Hobbes: OS and Runtime Support for Application Composition

Ron Brightwell (Lead PI), Sandia National Labs, rbbrigh@sandia.gov
Patrick Bridges, University of New Mexico, bridges@cs.unm.edu
David E. Bernholdt, Oak Ridge National Laboratory, bernholdtde@ornl.gov
Peter Dinda, Northwestern University, pdinda@northwestern.edu
Costin Iancu, Lawrence Berkeley National Laboratory, cciancu@lbl.gov
Michael Lang, Los Alamos National Laboratory, mlang@lanl.gov
Jack Lange, University of Pittsburgh, jacklange@pitt.edu
David Lowenthal, University of Arizona, dkl@arizona.edu
Frank Mueller, North Carolina State University, mueller@cs.ncsu.edu
Ada Gavrilovska, Georgia Institute of Technology, ada@gatech.edu
Thomas Sterling, Indiana University, tron@indiana.edu

August 2016

I. Software Products and Web Sites

- Pisces Co-kernel (http://www.prognosticlab.org/pisces/) Pisces is a lightweight co-kernel architecture that is designed to allow multiple native Operating Systems to run concurrently on the same local compute node. Each Operating System instance provides an isolated enclave to a co-located workload while ensuring that it's performance not impacted by other workloads on the same local node. Pisces is primarily designed to support in-situ and composed HPC applications, which require strong performance isolation to prevent cross workload interference. Pisces currently supports co-kernels based on the Kitten Lightweight Kernel and Palacios Virtual Machine Monitor.
- Kitten Lightweight Kernel (https://software.sandia.gov/trac/kitten) Kitten is a lightweight kernel (LWK) compute node operating system, similar to previous LWKs such as SUNMOS, Puma, Cougar, and Catamount. Kitten distinguishes itself from these prior LWKs by providing a Linux-compatible user environment, a more modern and extendable codebase, and a virtual machine monitor capability via Palacios that allows full-featured guest operating systems to be loaded on-demand. Kitten is used as the operating system for isolated enclaves in the current Pisces Co-kernel architecture implementation.
- Palacios VMM (http://www.v3vee.org/palacios/) Palacios is a virtual machine monitor (VMM) that is available for public use as a community resource. Palacios is highly configurable and designed to be embeddable into different host operating systems, such as Linux and the Kitten lightweight kernel. Palacios is a non-paravirtualized VMM that makes extensive use of the virtualization extensions in modern Intel and AMD x86 processors. Palacios is a compact codebase that has been designed to be easy to understand and readily configurable for different environments. It is unique in being designed to be embeddable into other OSes instead of being implemented in the context of a specific OS. Palacios is distributed under the BSD license.
- Palacios VMM for Pisces (http://www.prognosticlab.org/palacios/) A fork of the original Palacios VMM was created for the Pisces Co-kernel architecture. This version of Palacios has additional functionality that is needed to operate in the Pisces environment. Development of the original version of Palacios and this Pisces version of Palacios is occurring in parallel.
- Leviathan Node Manager (http://www.prognosticlab.org/leviathan/) Leviathan is a intra-node management and information service for multi-enclave environments. It's goal is to explore the use of in memory databases to manage and integrate enclave instances, each running independent and isolated OS/Rs. Leviathan also serves to integrate many of our other projects under a common runtime API. At the heart of Leviathan is an information service built on a in-memory No-SQL database.
- XEMEM Shared Memory (http://www.prognosticlab.org/xemem/) XEMEM is a cross enclave local shared memory architecture meant to allow applications to directly share memory even when they are deployed inside separate OS/R instances. XEMEM provides a common API that is portable across arbitrary enclave topologies and allows unmodified application binaries to be deployed to any OS/R instance based on runtime configuration decisions.
- Node Virtualization Layer (https://github.com/HobbesOSR/nvl) The Node Virtualization Layer (NVL) is the compute node operating system component of the Hobbes project. It is designed to integrate with the Linux-based compute node operating systems typically provided by vendors and to extend them with Hobbes-project developed technologies, including the Pisces Co-kernel, Kitten Lightweight Kernel, Palacios Virtual Machine Monitor, Leviathan Node Manager, and XEMEM

inter-enclave memory sharing mechanism. This enables isolated enclaves of resources (e.g., CPUs and memory) to be carved off from the vendor's operating system and booted into customized OS/R environments. The Hobbes NVL provides interfaces for managing the collection of OS/R enclaves running on a compute node and for inter-enclave composition. The NVL currently supports running on commodity x86 PC "white boxes" running standard Linux, as well as Cray XE and XK systems running Cray's Linux Environment.

