EXASCALE OPERATING SYSTEM AND RUN-TIME COMMON SOFTWARE ARCHITECTURE

Peter Beckman (ANL), Ron Brightwell (SNL) & Stefen Hofmeyer (LBNL)

John Kubiatowicz (LBNL), Barney Maccabe (ORNL) & Marc Snir (ANL)

May 2014

BACKGROUND

Motivation, Challenges, Design Principles

Background

- 3 OS/R projects funded under ExaOSR:
  - Argo (PI: Pete Beckman, Chief Scientist: Marc Snir)
  - Hobbes (PI: Ron Brightwell, Chief Scientist: Barney Maccabe)
  - X-ARCC (PI: Stefen Hofmeyer, Chief Scientist: John Kubiatowicz)
- Projects share a common view of Exascale OS and runtime structure
  - They work jointly on some of the components
  - Explore different implementations for others
  - Have complementary foci
- Talk presents common vision and status of projects

Exascale Challenges

- **Scale**: billions of threads
- **Heterogeneity**:
  - Cores: throughput and latency optimized cores & accelerators
  - Memory hierarchies, including SRAM, nearby and remote DRAM, NVRAM; heavy NUMA. possbly nonchoerent
- **Energy**: power as a first-class resource
- **Resilience**: frequent and possibly silent HW errors
- **Variability**: coping with continuous change and variable execution speed
- **New workloads**: Workflows, simulation+analysis, multicomponent applications,
- **Complexity**!

