SAND2021-3399C

Take Me to the Clouds Above: Bridging On Site HPC with Clouds for Capacity Workloads

Jay Lofstead
Sandia National Laboratories
Albuquerque, NM, USA
email: gflofst@sandia.gov

Dmitry Duplyakin
School of Computing, University of Utah
Salt Lake City, UT, USA
email: dmdu@cs.utah.edu

***Abstract*—Sites with limited compute cluster capacity aimed at supporting large-scale applications of scientific and parallel computing must deal with the additional demand for small-scale jobs—in many cases, single-node and coming in large volumes—that help further develop the capabilities of large-scale applications as well as run simple data analysis tasks. Many of these analysis tasks and other small-scale jobs are run on High Performance Computing (HPC) systems because they offer familiar environments, making future scaling convenient, or because the proximity to the input or output data sets is important. When these small-scale jobs explode in count and create significant competition for resources, ensuring that the most effective use of the cluster resources is achieved requires either non-trivial scheduler tuning for better management of job mixes or adoption of entirely different models for management of computing environments. We claim that a hybrid "on-site HPC + cloud" model can offer the best of both worlds. Thus, small jobs should target cloud resources in cases where they can be offloaded to clouds without significant penalties on performance, and, at the same time, large-scale application runs can better utilize HPC clusters, achieving lower wait times and increasing job throughput. This is not without challenges.**

***Index Terms*—HPC, Cloud, cloud bursting, job scheduling**

I. Introduction

Organizations with large-scale computing needs constantly have to balance two different workloads on their computing infrastructures. Some target workloads include *large-scale, scale-up* jobs that often use extra hardware. For exmaple, (Graphic Processing Units) GPUs offer extreme parallel processing at lower power usage, but with limitations on accessing resources outside the GPU package or potentially even other processes on the GPU. Field Programmable Gate Arrays, FPGAs, offer a way to program hardware enabling very low power processing with a highly customized processing structure. These kinds of speciality tools are becoming more common as ways to achieve more compute efficiency at large scale. At the same time, workloads with *small-scale* jobs need to co-exist on these systems with large-scale jobs. These workloads include much smaller jobs for development, analysis, and initial exploration that need tiny fractions of the available resources and may not be optimized to take advantage of the specialized hardware. Supporting both types of workloads and achieving good balance between them are important components of computing missions within many organizations, in academic, research, and commercial sectors.

For example, Sandia National Laboratories (Sandia) has workloads that match these patterns. At Sandia, we field a large machine (Trinity [1]) in cooperation with Los Alamos National Lab as well as smaller but still decently sized machines (e.g., Astra [2]) on our own. In spite of this compute capability, Sandia still fields numerous additional clusters to handle capacity needs. The intent is for the capacity machines to serve for development and small scale runs prompting scheduler priorities to focus on first fit rather than job age or size related priorities. Part of the challenge for Sandia is the presence of export controlled or classified computing jobs prompting on site computing resources expansion. Using a public, shared resource may not be possible due to the security restrictions. Some of the concerns with running specific workloads related to export control or sensitive or classified data or processing have special requirements of the cloud platform and the connection with it. Certification [3] can allow workloads and data for some workloads while higher consequence and more sensitive information still cannot use these infrastructures.

Even with this generous compute capacity on site, Sandia still has computing demands that could best be served in a cloud environment based on tool requirements or data locality, such as work on a cloud hosted, shared data set. Integrating these systems, particularly considering the security concerns, is problematic.

Smaller HPC systems, such as, for instances, university clusters and machines at smaller research laboratories, have similar usage demands but not as much computing capacity available. Unlike Sandia, many of these systems do not have strong security requirements, which allows their users to freely leverage public cloud resources as overflow capacity. However, even in the organizations where cloud transition is acceptable and supported by the internal cloud teams, not all users migrate their smaller jobs to the cloud resources. Such migration has much potential for on-site resources to be better serving the workloads that depend on their availability, yet this potential is primarily unrealized within the organization.

