~~New Directions in Extreme-Scale Operating Systems and Runtime Software~~

Thinking…….

Pete Beckman

Argonne National Laboratory

Co-Director, Northwestern-Argonne Institute for Science and Engineering

Senior Fellow, Computation Institute, University of Chicago

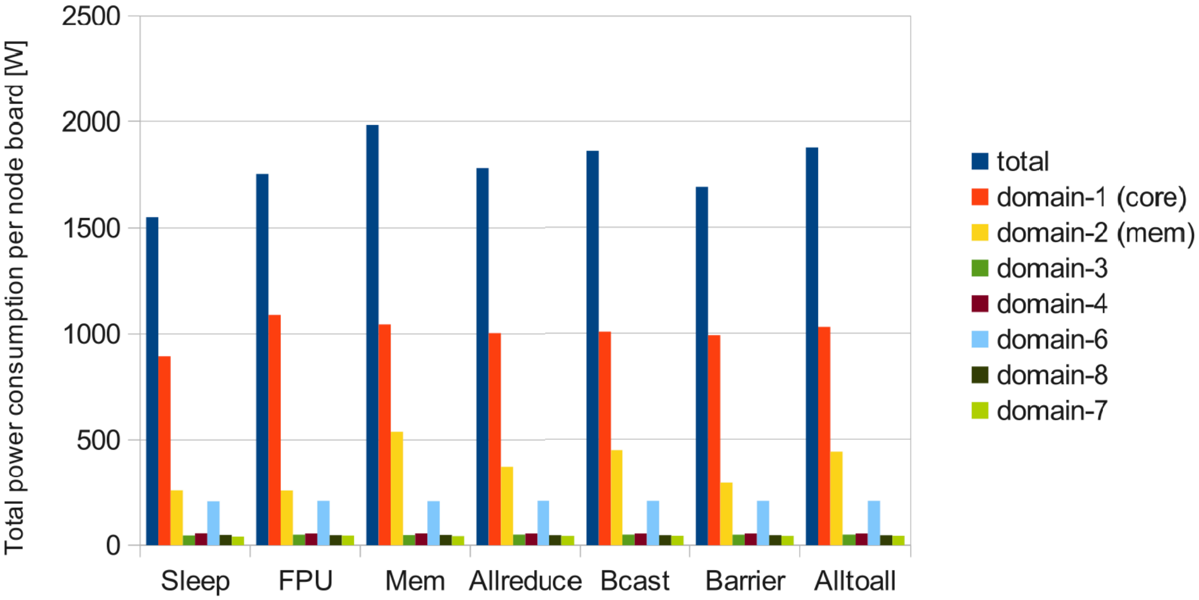
Been There, Done That… (or at least Started That)

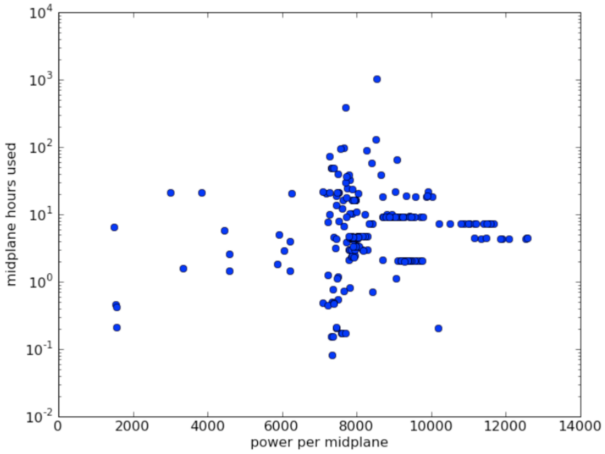
- Communication scalability, memory optimizations, topology mapping, threaded runtime (MPICH)
- Resilience
  - CIFTS (Coordinated Infrastructure for Fault Tolerant Systems): ANL, ORNL, UTK, IU, LBNL, OSU
  - MPI fault tolerance
  - Global View Resilience: DOE X-Stack 2012
  - Local checkpoint/restart, fault prediction
- Many-task, workflow, resource management, scheduling (Cobalt, ExM)
- Exascale co-design (CESAR)
- Programming models & architecture
  - New DAG-based models, memory hierarchy (LDRD)
  - Message passing + GPU
  - Efficient data movement across heterogeneous memory
  - Computer architecture (10x10, GVR)
  - Exploring Blue Gene and Intel MIC and active management
- Scalable Operating Systems & Runtime (ZeptoOS)

