

# Virtualized Cloud Computing for Exascale Performance

Vijay S. Pai, Stephen P. Crago, Dong-In Kang, Mikyung Kang,  
 Karandeep Singh, Jinwoo Suh, John Paul Walters, and Andrew J. Younge  
 University of Southern California – Information Sciences Institute  
 Email corresponding author: vpai@isi.edu

**Abstract**—This paper contends that achieving real exascale performance for a collection of useful applications requires the management features provided by virtualization and cloud computing. The critical concern is to achieve a performance target while minimizing cost, and the relevant costs are largely related to bandwidth and power. Virtualization technologies provide the substrate for managing heterogeneous computing platforms, providing isolation among applications, optimizing resource allocation and communication, and customizing operating environments as needed for different workloads.

## I. INTRODUCTION

Exascale performance enables qualitatively different kinds of applications from previous generations of high-performance computing (HPC), but also faces fundamentally different challenges [1]. Although performance is the key goal in achieving exascale, the costs are fundamentally different than current systems. The cost of transistors and cores is rapidly diminishing, but power remains a critical cost. Current trends point toward “dark silicon” where systems cannot provide enough power to supply all computational resources simultaneously [2]. Achieving exascale performance thus becomes an issue of managing power consumption efficiently while meeting workload performance needs.

Potential workloads for exascale include big data analysis, computational biology, and multiphysics simulation [3], [4]. These workloads show diverse characteristics, such as having different phases with performance bounds set primarily by computation, memory bandwidth, memory latency, network bandwidth, network latency, or I/O interactions. Differing limitations suggests the need to use different resources with widely varying power and performance characteristics. We argue that for exascale to reach a diverse set of workloads, runtime systems must support system-wide allocation and scheduling of shared resources such as accelerators and network links, intelligent control of heterogeneity, and workload-specific operating-system customization. All of these requirements build naturally upon the management features of virtualized cloud computing infrastructures. While virtualization adds some overhead, recent work shows that this overhead is shrinking and, in some cases, becoming negligible [5]–[7]. Further overhead reductions will arise as processors and devices continue to add hardware virtualization support [8], [9], ideally becoming comparable to the overhead of using a compiler rather than hand-coded assembly.

Note that exascale OS and runtime face numerous other challenges, such as fault tolerance and parallel programming. Our concurrent position paper addresses those issues more thoroughly, as well as the connection between them [10].

## II. PROPOSED STRATEGY

Two techniques form the foundation of the proposed OS and runtime strategies described here: cost-based resource

allocation and dynamic adaptation. The following describes each of these in more detail.

### A. Cost-based resource allocation

Current resource allocation models in the cloud use capability-based decision-making: for example, if an application requests a specific CPU architecture, it can only be assigned to a node with that architecture [11]. Similarly, code with a GPU section must be assigned to a GPU node. However, this model is both too restrictive and incomplete. It is too restrictive because any system can run any code *correctly*, even if not with the desired efficiency. It is incomplete because it does not provide a system-wide way of prioritizing jobs for heterogeneous resources or meeting power constraints.

Instead, this work suggests using a *cost model* for resource allocation decisions. The cost model considers characteristics of applications and architectures. For example, it may suggest that a control-heavy section of code can run on an x86 CPU ten times faster than on a GPU but that a highly concurrent region of the same code with a CUDA implementation can run 100 times faster on a GPU. At first, these cost parameters could be specified by the programmer; however, a better method would be to perform application-level analysis of kernels and correlate those to microbenchmark performance (following the method of Saavedra and Smith [12]). Further, work should be assigned to cores in heterogeneous systems so as to balance the workload when running parallel applications, as the scheduler can consider achievable platform performance to avoid load imbalances that hurt concurrency.

Beyond core-level architecture and performance, however, the cost model must also consider the impact of inter-core communication. Applications will have information about overall communication volume and frequency so that those applications with high communication needs can be scheduled with communicating cores topologically close to each other. This will both reduce the network latency of critical messages and reduce congestion. Applications with low communication needs can be topologically assigned in a “best effort” way.

The system must also consider power-efficiency. Each resource will register an active power estimate with the scheduler (again based either on an assertion by the system designer or based on active measurements). The scheduler must also know about system-level power constraints (such as the total amount of power that can be fed to a given multicore chip). Power costs arise not only from computational resources but also from communication. Just like communication performance, communication power depends on the volume of traffic, network congestion, and the topological distance of messages.