One hurdle to integrating multiple platforms is the lack of effective cross platform schedulers and job build and deployment resources. During the grid era, numerous efforts attempted to make a single front-end for various grid back-end systems [4]. They all suffered from complexity of different back-end system features and architectures, among other limitations.

In this context, the question of tools to address this from a policy and technology infrastructure basis need to be explored to inform effectively addressing workload balancing.

The rest of this paper is structured as follows. First, in Section II, we discuss the state of the art for job schedulers in HPC and cloud computing as well as the challenges that arise in integration of resources from different sources. Next, in Section III, we describe the cloud-related challenges and opportunities as they have been explored in various hybrid and multi-cloud studies. Next, Section IV presents a survey of related work. A discussion of recommendations on how to approach solving the hybrid bursting problem is discussed in Section V. Finally, Section VI offers takeaway requirements and challenges we need to address as a community to achieve seamless on site HPC and public cloud resources, whether the cloud has high security instances or not.

II. Job Scheduling

Task scheduling system generally fall into one of three categories with some newer systems attempting to bridge either the categories, platforms supported, or both.

A. Scale Up Task Scheduling

The predominant scheduler in use in HPC centers today is Slurm [5]. As an open source tool with strong community support, it has grown to handle most HPC compute management needs. In particular, it has recently been enhanced to offer support for multi-resource scheduling [6]. This enables heterogeneous node types to be effectively scheduled ensuring that jobs that require special hardware will only run on nodes that can support them while ensuring that jobs that do not need that special hardware do not overly delay the special jobs.

While Slurm is certainly not the only scheduler available, it is dominant at the USA's Federally Funded Research and Development Centers as it reduces costs and better supports building affordable infrastructure for the private sector to use. Other schedulers are discussed briefly in the related work in Section IV.

A next generation HPC scheduler, Flux [7] offers these features and is intending to extend into supporting integrated cloud infrastructures. However, Flux is just starting to explore how to incorporate cloud resources and has only scratched the surface of the vast complexities involved. It will take considerable time and effort for Flux to successfully integrate a single cloud platform. In the meantime, recommendations and policy changes that can be implemented quickly will offer relief.

HPC oriented schedulers focus on a single task or small number of tasks per job with each task potentially using 10000 nodes or more for each task. The priority is to gather sufficient resources according to the scheduler policy settings to successfully run each queued job at the right time. In these cases, task scheduling throughput in the 100s of jobs/tasks per second is sufficient to address the workload needs.

B. Scale Out Task Scheduling

At the other extreme are scale out workloads. These tend into two categories. First are the many data analytics tasks or other job types that add additional compute to reduce compute time. They use independent processing and can just divide the workload into more parts to achieve faster throughput. Genomics processing is one example with a massive data set that can be divided into many independent pieces for exploring matching against another genetic sequence. These tasks tend to be smaller, maybe 10s of nodes at most, and can be restarted independently if one should fail without affecting the other tasks. The other category is rapidly running tasks that collectively perform a more complex operation. If a task runs for 1 second or less, the scheduler must be able to schedule many thousands per second across a large machine to keep the scheduler from being the throughput bottleneck. Some modern software engineering architectures, such as function-as-a-service [8], embrace this kind of short task execution model as a central feature.

The need to handle scheduling a large number of short running tasks prompted the creation of Sparrow [9] and similar systems. This shift from the traditional very large single task with a long run time demanded these new tools to better handle resource use.

With large scale task oriented systems, task scheduling throughput more on the order of 10000s is required to ensure the compute is the bottleneck rather than the scheduler. Different job schedulers oriented for this environment, such as for Spark [10], offer better throughput using different base assumptions. For example, a single tasks can be scheduled either on a single node or a single core making resource selection a far easier task. Considerations about interference effects from interconnect network traffic are not important. Other interference effects, such as those from cache sharing, are typically not a top priority. Mesos [11] with systems like Aurora [12] and Yarn [13] offer examples of high throughput task oriented schedulers.

