Hobbes: OS and Runtime Support for Application Composition

Hobbes Team

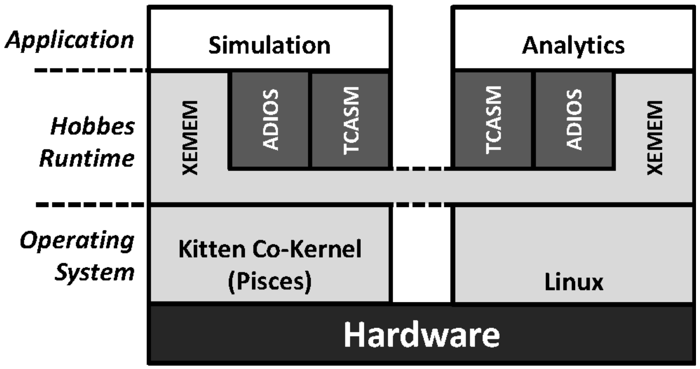
Institution	Person	Role
Georgia Institute of Technology	Karsten Schwan	PI
Indiana University	Thomas Sterling	PI
Los Alamos National Lab	Mike Lang	PI
Lawrence Berkeley National Lab	Costin Iancu	PI
North Carolina State University	Frank Mueller	PI
Northwestern University	Peter Dinda	PI
Oak Ridge National Laboratory	David Bernholdt	PI
Oak Ridge National Laboratory	Arthur B. Maccabe	Chief Scientist
Sandia National Laboratories	Ron Brightwell	Coordinating PI
University of Arizona	David Lowenthal	PI
University of California – Berkeley	Eric Brewer	PI
University of New Mexico	Patrick Bridges	PI
University of Pittsburgh	Jack Lange	PI

Project Goals

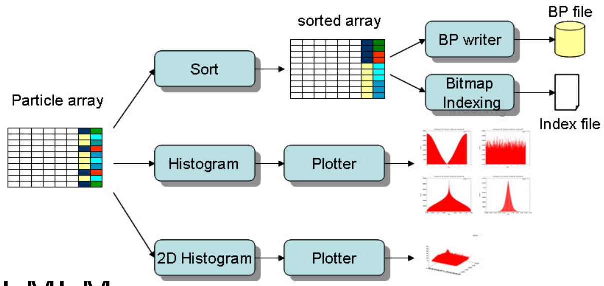
- Deliver prototype OS/R environment for R&D in extreme-scale scientific computing
- Focus on application composition as a fundamental driver
  - Develop necessary OS/R interfaces and system services required to support resource isolation and sharing
  - Support complex simulation and analysis workflows
- Provide a lightweight OS/R environment with flexibility to build custom runtimes
  - Compose applications from a collection of enclaves
- Leverage Kitten lightweight kernel and Palacios lightweight virtual machine monitor
  - Node Virtualization Layer (NVL)
  - Enable high-risk high-impact research in virtualization, energy/power, scheduling, and resilience

System-Level Support for Composition of Applications

- Problem
  - HPC applications are increasingly comprised of multiple distinct components with different requirements for OS, software stack and system resources
    - E.g., simulation+analytics, coupled multiphysics, scalable performance analysis and debugging



- Solution
  - Instantiate “enclaves” for each application **component** using high-performance virtualization technology
  - Provide OS and software stack tailored for application **component** within each enclave
  - Provide mechanisms for controlled interaction between enclaves (**components**)
    - Selective sharing of memory regions (data exchange)
    - Name service (discovery and rendezvous)

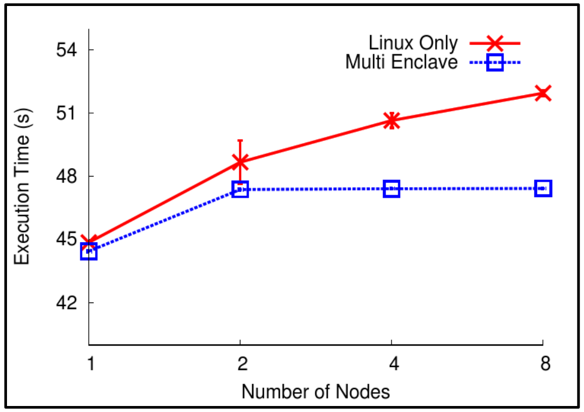


- Recent results
  - Proof-of-principle for XEMEM cross-enclave memory API
  - Use XEMEM as “transport” in ADIOS, TCASM coupling tools
  - Demonstrate composite simulation+analytics applications using XEMEM

– Impact
  - Composition can be made transparent at the application level (no changes, performance neutral)
  - Allows detailed resource management and scheduling among enclaves (other Hobbes R&D areas)