The scheduler can use a mathematical program solver to maximize aggregate throughput, minimize the latency of some job, or meet some other objective function while considering power, communication, and the performance of individual

components. Virtualization is a key enabling technology since the actual details of the computational resources are hidden from the application. Even the ISA can be considered as a virtualizable resource as accelerators begin to include features like virtualization support and virtualization platforms use binary translation. For example, a code component targeted for a GPU can be scheduled onto a CPU using binary translation if it is not performance-critical and there is no GPU currently available. The OS and runtime thus form the core of a vertically-integrated system for achieving high performance.

### B. Dynamic Adaptation

Run-time adaptation at both the application and system level is desirable. Our approach to introspection and self-management incorporates an observe-orient-decide-act (OODA) loop. The system observes the actual behavior of the running application, updating the information that was previously captured in cost model parameters used when scheduling. This includes understanding the actual sections of code that are dynamically executed and the types of operations used. Additionally, the system should monitor the dynamic communication pattern using a method such as instrumented communication libraries [13]. This analysis may lead to different outcomes than what was earlier predicted by either the application-level programmer or compiler-time static analysis.

Such introspective analysis of uncertain, irregular, or changing communication patterns enables dynamic adaptation. For example, runtime understanding of dynamic communication motivates live migration. Although live migration is often associated with fault tolerance, it can also be used for performance management [14]. Live migration is particularly enabled by virtualization since all state required for migration is closely linked. Previous work has shown that correctly mapping application ranks to actual network topology is essential for high-performance, but has typically dependent on programmer input [15], [16]. Runtime information can guide process migration, using hill-climbing or simulated annealing to gradually improve the configuration of a running job.

### III. OPPORTUNITIES

The combination of virtualization, intelligent resource allocation, and dynamic adaptation present new opportunities for achieving high performance and low power in cloud-based exascale systems. For example, virtualization allows nodes to migrate to or connect to needed resources (e.g., GPUs) only when they are actually being used, even if the hypervisor gives the impression of having those resources always available. Such dynamic utilization can lead to substantial power savings and utilization improvements by statistical multiplexing — providing only enough actual resources such that the utilization will be 100% given that applications will not typically use such resources all of the time. Additionally, this option allows the scheduler to actually allocate a given resource or just present it to the user via binary translation, recompilation, or some other software compatibility mechanism. This can be an issue if it is more expensive to use the resource (because of data copying, availability, etc) than just to emulate it. Whether through binary translation or dynamic scheduling, running jobs

in a virtualized platform can have the illusion of having their own private resources even if those resources are actually dynamically shared, underprovisioned, or completely absent.

Although much previous work in high-performance computing has sought to minimize OS activity [17], [18], many new workloads will actually benefit from having OS-controlled resources such as I/O devices closely engaged with computation. For example, big data analysis depends on both complex graph-processing algorithms and timely delivery of high-bandwidth data [19]. This observation suggests that some applications would benefit from having full-fledged OS while others would prefer trimmed OSes. Virtualization can provide this combination of features by combining a lightweight OS substrate with VM-specific OSes tuned to application needs.

Even within the scope of workloads that depend on heavyweight OS, different applications have vastly different needs [20]. For example, applications with complex communication structures benefit from network policies that emphasize low latency (e.g., disabling the Nagle algorithm [21]) while those dominated by bulk transfers prefer to minimize the interrupt overhead of acknowledgments (e.g., by receive offload). The hypervisor naturally provides isolation between the differently-tuned OSes running in subsections of the machine to minimize platform-wide interference.

### IV. CONCLUSION

This paper contends that the virtualization and management features of cloud systems make them an ideal design point for exascale OS and runtime. We specifically consider utilizing cost-based resource allocation and dynamic adaptation to improve computational throughput, power, and utilization. We show that recent work illustrates the viability of virtualization technologies in an HPC environment and the potential for energy savings when proper resource utilization is realized. This proposed research represents a unique and novel shift beyond current HPC technologies and will enable exascale for a broader class of applications.

### V. ASSESSMENT

- Challenges Addressed: resource management and adaption for performance, utilization, and power efficiency
- Maturity: Grounded in large-scale cloud deployments found throughout industry (e.g. Google, Amazon, Microsoft).
- Uniqueness: Cost-based resource allocation and dynamic adaptation combine with lightweight, high performance virtualization in a heterogeneous environment to drastically shift the design of a usable exascale system.
- Novelty: Exploits new opportunities enabled by cloud
- Applicability: Managed HPC cloud potentially lowers the barrier of entry for today's scientists and researchers using supercomputing technologies
- Effort: Near-term efforts focus on improving the applicability of virtualization by minimizing overhead and supporting live migration. Advances in virtual machine snapshotting will drastically improve migration times. Heterogeneous resource allocation and dynamic adaptation mechanisms will be explored. Long term goals include system-wide support for resiliency and performance management.

