Argonne NATIONAL LABORATORY

Update on MPI and OS/R Activities at Argonne

Rajeev Thakur

Deputy Director

Mathematics and Computer Science Division

Argonne National Laboratory

Joint work with Pavan Balaji, Pete Beckman, Bill Gropp, Rusty Lusk, Marc Snir, and many others

U.S. DEPARTMENT OF ENERGY

Outline

- Current status of MPI running on the largest systems in the world
- Current status of production MPI applications running on these systems
- Current efforts in our group on MPI and runtime
- New exascale operating system and runtime research project Argo

Current Status of MPI on the Largest Systems

- The vast majority of production parallel scientific applications today use MPI
  - Increasing number use (MPI + OpenMP) hybrid
  - Several exploring (MPI + accelerator) hybrid

- Today's largest systems in terms of number of regular cores (excluding GPU cores)

Tianhe-2 (China)	3,120,000 cores
Sequoia (LLNL)	1,572,864 cores
Mira (ANL)	786,432 cores
K computer	705,024 cores
Jülich BG/Q	393,216 cores
Blue Waters	386,816 cores
Titan (ORNL)	299,008 cores

- **MPI already runs on production systems with up to 3 million cores**

Application Successes

- All the high-end systems in the world are reporting success in running real applications at full scale
  - Sequoia, Blue Waters, Titan, Mira, K computer, …
- A few examples follow…
- Last year, IBM reported to have successfully scaled the LAMMPS application to over 3 million MPI ranks
- Several applications reported to run at full scale on LLNL's Sequoia (1.6 million cores) and achieving 12 to 14 petaflops of sustained performance

HACC Cosmology Code

- HACC cosmology code from Argonne (Salman Habib) achieved **14 petaflops** on Sequoia
  - Ran on full Sequoia system using MPI + OpenMP hybrid
  - Used 16 MPI ranks * 4 OpenMP threads on each node, which matches the hardware architecture: 16 cores per node with 4 hardware threads each
  - **~ 6.3 million way concurrency: 1,572,864 MPI ranks * 4 threads/rank**
  - http://www.hpcwire.com/hpcwire/2012-11-29/sequoia_supercomputer_runs_cosmology_code_at_14_petaflops.html
  - SC12 Gordon Bell prize finalist

Cardioid Cardiac Modeling Code

- Cardioid cardiac modeling code (IBM & LLNL) achieved **12 petaflops** on Sequoia
  - Models a beating human heart at near-cellular resolution
  - Ran at scale on full system (96 racks)
  - Used MPI + threads hybrid: 1 MPI rank per node and 64 threads
  - OpenMP was used for thread creation only; all other thread choreography and synchronization used custom code, not OpenMP pragmas
  - http://nnsa.energy.gov/mediaroom/pressreleases/sequoia112812
  - SC12 Gordon Bell Prize finalist

ROSS Parallel Discrete-Event Simulator

- ROSS parallel discrete-event simulator run with 7.8 million MPI ranks on a Sequoia/Vulcan combined system
  - PI: Chris Carothers, Rensselaer Polytechnic Institute
  - Used 96 racks of Sequoia + 24 racks Vulcan (a BG/Q system at LLNL for industrial collaborators)
  - 1,966,080 cores * 4 MPI ranks/core = 7,864,320 MPI ranks
  - Ran the PHOLD benchmark and achieved the highest event rate ever reported by a parallel discrete event simulation (504 billion events/sec)
  - Paper: "*Warp Speed: Executing Time Warp on 1,966,080 Cores*," PADS 2013
  - News: http://news.rpi.edu/luwakkey/3173?destination=node/40108

- And there are other applications running at similar scales…

Experiments with over 100 million MPI processes

- Prof. Alan Wagner's group at Univ. of British Columbia has run programs with over 100 million MPI processes (on 6,480 processor cores)

- They use a modified version of MPICH they develop called FG-MPI (Fine-Grain MPI) that implements MPI ranks as coroutines rather than operating system processes