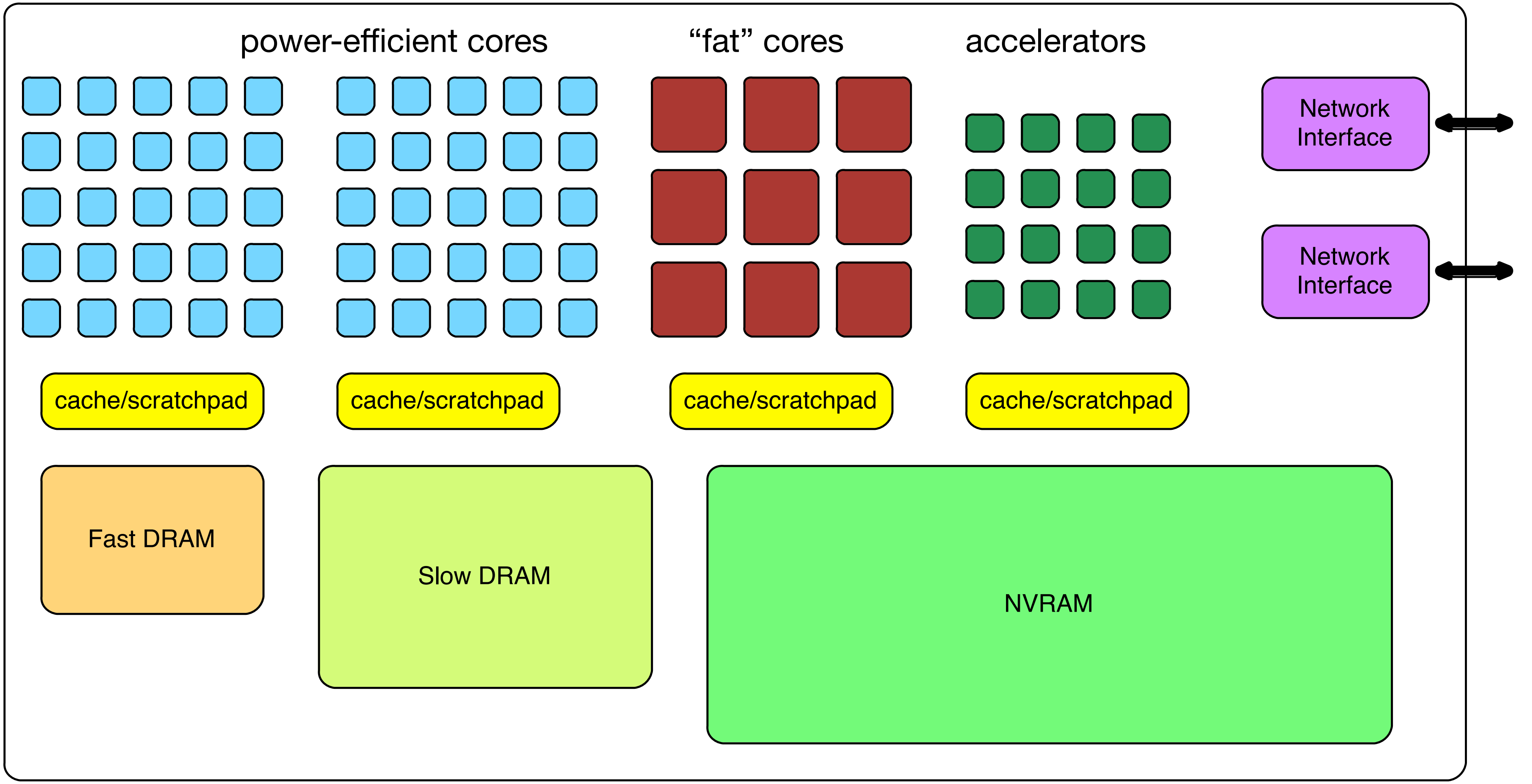
Current limitations

- Resource management is machine-global and static
- No management of power or network bandwidth and only limited management of I/O resources
- No flexibility in error/fault management
- No constructs for coordinating workflows
- Unproven (at best) capabilities for managing node challenges: O(1000) threads, heterogeneity of cores, or complex memory hierarchies
- Overly simplistic definitions and mechanisms for supporting isolation

Design Principles

- Exploit **hierarchy** to enable scalability
- Manage resources in **runtime**, rather than OS
- Runtime can be **application specific**
- Support for **adaptive resource management**: hierarchical control with feedback
- **Performance isolation** (QOS) to enhance resource utilization (and avoid over provisioning)
- **Fault isolation** to support local, independent recovery
- **Customization** to support variety in software and hardware