## REFERENCES

- [1] J. Dongarra, P. Beckman, T. Moore *et al.*, “The international exascale software project roadmap,” *International Journal of High Performance Computing Applications*, vol. 25, no. 1, pp. 3–60, 2011.
- [2] H. Esmailzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Dark Silicon and the End of Multicore Scaling,” in *Proceedings of the 38th International Symposium on Computer Architecture (ISCA)*, Jun. 2011.
- [3] A. Bishop *et al.*, “Scientific grand challenges in national security: the role of computing at the extreme scale,” Oct. 2009.
- [4] P. Kogge *et al.*, “Exascale computing study: Technology challenges in achieving exascale systems,” Sep. 2008.
- [5] C. Vecchiola, S. Pandey, and R. Buyya, “High-performance cloud computing: A view of scientific applications,” in *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009.
- [6] P. Luszczek, E. Meek, S. Moore, D. Terpstra, V. M. Weaver, and J. Dongarra, “Evaluation of the hpc challenge benchmarks in virtualized environments,” in *Proceedings of the 2011 international conference on Parallel Processing - Volume 2*, ser. Euro-Par’11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 436–445. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-29740-3\\_49](http://dx.doi.org/10.1007/978-3-642-29740-3_49)
- [7] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, “Analysis of virtualization technologies for high performance computing environments,” in *Proceedings of the IEEE International Conference on Cloud Computing*. IEEE Computer Society, 2011, pp. 9–16.
- [8] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert, “Intel virtualization technology for directed i/o,” *Intel technology journal*, vol. 10, no. 3, pp. 179–192, 2006.
- [9] L. Case, “Nvidia makes the GPU virtual,” *PCWorld*, May 2012.
- [10] J. P. Walters, S. P. Crago, V. S. Pai, K. Singh, A. J. Younge, and K. M. Zick, “Enabling resilience through introspection and virtualization,” Jul. 2012.
- [11] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. Walters, “Heterogeneous cloud computing,” in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, sept. 2011, pp. 378–385.
- [12] R. H. Saavedra and A. J. Smith, “Analysis of benchmark characteristics and benchmark performance prediction,” *ACM Trans. Comput. Syst.*, vol. 14, no. 4, pp. 344–384, Nov. 1996. [Online]. Available: <http://doi.acm.org/10.1145/235543.235545>
- [13] M. Kang, D.-I. Kang, S. P. Crago, G.-L. Park, and J. Lee, “Design and development of a run-time monitor for multi-core architectures in cloud computing,” *Sensors*, vol. 11, no. 4, pp. 3595–3610, 2011. [Online]. Available: <http://www.mdpi.com/1424-8220/11/4/3595>
- [14] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, “Proactive fault tolerance for hpc with xen virtualization,” in *Proceedings of the 21st annual international conference on Supercomputing*, ser. ICS ’07. New York, NY, USA: ACM, 2007, pp. 23–32. [Online]. Available: <http://doi.acm.org/10.1145/1274971.1274978>
- [15] P. Balaji, R. Gupta, A. Vishnu, and P. H. Beckman, “Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems,” *Computer Science - R&D*, vol. 26, no. 3-4, pp. 247–256, 2011.
- [16] V. Vishwanath, M. Hereld, V. Morozov, and M. E. Papka, “Topology-aware data movement and staging for i/o acceleration on blue gene/p supercomputing systems,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’11. New York, NY, USA: ACM, 2011, pp. 19:1–19:11. [Online]. Available: <http://doi.acm.org/10.1145/2063384.2063409>
- [17] P. De, V. Mann, and U. Mittaly, “Handling os jitter on multicore multithreaded systems,” in *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, ser. IPDPS ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–12. [Online]. Available: <http://dx.doi.org/10.1109/IPDPS.2009.5161046>
- [18] S. Oral *et al.*, “Reducing application runtime variability on Jaguar XT5,” in *Cray Users Group Conference*, May 2010.
- [19] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, “Introducing the Graph 500,” in *Cray Users Group Conference*, May 2010.
- [20] M. Liu, J. Zhai, Y. Zhai, X. Ma, and W. Chen, “One optimized i/o configuration per hpc application: leveraging the configurability of cloud,” in *Proceedings of the Second Asia-Pacific Workshop on Systems*, ser. APSys ’11. ACM, 2011, pp. 15:1–15:5. [Online]. Available: <http://doi.acm.org/10.1145/2103799.2103818>
- [21] J. Nagle, “RFC 896: Congestion control in IP/TCP internetworks,” Jan. 1984. [Online]. Available: <ftp://ftp.internic.net/rfc/rfc896.txt>, <ftp://ftp.math.utah.edu/pub/rfc/rfc896.txt>