Other systems like Omega [14] were built in frustration of the need to support heterogeneous clusters that evolved as new hardware was added over time with broken and obsolete hardware decommissioned. The priority for a system like this is to support a wide variety of hardware features and enable a reasonably efficient mapping from job requirements onto the available hardware balancing needs against availability.

C. Container Orchestration

The alternative approach for job scheduling has shifted more to container management rather than task management. Systems like Kubernetes [15] and Docker Swarm [16] offer increasingly rich and complex environments for deploying long-lived services that can dynamically scale on demand. This is a fundamentally different kind of workload making it a poor match for either scale out or scale up workloads without rethinking their architectures.

D. Discussion

The kinds of workloads each of these system classes addresses is different and difficult to address with a single scheduler and resource management system. This has led to the fragmentation of platform development efforts, where each platform is essentially treated as an independent direction for research and development and optimized to best address its own, particular subset of the workloads.

The real challenge being faced by large scale compute centers today is that the various tools used in each of these different environments are starting to be demanded within others. For example, machine learning tools are now being incorporated into scientific simulations. For example, in climate simulations [17], [18], machine learning models are substituting for parts of the model that may have too many parameters or the physics is not fully understood. Using models generated from observational data, reasonable estimates of these effects improve the simulation model quality overall.

Other examples of cross platform tool usage include data analytics tools for use on simulation generated data sets. Being able to run the analytics tools with the simulation would accelerate insight discovery by shortening the exploration time. How to best handle these hybrid workloads further complicates the scheduling picture.

III. Cloud Challenges and Opportunities

Public cloud systems are essentially walled gardens offering data storage, compute capacity, and often, many tools and services that make application development fast and easy. This is quite attractive for many different disciplines enabling less costly tech company startup costs (i.e., outsource all of the compute infrastructure needed to an on-demand cloud service eliminating the need for hardware purchases for peak usage times and hiring system administrators). Fueling innovation is not the only advantage. Other institutions that need a relatively large amount of computing power for a short period of time find cloud services a cost effective approach to meeting their computing needs.

In spite of the advantages offered, trying to combine a single cloud environment with either a second cloud or another platform is far from an easy or inexpensive endeavour.

A. Performance

A large number of studies reported performance limitations and downsides of cloud computing environments [19]–[21]. These studies pointed out the lack of low-latency networks, high virtualization overheads, significant performance variability due to resource sharing, among other concerns. However, one common fact about these studies is that it has been nearly a decade since they have been published and, therefore, the results they include essentially relate to the clouds of previous generations. On the contrary, the offerings from today's clouds provide access to resources with impressive processor performance, increased memory sizes, and highly optimized networks in isolation from other user's traffic. Additionally, a recent comprehensive study [22] indicates that the performance gap between HPC and clouds is essentially closed, at least at small and moderate scales. In fact, it is shown that a cloud system offers higher bandwidth and lower latency than a production HPC system, deployed as recently as several years ago.

B. Data Movement

One of the most serious concerns with cloud-based processing is the data movement. Cloud systems, such as Amazon Web Services, offer free data ingress as an enticement for moving a data set onto the service and to use their compute resources. However, while data movement onto the platform is free, moving data out frequently incurs a charge. Also, data storage on the platform usually involves charges as well. With limited ingress bandwidth, storing large datasets for repeated processing becomes the only reasonable option forcing recurring costs. A related challenge may be the security of data migraion between platforms [23].

C. Spot Pricing

In many cases, cloud pricing is not fixed, but varies dynamically based on supply and demand. In these cases, trying to control compute job costs by migrating between clouds or moving from an HPC resource to a cloud is further complicated. Effectively managing these costs variances has led to management systems [24]. It is a far from simple task to achieve the lowest costs compute, particularly when data movement delays and costs are incorporated.

