

Position Paper for DOE OS/R Call for Papers

Core Containers: System Support for Provisioning and Controlling Dynamic Exascale Codes
Karsten Schwan, Greg Eisenhauer, Ada Gavrilovska, Matt Wolf, Sudha Yalamanchili – Georgia Tech
Hasan Abbasi, Scott Klasky – ORNL; Jay Lofstead – Sandia

Challenges Addressed: Dynamics in Exascale Systems. System support for future high end machines must address the rich and varied nature of their applications. These will be comprised of multiple (not single) linked simulations interacting via dynamic application properties, running along with online analytics and visualizations that comprehend simulation outputs and use them to control ongoing computations, interacting with complex online methods for controlling resource usage, data movement, and dealing with failures in which certain machine resources may become unavailable while applications run or cause application's to fail in delivering useful output. These facts strongly distinguish the exascale from the petascale regime. The latter permitted single compiled and loaded MPI applications loaded into statically reserved machine resources, whereas the former requires system support in which resources are actively managed in ways that efficiently use machine resources, avoid unnecessary data movement, and directly address the potential failures seen on ever larger exascale platforms.

As a simple illustration of the dynamic behavior of future exascale codes, consider the new approach to data analytics and visualization pursued by current exascale teams, in which the analysis of 'data at rest' (i.e., on disk) is replaced by one in which data analytics are conducted and insights are obtained from 'data in motion' (i.e., analytics and visualizations are produced directly from the outputs generated by applications), on-die, on-node, and on-machine. Additional online methods also being run include application-specific techniques like uncertainty quantification, to ensure the correct functioning of application codes and/or the validity of the results they produce. Such 'on demand' applications, then, run when and if needed, at application-specific time intervals or upon occurrence of certain application or system events, and their execution may also be triggered by user desires to 'zoom into' and to deal with interesting or problematic application behavior.

Contributions and Applicability. We propose research that responds to the increasingly dynamic nature of high end applications on future machines: (1) *Dynamic change and control*: online resource controls are needed to meet the functional and operational challenges articulated above, guided by diverse metrics, including power consumption and data movement. (2) *Online constraints*: must be enforced to meet metric requirements. (3) *Resource customization*: is needed to adjust to multiple sets of dynamically defined and deployed application codes, which can be run, on demand and upon need.

We target extreme-scale science codes with our work, through ongoing collaborations with a large set of DOE and industry researchers, including (i) fusion modeling researchers, the combustion co-design effort, astrophysics researchers, and other petascale applications, (ii) leveraging the ADIOS I/O methods successfully used with petascale codes, (iii) involving research on light-weight, continuous fault detection and mitigation at ORNL (Rao) and more general resilience methods at Sandia (Lofstead and others), and (iv) in long-standing collaborative research with Intel and HP Corporations.

Concrete Elements: Constructing Core Containers. Today's batch-scheduled petascale machines offer inadequate support for running and controlling the many dynamic and concurrent activities seen on future exascale platforms. Needed are runtime methods for allocating and controlling the resources of future exascale machines, which we term *Core Containers*. The goal is to enrich machines with new functionality that provides effective, lightweight, and scalable control over how machine resources are used. (i) *Core Containers* comprising sets of nodes and node resources (e.g., cores) are the means for providing resources to the different activities required by exascale science applications (ii) They track per activity progress in lieu of container resources; and they (iii) have active controls to meet their activities' dynamic resource requirements, using methods that range from those employing spare resources upon need (e.g., to otherwise use them as 'dark silicon' to curtail power usage), to those that 'trade' resources between different containers. (iv) Allocation and management decisions are driven by flexible metrics, foremost concerned with application progress, but also to take into account the failure likelihood of container resources (e.g., due to prior detected abnormal behavior) in lieu of the robustness to failures of the activities being run, and in addition, to consider the effects on activities from the resources shared between containers (e.g., their use of a common machine interconnect).

With *core containers*, each of the actions, be they analytics, visualization, verification, or coupled simulation code, are launched into a dynamically provisionable container. This contrasts with the static, compute socket-based allocations used by most HPC schedulers today. Because launching an activity into a container is more than just placing it onto some particular node within an interconnect, we can track important execution metadata, like current connectivity or degree of resilience of the hardware, and make it easier for the application to utilize without the end user having to code all of it by hand. Above all, there is transparency for end users – it is as effortless to launch a parallel activity into a container as it is to launch a parallel job with today's static schedulers, and containers will offer high performance communication services that will appear to end users to be as easy to use as MPI or simple Fortran write statements are today. In summary, the proposed container-based runtime will move science from today's static use of petascale machine resources to a regime in which there are differential controls on resource allocations to the many concurrent activities running on future exascale machines.