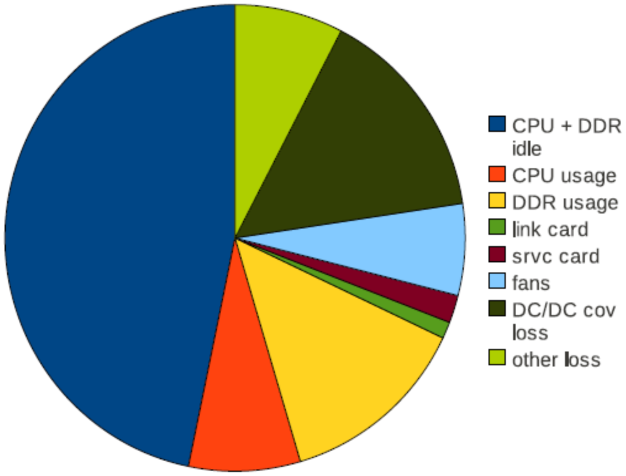
OS/R

- Broad Experience:
  - Memory allocators
  - Simple power measurements
  - Understanding of noise
  - I/O forwarding, collective operations
  - Very small kernels
  - Messaging
  - Fault sharing / managing

A few BG/P & BG/Q Power Experiments

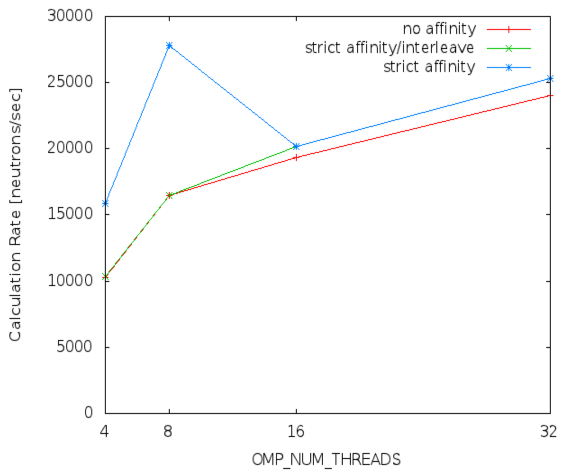


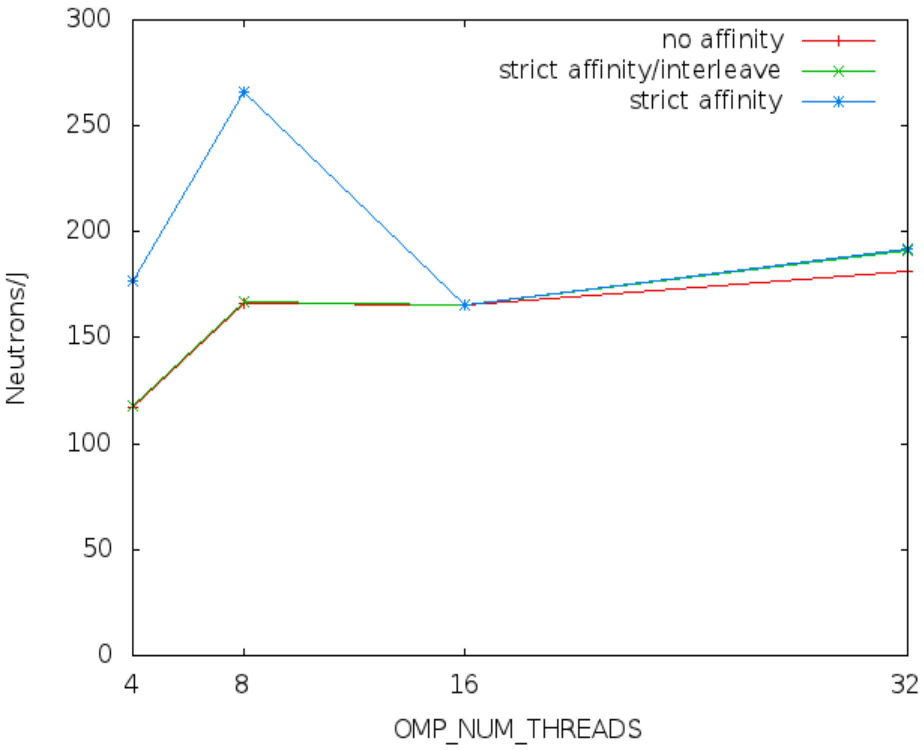


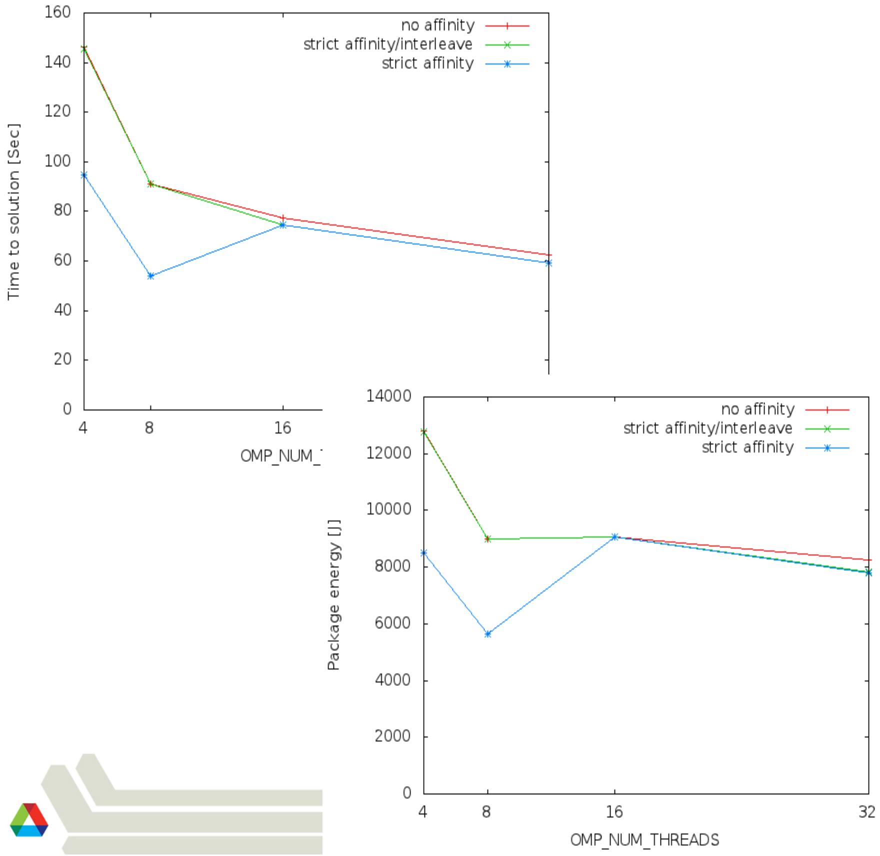


BGP power distribution for QCD running on a single rack

Energy Profiling: OpenMC

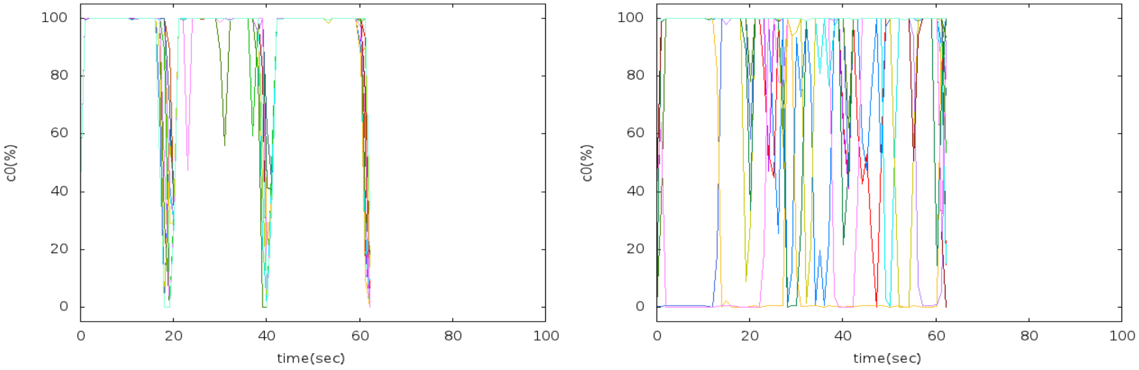






Approx. 30% improvement in power efficiency with 8 threads, strict