D. Job Requirements

The way applications are packaged for use on an HPC resource compared to each different cloud is different. In some cases, it is a container. In others, it is virtual machines. Others still require that you build your application in the cloud environment and it is stored using an internal format, typically something similar to a virtual machine. With an HPC job that uses Slurm or any of the other HPC-oriented schedulers, the applications and job scripts have been written and optimized for a particular platform with specific dynamic library versions available and access to particular storage systems with specific interfaces. For example, the S3 interface on AWS has a completely different interface than the typical POSIX API of an HPC scratch space or one of the database (SQL or NoSQL both) systems. Key-value stores have yet another interface, although it is most similar to S3. This variety of tools required for processing makes deploying an application automatically from and HPC to cloud or vice versa a difficult proposition.

E. Discussion

While the cloud environment is best (cheapest) when it can be used with small, transient data on a single cloud data center, the above research demonstrates the potential benefits and deep challenges with effectively combining one cloud data center with some other compute resource, be it another cloud or an onsite HPC or cloud platform. The challenges are not insurmountable, but are difficult.

The different deployment and job scheduling interfaces are serious challenges that are not a simple matter of money. Instead they require significant research and development effort to construct a reasonable approach that considers many of the challenges identified above and the others not listed here.

In spite of these challenges and costs involved, cloud can also be a better option to manage on site inter-group conflicts. For example, at the 2017 University of California, Santa Cruz CROSS Symposium [25], the genomics institute talked about why they used AWS for their workloads even though they had sufficient infrastructure on site to handle their workloads. The challenge came down to having a third party arbiter for who should pay for what and how much they use. For example, if a researcher is using "paid for" on campus compute, they may demand to keep 10 PiB of data because it is all precious. That same researcher, when presented with a US$20,000 bill for storing that data, they may suddenly decide that more than 90% of it was not that critical after all. Further, by using the third party arbiter, deciding who gets to use the compute and when is more a matter of just allocating budget rather than arguments over whose turn it is to use the machines. Even though the costs can be significantly higher than on site resources, having the direct costs and third party arbiter proved to be a significant advantage for managing multiple research groups using a shared resource.

IV. Related Work

Two commercial job schedulers for HPC workloads include the Cray scheduler (ALPS) [26] and IBM Cobalt Scheduler [27]. Both of these systems offer full features and excellent performance optimized for their proprietary platform environment, but are limited to the vendor platforms. With the strong emphasis on open source for HPC through efforts like OpenHPC [28], these systems offer a difficult value proposition. Users can get the same tools across a variety of platforms, but have to switch their job scripts and other management infrastructure for every user if adopting one of the proprietary platforms.

The literature that discusses HPC systems at national laboratories [29]–[31] provides rich information about HPC usage trends, resource utilization metrics, evolution of supercomputers over time, among many other user- and system-centered topics. Most of the existing studies on HPC environments—both historic and also recent—pay little attention to cloud computing and its benefits, considering integration with clouds to be secondary or optional in nature. With the shifting workload demands, revisiting these investigations is an important priority.

Among the counterexamples, a study of computing resources at the Texas Advanced Computing Center (TACC) [32] stands out as it describes a considerable number of cloud-style jobs being processed as a result of integration of TACC facilities with Jetstream [33], a cloud computing facility sponsored and managed by the USA's National Science Foundation (NSF).

Chameleon Cloud [34] offers a bare-metal-as-a-service cloud option. While this is an NSF supported effort focused on supporting both research and education, the resources are not as extreme as leadership computing facilities. Instead, the focus is on supporting smaller scale efforts with a strong tie to educational environments rather than production-style workloads.

CloudLab [35] and other cloud testbeds offer a different take on the cloud environment by incorporating new hardware and software to test out how to use these new tools in a cloud environment. These systems, while looking at experimental system design, do not have the strong ties to a production HPC environment nor the capacity to handle offload for a heavy workload.

With traditional HPC requiring more specialized system administration capabilities and more expert friendly compute management interfaces, the cloud oriented environment enables running scale out workloads or even scale up workloads with friendlier tools. Academic institutions trying to address their entire user base focus on the majority users with a cloud-friendly configuration that can also support, albiet without fully optimized configurations, scale up computing workloads to some degree. This has been discussed in good detail by Hwang, et al. [36].