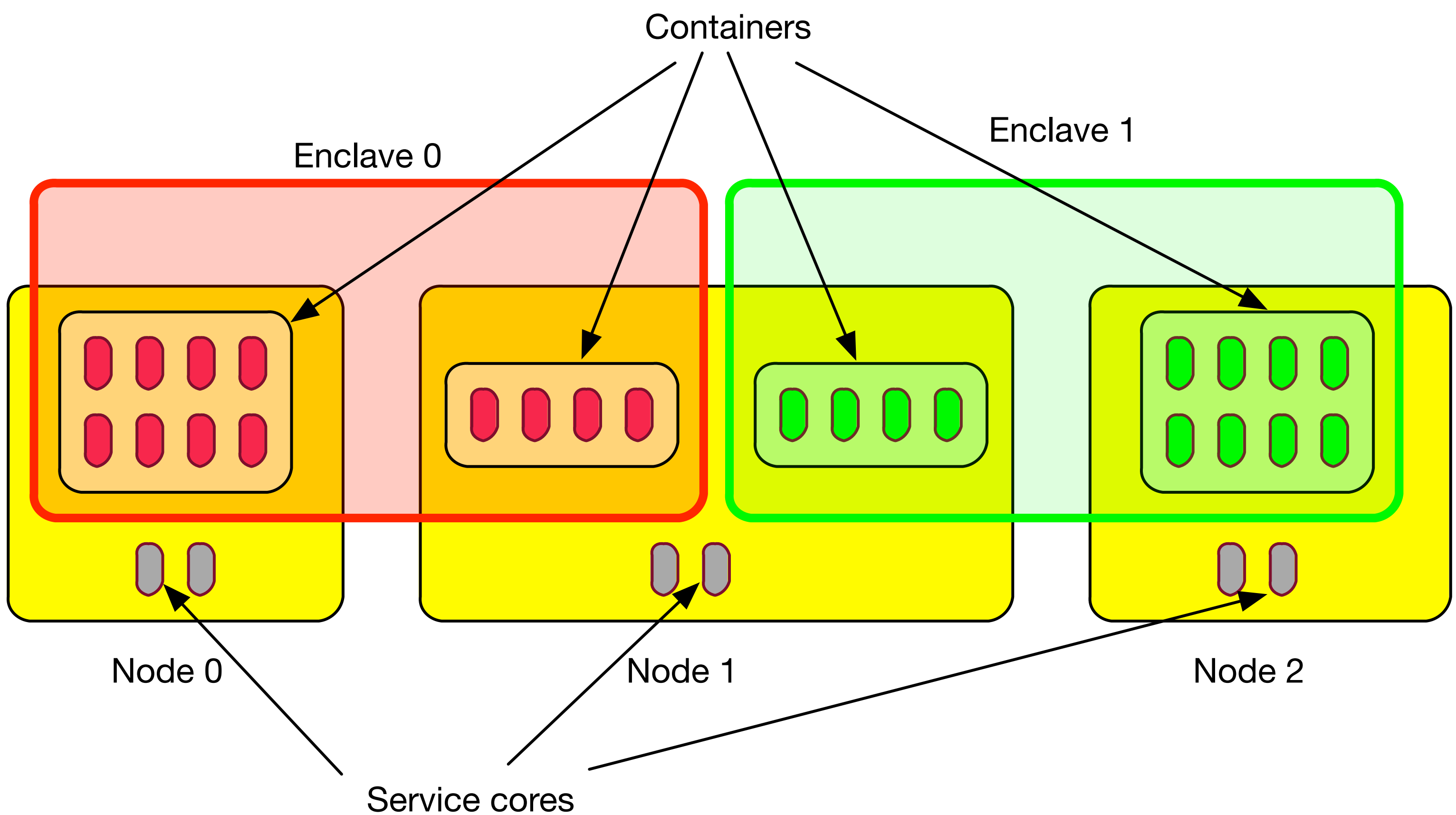
Future Node


power-efficient cores
“fat” cores
accelerators
Network Interface
Network Interface
cache/scratchpad
cache/scratchpad
cache/scratchpad
cache/scratchpad
Fast DRAM
Slow DRAM
NVRAM

ARCHITECTURE

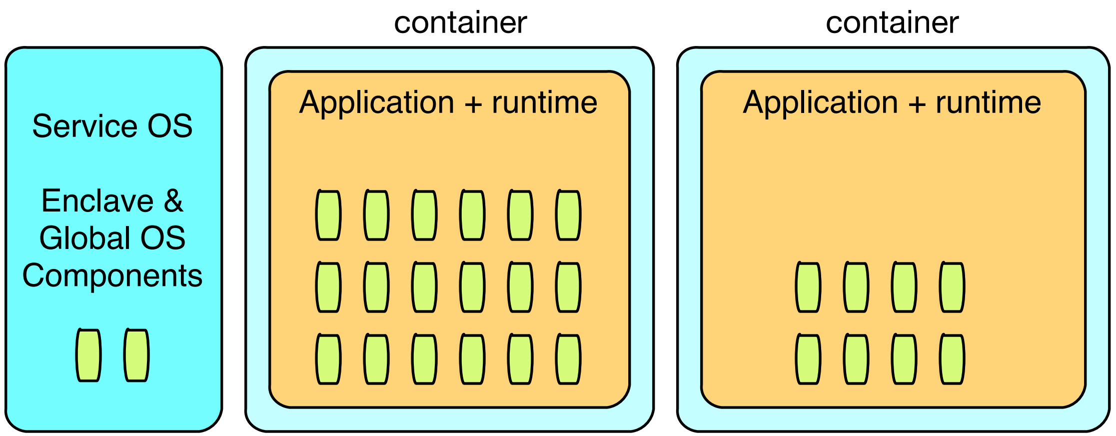
Key Constructs

Key Constructs



Container

- Resources at a node are partitioned into one or more **containers.** Each container is dedicated to one parallel application
- Resources allocated to a container are managed by the container runtime (possibly different on each container)
- Additional core(s) provide general OS services and control the containers – *containers are free of OS noise*