affinity, compared to the system default (32 threads, no affinity)

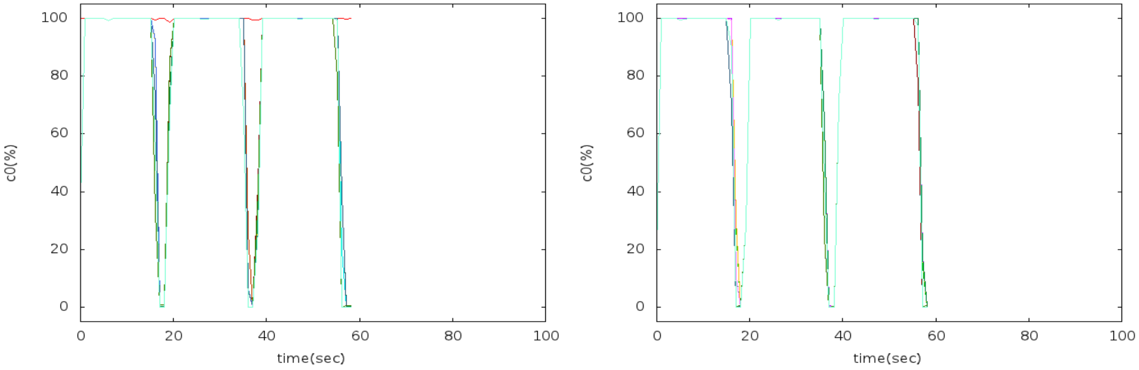
What’s Happening:



N=32 without affinity set (default)

The number of OMP threads is set to the maximum number. 32 threads in this case.

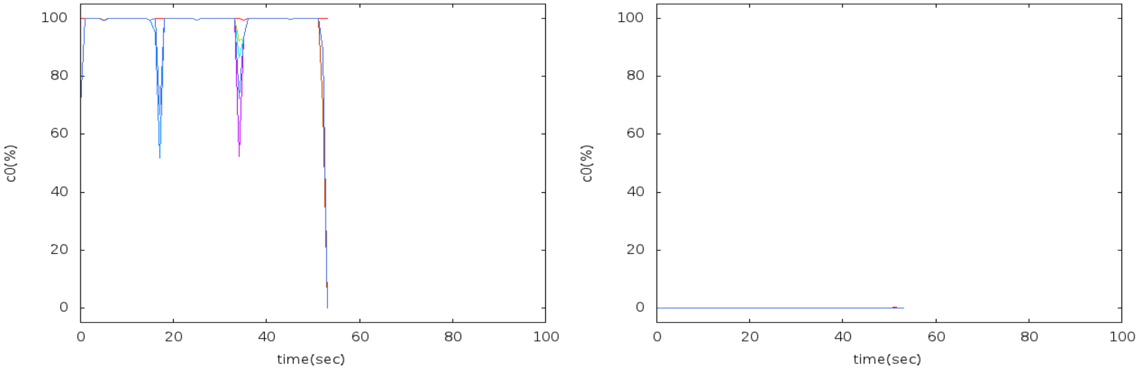
Threads will migrate.