Enabling Multi-OS/R Stack Application Composition

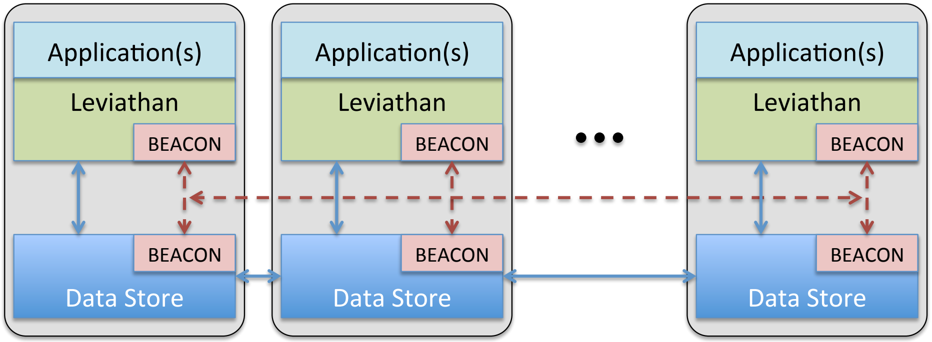
- Problem
  - HPC applications evolving to more compositional approach, overall application is a composition of coupled simulation, analysis, and tool components
  - Each component may have different OS/R requirements, no "one-size-fits-all" OS/R stack
- Solution
  - Partition node-level resources into "enclaves", run different OS/R instance in each enclave
    Pisces Co-kernel Architecture: http://www.prognosticlab.org/pisces/
  - Provide tools for creating and managing enclaves, launching applications into enclaves
    Leviathan Node Manager: http://www.prognosticlab.org/leviathan/
  - Provide mechanisms for cross-enclave application composition and synchronization
    XEMEM Shared Memory: http://www.prognosticlab.org/xemem/
- Recent results
  - Demonstrated Multi-OS/R approach provides excellent performance isolation; better than native performance possible
  - Demonstrated drop in compatibility with both commodity and Cray Linux environments
- Impact
  - Application components with differing OS/R requirements can be composed together efficiently within a compute node, minimizing off-node data movement
  - Compatible with unmodified vendor provided OS/R environments, simplifies deployment



In-situ Simulation + Analytics composition in single Linux OS vs. Multiple Enclaves

Support for extreme-scale OS/R monitoring and control

- Problem
  - Operating system/runtime (OS/R) components running throughout system must be monitored and controlled, but extreme system scale makes it difficult to do so (too much data, and/or too many "hops" to get data from one part of system to another)
- Solution
  - Integrate scalable, distributed data store with publish and subscribe service in a Global Information Bus (GIB)
  - Interface with Hobbes Leviathan node-level resource manager

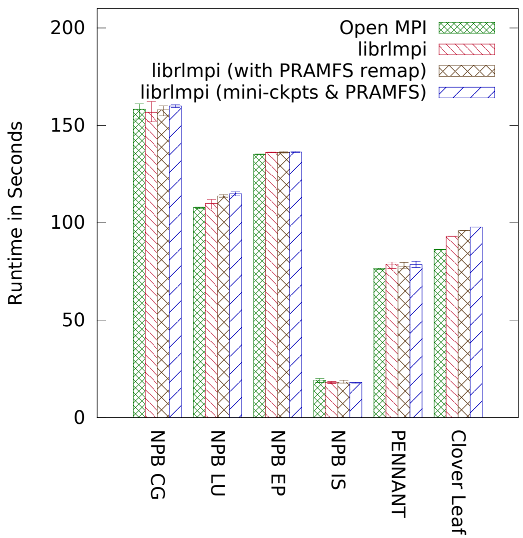


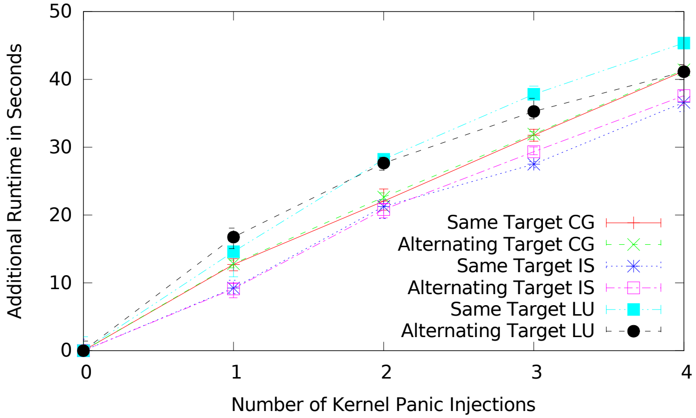
GIB data store and publish/subscribe components
Dashed lines indicate potential notifications from publishers to subscribers

– Recent progress
  - Defined important GIB use cases
    - System boot
    - Launch application
    - Respond to application failure
    - Respond to application termination
  - Designed and began pilot implementation of integration of distributed data store based on Riak open source database, BEACON publish-subscribe software from ARGO project, and Leviathan
- Impact
  - Supports monitoring and control of a large number of system software components without excessive application intrusion
  - Usable by both Hobbes and ARGO projects

mini-ckpts: Surviving OS Failures in Persistent Memory

- Problem
  - A failure of the operating system (OS) causes a failure of an otherwise healthy HPC application
- Solution
  - Execute the application in persistent memory (PRAMFS in DRAM) that is able to survive OS failures and reboots
  - Track OS state used by the application and MPI for recovery
  - Rejuvenate (warm reboot) the OS in case of a failure
  - Restore tracked OS state used by the application and MPI
  - Transparently continue to execute the application in persistent memory without loss of state/progress
- Recent results
  - Prototype implementation supports OpenMP and MPI applications with certain limitations
  - OS rejuvenation and recovery takes 3-6 seconds
  - Failure-free runtime overhead is of 3-5% for a number of key HPC workloads





- Impact
  - First solution that transparently offers OS failure tolerance without loss of state/progress
  - Transparently handling OS failures locally reduces the need for global checkpoint/restart
  - Latent OS errors that have not resulted in a failure can be cleared by rejuvinating the OS