Detailed technical elements include the following. (1) *Container-based runtime infrastructure*: efficient and scalable runtime abstractions for creating, deleting, and managing containers; per container managers customized to each assembly activity’s needs; inter-container methods to acquire/release resources, driven by metrics pertaining to application requirements and platform needs (e.g., power consumption). (2) *Container resilience*: methods that efficiently and reliably manage containers and can make guarantees about container states (i.e., their data), using transactional techniques using additional lightweight methods for runtime hardware/software fault monitoring, and exploiting per-node persistent memory like NVRAM (already ongoing in joint work with HP Labs). (3) *Efficient execution of dynamic exascale codes*: representative codes run in containers, to include (i) online analytics applied to the output data generated by scientific simulations and/or (ii) visual depictions of said data to better answer end user questions about current simulation state or progress, (iii) uncertainty quantification or similar actions run for representative science codes, using containers to differentially and dynamically allocate machine resources to simulation or ‘checking’ actions, and (iv) application of the concept to representative applications, leveraging our participation in the fusion modeling community and the combustion co-design team. (4) *Accelerated science*: using realistic science codes, demonstrate how dynamically elastic containers can better meet application performance goals and improve the delivery of scientific results to end users, and can help maintain platform properties, such as caps on power consumption.

Maturity. Core container technology interacts with but does not depend on specific operating systems or compiler runtimes. Instead, much like hypervisors, it represents a control layer inserted between the exascale execution environment provided by vendors and the program runtime assumed by compilers. The purpose of this layer is to manage runtime resources in the ways needed by applications, and to do so in ways that are scalable and sustainable. Scalability is obtained from the ‘divide and conquer’ approach being used, where different containers can have entirely different managers and use entirely different strategies for controlling how their internal activities use container resource sets. Those resource sets, in fact, can be quite large, extending to say, the petascale-sized resource sets used by current high end simulations. Sustainability is derived from the fact that containers and their management methods do not assume certain programming paradigms, so, do not impose limitations on the programming models now being explored by exascale researchers.

Container technology exists on individual exascale nodes and across many machine nodes. For individual manycore platforms, our previous work has developed related technology on the basis of existing open source software [2, 10], resulting in multiple on-node ‘domains’ that are isolated from each other with respect to failures but can also be elastically sized to provide to applications the resources they need. Our team is already creating separately managed containers that extend across subsets of nodes on petascale machines, a simple use case being one in which one container manages online analytics for simulation outputs [19] and the other runs visualization code [17]. Differential resource allocations to analytics vs. visualization includes those that extend across both compute and ‘staging’ nodes [1]. There will be reliability guarantees for data movements across containers [4] and in addition, to assess tradeoffs in such data movement vs. simulation/analytics performance [18].

Past research other than ours has established the utility of core containers both on and across platforms. Operating system efforts include Cellular-Disco [7], Hive [3], and K42 [9]. [3] uses resource-partitions for fault-containment, and [9] uses them to exploit locality. Missing from such work, however, are the abstractions and methods for inter-container interactions required to maintain desired larger-scale application- or platform/machine-level requirements. In particular, in exascale systems, it will not be sufficient to simply ‘gang-schedule’ the activities running in different containers (e.g., as done in [7]). Instead, the container runtime must permit arbitrary interactions, such as those needed for ‘pipelined’ containers (e.g., when one container’s activity – online analytics – processes the outputs of another’s – the simulation producing the outputs to be analyzed). Further, for such interactions, one must distinguish the control from the data plane. To move application data, one can use I/O primitives like those provided by the ADIOS componentized I/O infrastructure [11, 5]. For the control communications linking different container managers, we will leverage rich past work on many-to-many eventing mechanisms [6], as well as collective communicators [13]. In addition, we adopt from prior work an approach that obtains scalability and independence by only loosely coupling multiple container controllers by use of scalable eventing primitives; we will add discretionary reliability properties to those primitives [4]; and we will design higher level semantics that explicitly support adaptive control as a key purpose of inter-container interaction, in part by drawing on the ‘coordination’ abstractions used in the substantial body of work in distributed control theory and multi-agent systems, as described in [12].

In collaboration with industry (HP Labs) and DOE (DOE Sandia), we have gained extensive experience with the resilience technologies needed to make containers succeed, which includes online monitoring/tracking – from [8] to [15]; analysis techniques at scale [16]; mitigation methods to deal with the effects of detected abnormalities or failures (e.g., detected deviations in the outputs of a molecular simulation triggering an atomistic simulation to correct them); and prevention, methods to prevent faults or failures, like the transactional techniques for select data movement or control actions across containers.