N=32 with affinity set

Threads won't migrate

two sockets are in use

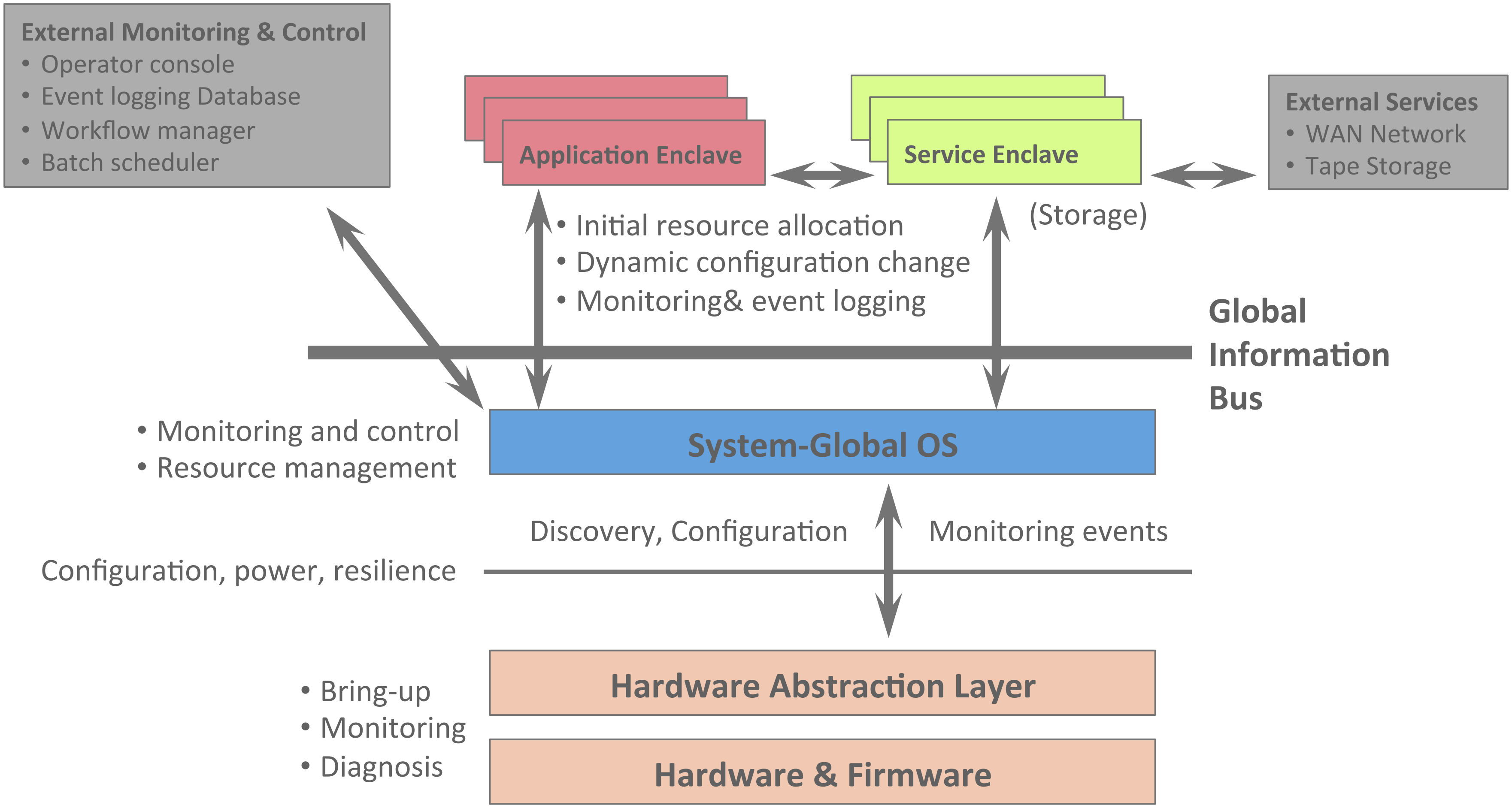


N=8 with affinity set

Threads won't migrate

Only one socket is in use

SYSTEM VIEW



External Monitoring & Control
- Operator console
- Event logging Database
- Workflow manager
- Batch scheduler

Application Enclave

Service Enclave

External Services
- WAN Network
- Tape Storage

- Initial resource allocation
- Dynamic configuration change
- Monitoring& event logging