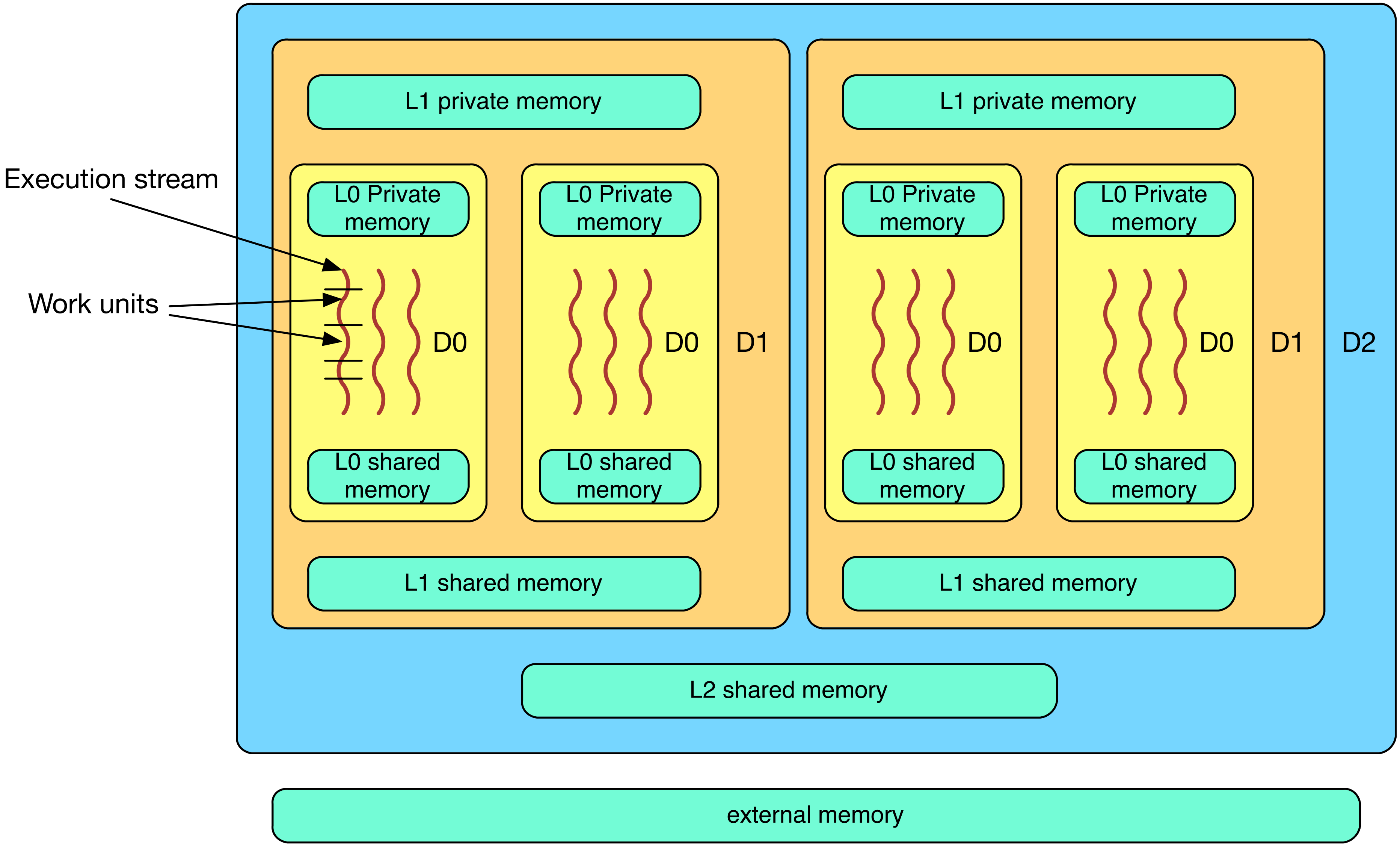
Containers -- Issues

- How to implement:
  - Linux container (using OS and architecture support)
  - Virtual machine
  - Fused OS
  - Some combination
- Metrics
  - Performance isolation
  - Fault containment
  - Reconfiguration overhead
  - Steady-state overhead
  - Ease of customization
  - Mechanisms for communication across containers
  - Ease of implementation and support

Container Run-Time

- Specialized “machine looking” runtime for compute containers
  - Programming model specific runtimes implemented atop core runtime
- Goals:
  - Efficient support for task model (light-weight threads and tasks)
  - Reduced scheduling overhead
  - Improved user-space event handling
  - Customization – customized schedulers
  - Runtime memory management (allocation, copying, caching)
  - Runtime power management

Container Run-Time (1)