The final part of the related work concerns cloud bursting. These systems look at how to use a cloud resource as "overflow" for an onsite or just another large scale compute resource. Work on these bursting approaches [19], [37], [38] show the challenges and potential for making these systems work. Microsoft, with the Azure platform, offer this as a fundamental part of their cloud strategy. They encourage users to install an Azure instance at their premises and to use Microsoft's private cloud instances as bursting capacity. This enables customers to right size their on site compute resources to control costs while not hitting limits for transient peak workloads. Achieving this kind of balance for HPC and Cloud would offer an excellent balance by deploying jobs that do not need the HPC platform characteristics onto the associated cloud when there is demand for the HPC platform specific characteristics by jobs. However, these capabilities still do not exist.

V. Recommendations

As with most things in life, you get what you measure and what you reward. In this case, using incentives can change behavior prompting users to move workload most compatible with cloud infrastructures off the on-site HPC resources. Rewarding "good behavior" while monitoring systems to detect "bad beahvior", these goals can be achieved quickly.

We were unable to find any publications about these kinds of management strategies prompting our work to formalize the understanding of the problem space and start investigating and measuring solution effectiveness. This paper is the first step in that process.

A. Long Term

Long term, using a scheduler like Flux, once it has fully incorporated cloud capabilities, will be the “correct” choice for managing multi-platform resources with varying workload characteristics. By “correct”, we mean that it will offer a general, simple, least cost way to bridge between the platforms with low user involvement. Other approaches will likely still be better for specific use cases. In the mean time, some simpler approaches can be used to encourage users to reconsider which platform they run their workloads on.

Also long term, adopting a Kubernetes-like environment with virtual machines and containers, deploying different workloads may be easier at a performance and complexity cost. Versioning these artifacts makes reproducibility and replicability of computational science easier as a side effect and worth considering for simply that reason. Further discussion about this topic is beyond the scope of this paper.

B. Short Term

Short term approaches focus on the encouragement via measurement and reward approaches. Below are a selection of ideas to explore in this space.

First, judicially use time-based priority based on job size and resource requirement. For tiny jobs, impose something like a 1 year age penalty for scheduling priority. For large jobs, give a bonus. Then the scheduler can automatically adjust which jobs get run. While this seems simplistic, it can offer a basic metric to shift the workload balance. Gaming the system by allocating more resources than needed is likely prompting other approaches in combination (see below).

Second, adjust allocations and priority based on how the compute resources are used. For example, if nodes have GPUs, ensuring that jobs significantly use the GPUs, can improve effective usage. In this case, monitoring system characteristics, such as power and heat on different system components, can yield a better measure of what kind of work a particular job actually performs. This can offer bonuses or penalties for future compute use based on the past job characteristics. This is not a perfect measure since a user could incorporate a GPU benchmark running in the background while their simple CPU-only workload runs, the additional effort to incorporate such cheating is not worth the effort. With policy that could ban users for such behavior and user support to use alternative resources, such as cloud systems, the chances are reduced considerably. For special cases, a user could apply for an exemption for a special project enabling exceptional cases while automating a strong policy. In combination with the time priority, the instances of cheating can be greatly curtailed.

Third, consider offering HPC allocation bonuses based on cloud usage. If a user makes an effort to run their smaller jobs on the cloud, rewarding them with additional large run compute allocations can make achieving their scientific goals easier and faster. Once the learning curve of using the cloud is mastered, the user becomes bi-lingual and can deploy on whatever platform makes the most sense. This is not an easy goal to achieve, but possible with minimal system interventions and offering training and user support that is likely currently supported within the organization.

Overall, any attempt to encourage users to take advantage of a second platform needs to consider the added complexity for the user. What benefit (or avoided penalty) is the user gaining by spending the distraction time to learn how to use a new platform? Also, these need to consider that the approach may work too well causing a mass migration to the cloud leaving an expensive HPC resource underused. While this latter case is less likely, it is a consideration if the penalties are too harsh.