(Storage)

Global Information Bus

- Monitoring and control
- Resource management

System-Global OS

Discovery, Configuration

Monitoring events

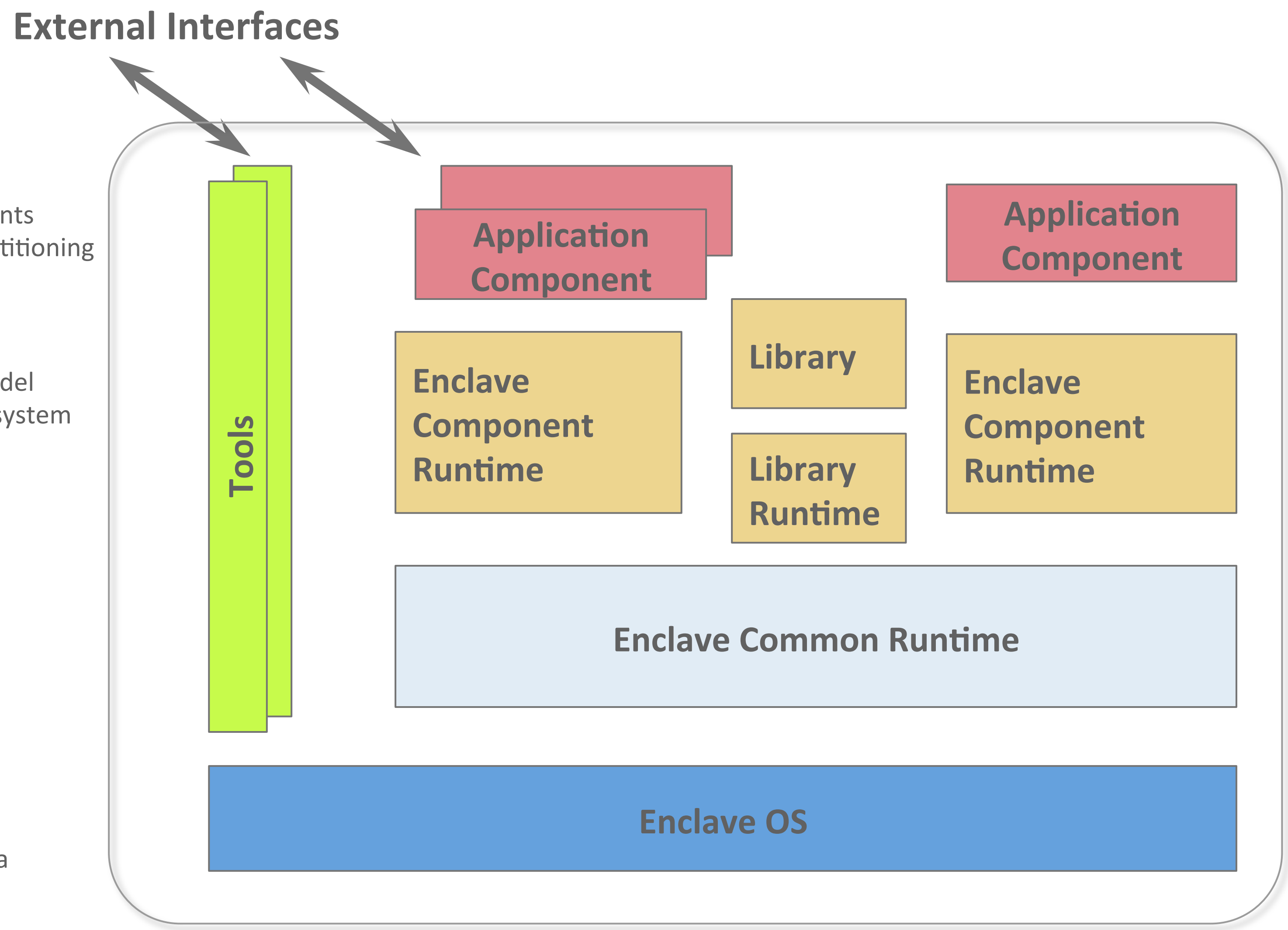
Configuration, power, resilience

Hardware Abstraction Layer

Hardware & Firmware

- Bring-up
- Monitoring
- Diagnosis

ENCLAVE VIEW



External Interfaces

Tools

Application Component

Application Component

Enclave Component Runtime

Library

Library Runtime

Enclave Component Runtime