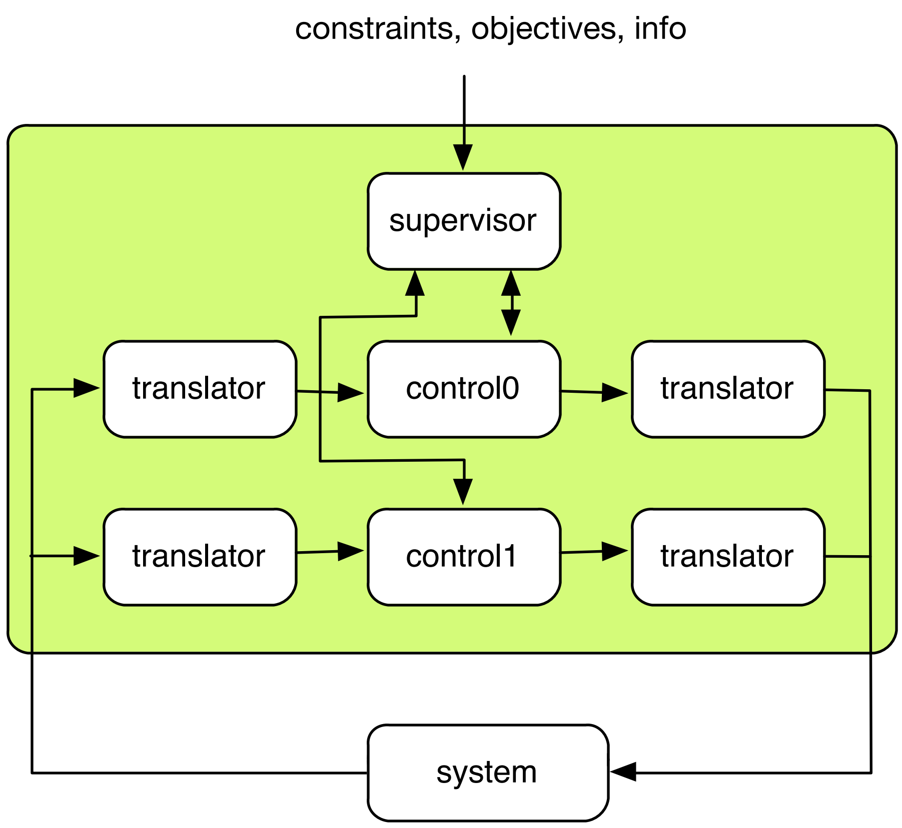


Global OS & Enclave OS/R

- Global OS Allocates (dynamically) resources to *enclaves*
  - Nodes, power, I/O services, switch bandwidth (?)
  - Allocation can be hierarchical
- Enclave OS/R configures the enclave resources and maps logical entities to physical resources
- Composed applications may involve multiple enclaves
  - Global OS controls enclave connectivity and time-space scheduling of enclaves
  - APIs are provided for parallel connectors

Generic Resource Manager

- Feedback loop for each managed resource
  - sensors: application progress and HW monitoring
- Supervisor ensures coordinated resource management
  - higher-level specification of constraints, goals and information on execution software
  - Use of ML to adjust feedback functions



Resilience

- OS services for application resilience
  - hardened data structures, hardened/persistent storage, hardened execution
- Hierarchical error-handling: Error reported to lowest level (container or enclave if node failed). Error is handled at that level or raised to next level of the hierarchy
  - Managing levels of confidence