These are but three approaches under evaluation. Others are certainly possible and left as future work.

VI. Conclusions

Throughout the paper, we have presented a series of requirements and challenges with integrating cloud with on site HPC. A highlight of these requirements and challenges are presented below.

For requirements, first, in all cases, training will be critical to help users understand how to use a new computing platform with different interfaces and software packing requirements. Second, proper incentives can help encourage users that are in the best interest of the overall user community. Third, considering the full cost of moving compute from one platform to another needs to be considered. Data movement in particular can incur long delays and significant costs.

For challenges, first, offering strong enough incentives that users are willing to consider using a very different platform without them moving all of their use is a delicate balance. Too little and the on site HPC still is dominated by jobs that could be offloaded. Too harsh and users will just use the cloud for everything even though the on site HPC may sit mostly idle. Second, offering a seamless interface that deploys on either platform is the ideal goal. However, the numerous challenges related to application deployment, data movement, and job submission differences have proven challenging for scientific computing system interfaces. Significant investment in these areas is required for a usable system. Third, managing costs on spot priced clouds makes deploying jobs an activity potentially fraught with danger for budgets.

This paper presents a discussion of the challenges of hybrid environments and attempting to use multiple kinds of platforms for a single workload. With broad knowledge of what traditional HPC workloads and environments look like, what cloud workloads and environments look like, and what pieces of each are most important and problematic, we believe we can take HPC to the Clouds Above!

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

References

[1] M. J. Cordery, B. Austin, H. Wassermann, C. S. Daley, N. J. Wright, S. D. Hammond, and D. Doerfler, "Analysis of cray xc30 performance using trinity-nersc-8 benchmarks and comparison with cray xe6 and ibm bg/q," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pp. 52–72, Springer, 2013.

[2] K. Pedretti, A. J. Younge, S. D. Hammond, J. Laros, M. L. Curry, M. J. Aguilar, R. J. Hoekstra, and R. Brightwell, "Chronicles of astra: challenges and lessons from the first petascale arm supercomputer," in *SC20: Proc. Int. Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[3] D. I. S. Agency, "Department of defense cloud computing security requirements guide." https://rmf.org/wp-content/uploads/2018/05/Cloud_Computing_SRG_v1r3.pdf, 2017.

[4] D. M. Batista, N. L. S. da Fonseca, and F. K. Miyazawa, "A set of schedulers for grid networks," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, (New York, NY, USA), p. 209–213, Association for Computing Machinery, 2007.

[5] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*, pp. 44–60, Springer, 2003.

[6] Y. Fan, Z. Lan, P. Rich, W. E. Allcock, M. E. Papka, B. Austin, and D. Paul, "Scheduling beyond cpus for hpc," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 97–108, 2019.

[7] D. H. Ahn, J. Garlick, M. Grondona, D. Lipari, B. Springmeyer, and M. Schulz, "Flux: A next-generation resource management framework for large hpc centers," in *2014 43rd International Conference on Parallel Processing Workshops*, pp. 9–17, IEEE, 2014.

[8] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha, "A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 162–169, 2017.

[9] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pp. 69–84, 2013.

[10] D. Cheng, Y. Chen, X. Zhou, D. Gmach, and D. Milojicic, "Adaptive scheduling of parallel jobs in spark streaming," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[11] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center.," in *NSDI*, vol. 11, pp. 22–22, 2011.

[12] F. Pfeiffer, "A scalable and resilient microservice environment with apache mesos and apache aurora," *SREcon15 Europe*, May 2015.

[13] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, pp. 1–16, 2013.

[14] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 351–364, 2013.

[15] E. A. Brewer, "Kubernetes and the path to cloud native," in *Proceedings of the sixth ACM symposium on cloud computing*, pp. 167–167, 2015.

[16] J. Turnbull, *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

[17] P. A. O'Gorman and J. G. Dwyer, "Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events," *Journal of Advances in Modeling Earth Systems*, vol. 10, no. 10, pp. 2548–2563, 2018.