Enclave Common Runtime

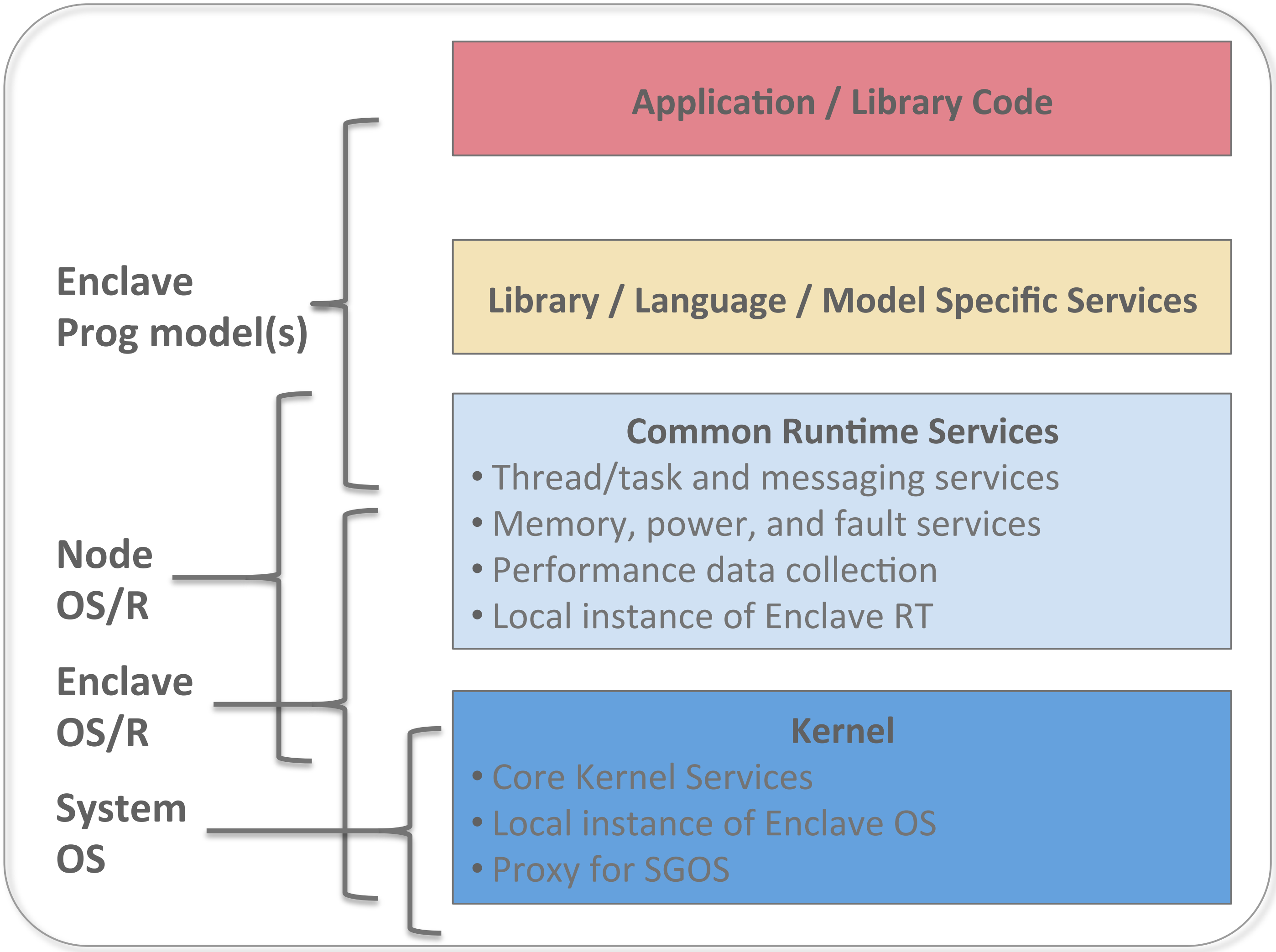
Enclave OS

Parallel components
time or space partitioning

Programming model
Specific runtime system

Power
Resilience
Performance Data

NODE-LOCAL VIEW



Application / Library Code

Library / Language / Model Specific Services

Common Runtime Services

- Thread/task and messaging services
- Memory, power, and fault services
- Performance data collection
- Local instance of Enclave RT