Literature Cited

- [1] Hasan Abbasi, Greg Eisenhauer, Matthew Wolf, Karsten Schwan, and Scott Klasky. Just in time: adding value to the io pipelines of high performance applications with jitstaging. In *Proceedings of the International ACM Symposium on High-Performance Parallel and Distributed Computing*, pages 27–36, 2011.
- [2] Paul Barham, Boris Dragovic, et al. Xen and the Art of Virtualization. In *SOSP*, 2003.
- [3] J Chapin, M Rosenblum, et al. Hive: Fault Containment for Shared-memory Multiprocessors. *SIGOPS Oper. Syst. Rev.*, 1995.
- [4] Jai Dayal, Jay Lofstead, Karsten Schwan, and Ron Oldfield. Resilient data staging through mxn distributed transactions. Poster at the 6th Parallel Data Storage Workshop Supercomputing '11, November 2011.
- [5] Greg Eisenhauer, Fabián E. Bustamante, and Karsten Schwan. Event services in high performance systems. *Cluster Computing*, 4(3):243–252, 2001.
- [6] Patrick Eugster, Pascal Felber, Rachid Guerraoui, and A.-M. Kerrmárec. The many faces of publish/subscribe. Tech. Report DSC-ID:200104, École Polytechnique Fédérale de Lausanne, Lausanne, France, January 2001.
- [7] Kinshuk Govil, Dan Teodosiu, et al. Cellular Disco: Resource Management using Virtual Clusters on Shared-memory Multiprocessors. *ACM Trans. Comput. Syst.*, 2000.
- [8] Weiming Gu, Greg Eisenhauer, Karsten Schwan, and Jeffrey S. Vetter. Falcon: On-line monitoring for steering parallel programs. *Concurrency - Practice and Experience*, 10(9):699–736, 1998.
- [9] Orran Krieger, Marc Auslander, et al. K42: Building a Complete Operating System. In *EuroSys*, 2006.
- [10] Sanjay Kumar, Himanshu Raj, Karsten Schwan, and Ivan Ganey. Re-Architecting VMs for Multi-core Systems: The Sidecore Approach. In *Workshop on Interaction between Operating Systems and Computer Architecture (WIOSCA), in conjunction with ISCA'07*, 2007.
- [11] Jay Lofstead, Fang Zheng, Scott Klasky, and Karsten Schwan. Adaptable, metadata rich io methods for portable high performance io. In *In IPDPS'09, May 25-29, Rome, Italy*, 2009.
- [12] S McArthur, E Davidson, et al. Multi-Agent Systems for Power Engineering Applications Part I: Concepts, Approaches, and Technical Challenges. In *IEEE Transactions on Power Systems*, 2007.
- [13] Bala Vasanth, Jehoshua Bruck, et al. CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers. *IEEE Transactions on Parallel and Distributed Systems*, 1995.
- [14] Chengwei Wang. A flexible system integrating monitoring and analytics for managing large-scale data centers. presented at the Intel Science & Technology Center – Cloud Computing (ISTC-CC) Retreat, December 2011.
- [15] Chengwei Wang, Karsten Schwan, Vanish Talwar, Greg Eisenhauer, Liting Hu, and Matthew Wolf. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In *Proceedings of the 8th ACM international conference on Autonomic computing, ICAC '11*, pages 141–150, New York, NY, USA, 2011. ACM.
- [16] Chengwei Wang, Karsten Schwan, and Matt Wolf. Ebat: An entropy based online anomaly tester for data center management. In *Proceedings of BDIM 2009: 4th IEEE/IFIP International Workshop on Business-driven IT Management*, 2009.
- [17] Matthew Wolf, Zhongtang Cai, Weiyun Huang, and Karsten Schwan. Smartpointers: Personalized scientific data portals in your hand. In *Proceedings of SuperComputing 2002*, Nov 2002. <http://www.sc-2002.org/paperspdfs/pap.pap304.pdf>.
- [18] Fang Zheng, Hasan Abbasi, Jiangting Cao, Jai Dayal, Karsten Schwan, Matthew Wolf, Scott Klasky, and Norbert Podhorszki. In-situ i/o processing: A case for location flexibility. In *Proceedings of the 6th Parallel Data Storage Workshop Supercomputing '11*, November 2011.
- [19] Fang Zheng, Hasan Abbasi, Ciprian Docan, Jay F. Lofstead, Qing Liu, Scott Klasky, Manish Parashar, Norbert Podhorszki, Karsten Schwan, and Matthew Wolf. Predata - preparatory data analytics on peta-scale machines. In *IPDPS*, pages 1–12, 2010.