC cluster we will reconsider the options to reduce the complexity of the VRE scheduling. Following the Melbourne model a distinct cloud partition for the future system is definitely an option.

Acknowledgment Lev Lafayette would like to thank Bernard Meade, Linh Vu, Greg Sauter, and David Perry from the University of Melbourne for their contributions to this document. The authors from Freiburg University would like to thank the Ministry of Science, Research and the Arts of Baden-Württemberg (MWK) and the German Research Foundation (DFG) which funded both the research infrastructure NEMO and the ViCE eScience project on Virtual Research Environments.

References

[1] L. Lafayette and B. Wiebelt, "Spartan and NEMO: Two HPC-cloud hybrid implementations", in *13th International Conference on e-Science*, IEEE, Oct. 2017. DOI: 10.1109/escience.2017.70.

[2] V. V. Sochat, C. J. Prybol, and G. M. Kurtzer, "Enhancing reproducibility in scientific computing: Metrics and registry for singularity containers", *PLOS ONE*, vol. 12, no. 11, C. Antoniewski, Ed., e0188511, Nov. 2017. DOI: 10.1371/journal.pone.0188511.

[3] Z. Li, R. Ranjan, L. O'Brien, H. Zhang, M. A. Babar, A. Y. Zomaya, and L. Wang, "On the communication variability analysis of the NeCTAR research cloud system", *IEEE Systems Journal*, vol. 12, no. 2, pp. 1506–1517, Jun. 2018. DOI: 10.1109/jsyst.2015.2483759.

[4] B. Meade, L. Lafayette, G. Sauter, and D. Tosello, *Spartan HPC-Cloud Hybrid: Delivering performance and flexibility*, 2017. DOI: 10.4225/49/58ead90dceaaa.

[5] M. Janczyk, B. Wiebelt, and D. von Suchodoletz, "Virtualized research environments on the bwForCluster NEMO", in *Proceedings of the 4th bwHPC Symposium*, 2017. DOI: 10.15496/publikation-25205.

[6] G. Kurtzer, *Singularity: Containers for science, reproducibility, and high performance computing*, Keynote at Stanford Conference, 2017.

[7] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute", *PLOS ONE*, vol. 12, no. 5, A. Gursoy, Ed., e0177459, May 2017. DOI: 10.1371/journal.pone.0177459.

[8] D. M. Jacobsen and R. S. Canon, "Contain this, unleashing docker for HPC", in *Proceedings of the Cray User Group*, 2015.

[9] L. Gilbert, *Particular Passions: Grace Murray Hopper (Women of Wisdom)*. Lynn Gilbert Inc., 2012.

[10] A. Gupta and D. Milojicic, "Evaluation of HPC applications on cloud", in *Sixth Open Cirrus Summit*, IEEE, Oct. 2011. DOI: 10.1109/ocs.2011.10.

[11] P. Church and A. Goscinski, "Iaas clouds vs. clusters for hpc: A performance study", in *Cloud Computing 2011 : The 2nd International Conference on Cloud Computing, GRIDS, and Virtualization*, IARIA, 2011, pp. 39–45.

[12] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, "A comparative study of high-performance computing on the cloud", in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing - HPDC '13*, ACM Press, 2013. DOI: 10.1145/2493123.2462919.