Kernel

- Core Kernel Services
- Local instance of Enclave OS
- Proxy for SGOS

Enclave Prog model(s)

Node OS/R

Enclave OS/R

System OS

Understanding Next Generation Memory Systems (Nonvolatile Memory) (from Jeff Vetter, et al)

- Use of NV-SCAVANGER, a tool for studying potential impact of DRAM-NVRAM partitioning of an application's data structures
  - Results on Nek on 2D eddy problem
  - 31% of memory footprint accessed in pattern suitable for NVRAM, suggesting a 28% reduction in overall power consumption.

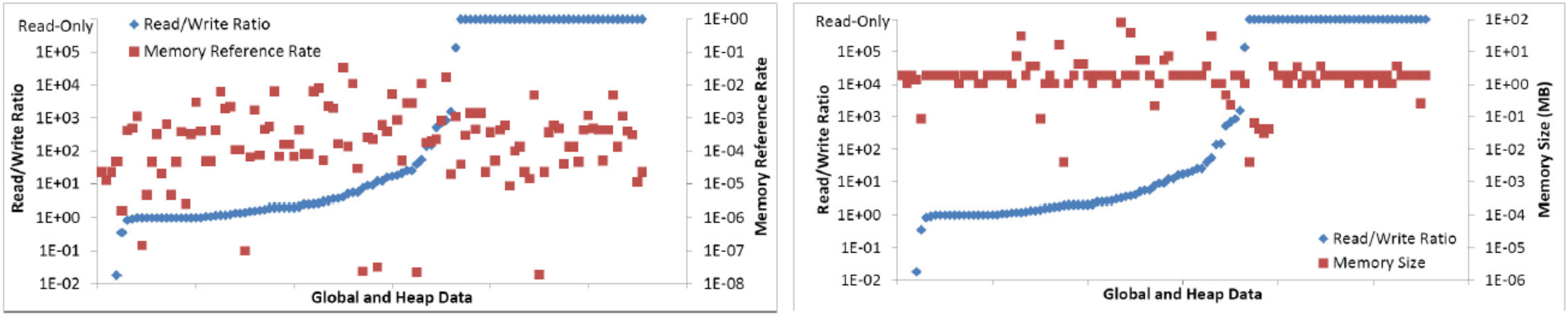


Figure 3: Read/write ratios, memory reference rates and memory object sizes for memory objects in Nek5000

D. Li, J.S. Vetter *et al., "Identifying Opportunities for Byte-Addressable Non-Volatile Memory in Extreme-Scale Scientific Applications," in IEEE International Parallel and Distributed Processing Symposium (IPDPS). Shanghai: IEEEE, 2012*

But… Where I *want* to go with OS/R

- Power
  - Understand it
  - Use it wisely to **GO FASTER**
  - Manage it in real time based on model of usage
- Fault
  - Multiple memory allocators for different fault responses
  - Whole enclave fault response
- Memory
  - Multiple hierarchical memory allocators with auto migration to NVRAM
  - Structured blocked memory transfers within a node (send/recv; put/get) across the hierarchy
- Messaging
  - HW support for large numbers of message-activated lightweight threads
  - lightweight threads in the kernel connected to message queues and RMI queues
  - *REAL* active messages & Put/Get
- Kernel:
  - Core specialization / Fused OS
  - Dynamic Task Graph Execution
  - Lightweight performance tools for adaptation, power, fault, etc.
  - Auto migration big.LITTLE for power / speed
- The ***New Argonne Parallel Programming Model***
- Not interesting: Virtual Machines, writing kernels from scratch