- Philix (http://philix.halek.co) is a toolchain for simplifying the creation of new third party operating systems for the Intel Xeon Phi.
- Nautilus Aerokernel (http://nautilus.halek.co/) is an extremely lightweight kernel framework designed for building hybrid run-times (HRTs). An HRT is a parallel run-time system (and its application) that runs entirely in kernel mode with full privileged access to the hardware and the ability to implement any kernel abstractions it finds useful. Currently, Nautilus runs on x86-64 NUMA machines, on the Intel Xeon Phi, and within a Hybrid Virtual Machine (HVM). We have ported the Stanford Legion parallel run-time to be an HRT with LANL's port of the HPCG benchmark to Legion as the typical benchmark. We have seen performance gains of 10-40% on x86-64 and Phi compared to HPCG and Legion on Linux. Additional run-times (with less extensive implementations) running on Nautilus include the NESL VCODE interpreter and a home-grown nested data parallel language.
- HVM (Hybrid Virtual Machine) is a component of Palacios (http://www.v3vee.org/palacios/) that allows the creation of a VM that has its cores, memory, and interrupt logic partitioned between a "regular" OS (ROS, like Linux) and an HRT, which run simultaneously. This allows an HRT to operate on the data of a user process in the ROS (via a merger of address spaces), allowing this process to treat the HRT much like an accelerator. Additionally, the HRT can be booted and invoked with latencies comparable to a process startup and an IPI, respectively. Continued development of HVM has the goal of making the ROS process/HRT interaction increasingly seamless. The overall purpose is to ease the porting of parallel run-times into the HRT model by allowing the ROS to always be used as a fallback, and by allowing a partitioning of legacy compatibility concerns (e.g., Linux system calls, files, etc) and performance concerns.
- TCASM (Transparent, Consistent, Asynchronous Shared Memory) is a cross enclave local shared memory architecture meant to allow applications to directly share memory within a single OS/R enclave or between multiple OS/R enclaves. TCASM allows an application to make asynchronous progress from the coupled co-application(s) and frees the user from having to manually manage locking. TCASM leverages copy-on-write (COW) to present a simpler interface to applications and coordinates different versions of application data.

II. Publications

[1] R. Brightwell, R. Oldfield, A. B. Maccabe, and D. E. Bernholdt. Hobbes: Composition and virtualization as the foundations of an extreme-scale OS/R. In *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '13. ACM, June 2013.

[2] J. Elliott, M. Hoemmen, and F. Mueller. Evaluating the impact of SDC on the GMRES iterative solver. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, May 2014.

[3] J. Elliott, M. Hoemmen, and F. Mueller. Exploiting data representation for fault tolerance. In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, November 2014.

[4] J. Elliott, M. Hoemmen, and F. Mueller. A numerical soft fault model for iterative linear solvers. In *Proceedings of the ACM International Symposium on High-Performance Parallel and Distributed Computing*, June 2015.

[5] D. Fiala, F. Mueller, and K. B. Ferreira. Flipsphere: A software-based DRAM error detection and correction library for HPC. In *Proceedings of the IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, September 2016.

[6] D. Fiala, F. Mueller, K. B. Ferreira, and C. Engelmann. Mini-ckpts: Surviving OS failures in persistent memory. In *Proceedings of the International Conference on Supercomputing*, June 2016.

[7] K. Hale and P. Dinda. A case for transforming parallel runtime systems into operating system kernels. In *Proceedings of the 24th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, June 2015.

[8] K. Hale and P. Dinda. Details of the case for transforming parallel runtime systems into operating system kernels. Technical Report NWU-EECS-15-1, Department of Electrical Engineering and Computer Science, Northwestern University, April 2015.