[18] V. M. Krasnopolsky and M. S. Fox-Rabinovitz, "Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction," *Neural Networks*, vol. 19, no. 2, pp. 122–134, 2006.

[19] A. Gupta, P. Faraboschi, F. Gioachin, L. V. Kale, R. Kaufmann, B.-S. Lee, V. March, D. Milojicic, and C. H. Suen, "Evaluating and improving the performance and scheduling of hpc applications in cloud," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 307–321, 2014.

[20] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running hpc applications in public clouds," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 395–401, 2010.

[21] M. A. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. Cunha, and R. Buyya, "Hpc cloud for scientific and business applications: taxonomy, vision, and research challenges," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–29, 2018.

[22] G. Guidi, M. Ellis, A. Buluc, K. Yelick, and D. Culler, "10 years later: Cloud computing is closing the performance gap," *arXiv preprint arXiv:2011.00656*, 2020.

[23] I. Khalil, I. Hababeh, and A. Khreishah, "Secure inter cloud data migration," in *2016 7th International Conference on Information and Communication Systems (ICICS)*, pp. 62–67, IEEE, 2016.

[24] A. S. Sabyasachi, H. M. D. Kabir, A. M. Abdelmoniem, and S. K. Mondal, "A resilient auction framework for deadline-aware jobs in cloud spot market," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 247–249, IEEE, 2017.

[25] S. C. University of California, "CROSS Symposium." https://cross.ucsc.edu/symposium/index.html, October 2017.

[26] M. Karo, R. Lagerstrom, M. Kohnke, and C. Albing, "The application level placement scheduler," *Cray User Group*, pp. 1–7, 2006.

[27] N. Desai, "Cobalt: an open source platform for hpc system software research," in *Edinburgh BG/L System Software Workshop*, pp. 803–820, 2005.

[28] S. Q. Lau, S. Campbell, W. T. Kramer, and B. L. Tierney, "Computing protection in open hpc environments," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, pp. 207–es, 2006.

[29] T. Patel, Z. Liu, R. Kettimuthu, P. Rich, W. Allcock, and D. Tiwari, "Job characteristics on large-scale systems: Long-term analysis, quantification and implications," in *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1186–1202, IEEE Computer Society, 2020.

[30] G. P. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards understanding hpc users and systems: a nersc case study," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 206–221, 2018.

[31] G. Amvrosiadis, J. W. Park, G. R. Ganger, G. A. Gibson, E. Baseman, and N. DeBardeleben, "On the diversity of cluster workloads and its impact on research results," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pp. 533–546, 2018.

[32] N. A. Simakov, J. P. White, R. L. DeLeon, S. M. Gallo, M. D. Jones, J. T. Palmer, B. Plessinger, and T. R. Furlani, "A workload analysis of nsf's innovative hpc resources using xdmod," *arXiv preprint arXiv:1801.04306*, 2018.

[33] C. A. Stewart, T. M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. Turner, *et al.*, "Jetstream: a self-provisioned, scalable science and engineering cloud environment," in *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, pp. 1–8, 2015.

[34] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, *et al.*, "Lessons learned from the chameleon testbed," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp. 219–233, 2020.

[35] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, *et al.*, "The design and operation of cloudlab," in *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, pp. 1–14, 2019.

[36] B.-N. Hwang, C.-Y. Huang, and C.-L. Yang, "Determinants and their causal relationships affecting the adoption of cloud computing in science and technology institutions," *Innovation*, vol. 18, no. 2, pp. 164–190, 2016.

[37] T. Bicer, D. Chiu, and G. Agrawal, "A framework for data-intensive computing with cloud bursting," in *2011 IEEE international conference on cluster computing*, pp. 169–177, IEEE, 2011.

[38] W. C. Proctor, M. Packard, A. Jamthe, R. Cardone, and J. Stubbs, "Virtualizing the stampede2 supercomputer with applications to hpc in the cloud," in *Proceedings of the Practice and Experience on Advanced Research Computing*, pp. 1–6, ACM, 2018.