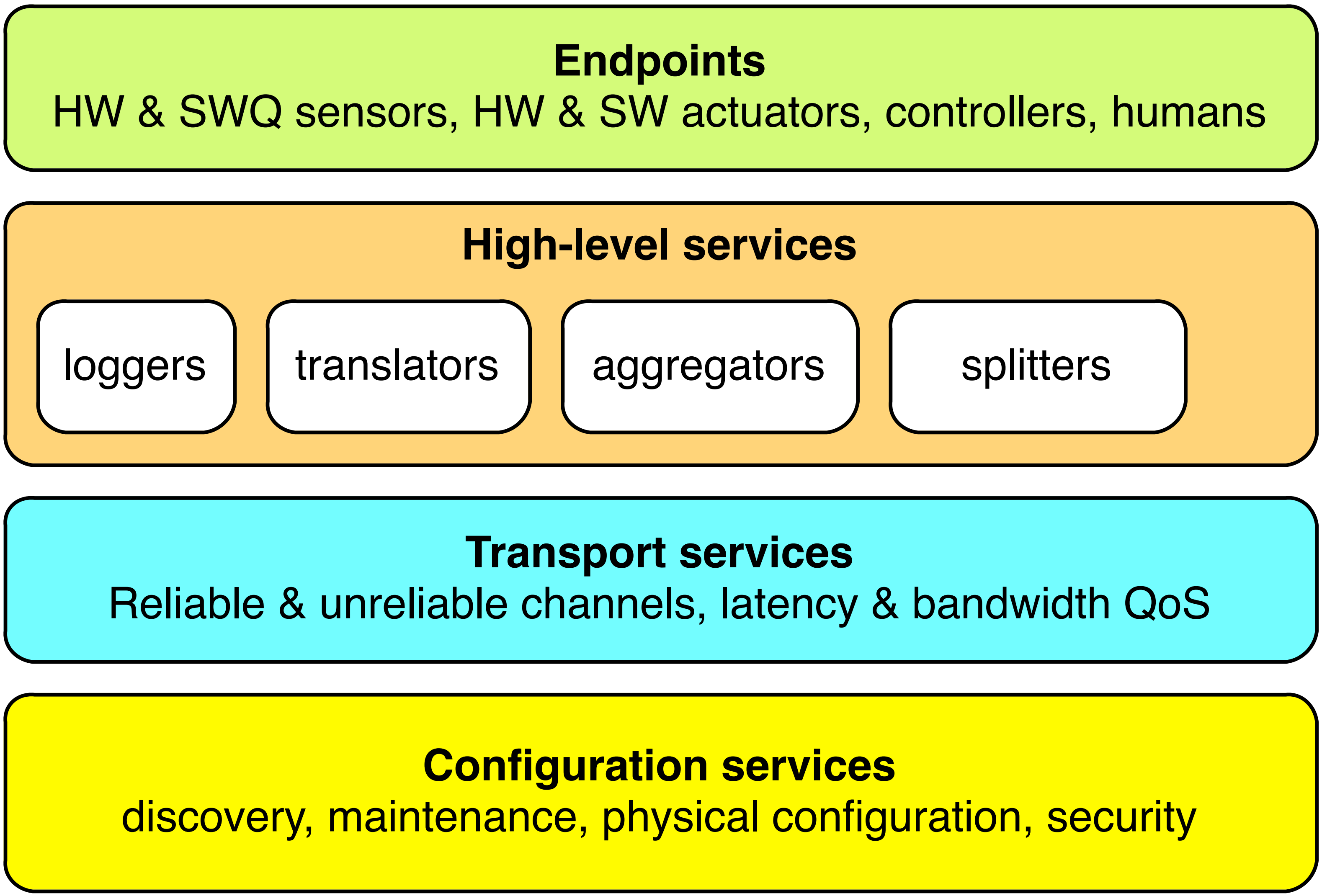
Global Information Bus

- Franck Cappello, Allen Malony (Argo)
- Karsten Schwan, Philip C. Roth (Hobbes)

- Problem: Moving information from sensors to controllers and from controllers to actuators
  - Physical structure of system many not match logical structure of containers and enclaves.
  - In-band vs. out-of-band
  - Non-coherent information
  - Shelf-life constraints

- Solution: Global publish-subscribe system
  - Configured to ensure locality of communication and robustness

Global Information Bus (cont.)



Current Candidate Use Cases

- CTH+Paraview/Catalyst “in transit” analysis
  - From Kevin Pedretti, SNL
- DNS+LES combustion high-/low-fidelity verification
- XGC-1+XGC-a coupled fusion high-/low-fidelity modeling
  - From Hasan Abbasi, ORNL
- CASL VERA coupled multiphysics modeling
  - From John Turner via Barney Maccabe, ORNL
- SACLA and K Computer data analysis
  - From Atsushi Hori, RIKEN, via Franck Cappello, ANL (BDEC’14)
- HACC simulation / analysis /visualization workflow
  - From Salman Habib via Franck Cappello, ANL (BDEC’14)