[9] K. Hale and P. Dinda. Enabling hybrid parallel runtimes through kernel and virtualization support. In *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, April 2016.

[10] K. Hale, C. Hetland, and P. Dinda. Automatic hybridization of runtime systems. In *Proceedings of the ACM Symposium on High-Performance Parallel and Distributed Computing*, June 2016.

[11] K. C. Hale and P. A. Dinda. Guarded modules: Adaptively extending the VMM's privilege into the guest. In *11th International Conference on Autonomic Computing*, ICAC '14, June 2014.

[12] A. Katti, G. Di Fatta, T. Naughton, and C. Engelmann. Scalable and fault tolerant failure detection and consensus. In *Proceedings of the 22nd European MPI Users' Group Meeting*, EuroMPI '15, September 2015.

[13] B. Kocoloski and J. Lange. Hpmmap: Lightweight memory management for commodity operating systems. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, May 2014.

[14] B. Kocoloski and J. Lange. Lightweight memory management for high performance applications in consolidated environments. *IEEE Transactions on Parallel and Distributed Systems*, PP(99), February 2015.

[15] B. Kocoloski and J. Lange. XEMEM: Efficient shared memory for composed applications on multi-OS/R exascale systems. In *Proceedings of the 24th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, June 2015.

[16] B. Kocoloski, J. Lange, H. Abbasi, D. E. Bernholdt, T. R. Jones, J. Dayal, N. Evans, M. Lang, J. Lofstead, K. Pedretti, and P. G. Bridges. System-level support for composition of applications. In *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '15, June 2015.

[17] W. Lavrijsen, C. Iancu, W. de Jong, X. Chen, and K. Schwan. Exploiting variability for energy optimization of parallel programs. In *Proceedings of the Eleventh European Conference on Computer Systems*, April 2016.

[18] S. Levy, K. B. Ferreira, P. Widener, P. G. Bridges, and O. Mondragon. How i learned to stop worrying and love in situ analytics: Leveraging latent synchronization in MPI collective algorithms. In *Proceedings of the European MPI Users and Developers Conference*, September 2016. To appear.

[19] O. H. Mondragon, P. G. Bridges, and T. Jones. Quantifying scheduling challenges for exascale system software. In *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '15, June 2015.

[20] O. H. Mondragon, P. G. Bridges, S. Levy, K. B. Ferreira, and P. Widener. Scheduling in-situ analytics in next-generation applications. In *Proceeding of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2016.

[21] O. H. Mondragon, P. G. Bridges, S. Levy, K. B. Ferreira, and P. Widener. Understanding performance interference in next-generation HPC systems. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2016. To appear. [Best Student Paper Nominee].

[22] T. Naughton, G. Smith, C. Engelmann, G. Valle, F. Aderholdt, and S. Scott. What is the right balance for performance and isolation with virtualization in HPC? In *Proceedings of the 7th Workshop on Resiliency in High Performance Computing with Clouds, Grids, and Clusters*, August 2014.

[23] D. Otstott, N. Evans, L. Ionkov, M. Zhao, and M. Lang. Enabling composite applications through an asynchronous shared memory interface. In *Proceedings of the IEEE International Conference on Big Data*, Big Data '14, October 2014.

[24] J. Ouyang, B. Kocoloski, J. R. Lange, and K. Pedretti. Achieving performance isolation with lightweight co-kernels. In *Proceedings of the 24th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, June 2015.

[25] L. Savoie, D. Lowenthal, B. R. de Supinski, T. Islam, K. Mohror, B. L. Rountree, and M. Schulz. IO aware power shifting. In *Proceedings of the ACM/IEEE International Parallel and Distributed Processing Symposium*, May 2016.

[26] M. Swiech, K. C. Hale, and P. Dinda. VMM emulation of intel hardware transactional memory. In *Proceedings of the 4th International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '14, June 2014.

[27] L. Xia, K. Hale, and P. Dinda. ConCORD: Easily exploiting memory content redundancy through the content-aware service command. In *Proceedings of the 23th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '14, June 2014.