An Exascale Operating System and Runtime Research Project

Pee Beckman
Argonne National Laboratory

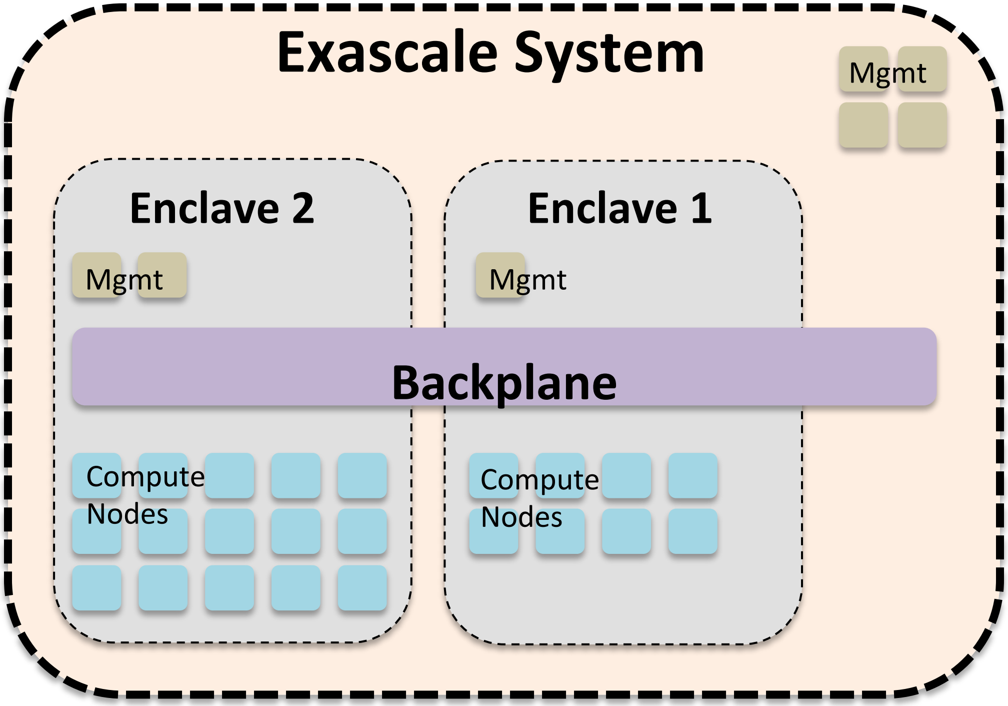
U.S. DEPARTMENT OF ENERGY

The Argo Team:

- **ANL**: Pete Beckman, Marc Snir, Pavan Balaji, Rinku Gupta, Kamil Iskra, Franck Cappello, Rajeev Thakur, Kazutomo Yoshii
- **BU**: Jonathan Appavoo, Orran Krieger
- **LLNL**: Maya Gokhale, Edgar Leon, Barry Rountree, Martin Schulz, Brian Van Essen
- **PNNL**: Sriram Krishnamoorthy, Roberto Gioiosa
- **UC**: Henry Hoffmann
- **UIUC**: Laxmikant Kale, Eric Bohm, Ramprasad Venkataraman
- **UO**: Allen Malony, Sameer Shende, Kevin Huck
- **UTK**: Jack Dongarra, George Bosilca, Thomas Herault
- Industry Advisory Committee:
  - Michael Schulte (AMD), Eric Van Hensbergen (ARM), Larry Kaplan (CRAY),
  - Kyung Ryu (IBM), Bob Wisniewski (Intel), Don Becker (NVIDIA)

Argo Key Research Areas

- Node OSR
  - Kernels, memory, and HW
- Concurrency
  - Lightweight thread/task runtime
- Backplane
  - Event, Control, and Performance
- Global View
  - Optimization & Goal-based management



Hobbes Team

Institution	Person	Role	PO Approval Date
Georgia Institute of Technology	Karsten Schwan	PI	Feb 6, 2014
Indiana University	Thomas Sterling	PI	Nov 22, 2013
Los Alamos National Lab	Mike Lang	PI	
Lawrence Berkeley National Lab	Costin Iancu	PI	
North Carolina State University	Frank Mueller	PI	Oct 24, 2013
Northwestern University	Peter Dinda	PI	Nov 25, 2013
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	Oct 21, 2013
University of California – Berkeley	Eric Brewer	PI	
University of New Mexico	Patrick Bridges	PI	Oct 2, 2013
University of Pittsburgh	Jack Lange	PI	Mar 19, 2014

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
  - Enable high-risk high-impact research in virtualization, energy/power, scheduling, and resilience

Exploring Adaptive Resource Centric Computing for Exascale with Tessellation (X-ARCC)

Steven Hofmeyr — LBNL

John Kubiatowicz — UC Berkeley

April 16 2014

Key Features of ARCC

Need to reason predictably about resources:

- Cells: lightweight containers with user-level access to guaranteed resources (cores, memory, bw, etc.)
- Services in cells provide QoS-guaranteed access to hardware, e.g. network, block device

Ensure maximum utilization:

- Customizable runtimes with minimal OS interference: user-level scheduling & memory management, etc.
- Space-time partitioning for maximum flexibility of resource allocation
- Gang-scheduling for performance predictability

BERKELEY LAB

SWARMLAB UC BERKELEY