- https://www.westgrid.ca/westgrid_news/2013-01-14/ubc_researchers_use_westgrid_explore_exascale_computing

High-Level Goals of our MPI Project

- Existing MPI applications developed over several years represent billions of dollars worth of investment
- As we progress from today's petascale to future exascale systems, MPI must evolve to run as efficiently as possible on these systems so that applications can continue to gain the performance benefits
- This requires that both the MPI standard as well as MPI implementations address the challenges posed by the architectural features, limitations, and constraints expected in future extreme-scale systems
  - E.g., lower memory per core, higher thread concurrency, power constraints, performance scalability, resilience to failures, …

Specific Goals of our MPI Project (1)

- MPI Standardization
  - Work with the MPI Forum to ensure that the MPI *specification* evolves to meet the needs of future systems and of applications, libraries, and higher-level languages
- MPI Implementation
  - Continued enhancement of the MPICH implementation of MPI to support the new features in the MPI standard (MPI-3 and beyond) and to address the implementation challenges posed by exascale systems
- Transfer Technology to Vendors
  - Continue to collaborate with our vendor partners (IBM, Cray, Intel, …) to enable them to make the latest MPI features available to users on production systems

Specific Goals of our MPI Project (2)

- Going Beyond Current MPI
  - Investigate new programming approaches for potential inclusion in future versions of the MPI standard
    - E.g., generalized user-defined callbacks, lightweight tasking, extensions for heterogeneous computing systems and accelerators, …
- Efficient Runtime for Implementing Higher-level Programming Models
  - Dynamic execution environments (e.g., Charm++, ADLB)
  - Global communication models (e.g., PGAS models, Global Arrays, GVR)
- Interoperability with other Programming Models

MPI Standardization

- The MPI Forum released the MPI-3 standard in Sept 2012 after about three years of effort
- Several of us were active in the definition of MPI-3
  - RMA, hybrid, fault tolerance, and tools working groups
- The MPI Forum continues to meet every three months to define future versions of MPI (MPI 3.x, MPI 4.0), and we continue to be actively involved in those efforts
  - (There was an MPI Forum meeting just last week in the Bay Area)

MPI Standard Timeline

- MPI-1 (1994), presented at SC'93
  - Basic point-to-point communication, collectives, datatypes, etc
- MPI-2 (1997)
  - Added parallel I/O, RMA, dynamic processes, thread support, C++ bindings, …

- ---- Stable for 10 years ----

- MPI-2.1 (2008)
  - Minor clarifications and bug fixes to MPI-2
- MPI-2.2 (2009)
  - Small updates and additions to MPI 2.1
- MPI-3 (2012)
  - Major new features and additions to MPI

Overview of New Features in MPI-3

- Major new features
  - Nonblocking collectives
  - Neighborhood collectives
  - Improved one-sided communication interface
  - Tools interface
  - Fortran 2008 bindings
- Other new features
  - Matching Probe and Recv for thread-safe probe and receive
  - Noncollective communicator creation function
  - "const" correct C bindings
  - Comm_split_type function
  - Nonblocking Comm_dup
  - Type_create_hindexed_block function
- C++ bindings removed. Users can use C bindings from C++ programs.
- Previously deprecated functions removed

MPI Implementation

- The MPI implementation that we develop, MPICH, has always closely tracked the evolving MPI standard since the beginning of MPI
- For MPI-3, we set an aggressive goal of implementing all of MPI-3 by SC12, i.e., ***barely a month and a half after the standard was released***
- Thanks to the heroic efforts of our project members, we were successful in releasing at the MPICH BoF at SC12 an all-new version of MPICH (3.0) that supports all of MPI-3
- We also unveiled a new, redesigned web site for MPICH, www.mpich.org
- Although this version of MPICH supports all of MPI-3, performance tuning is needed in many parts, which we continually work on

Vendor Interactions

- Continue to collaborate with our vendor partners to give them a running start toward supporting the latest MPI features on their platforms

- IBM has merged its separate MPI implementation efforts for the Blue Gene and POWER platforms into a single implementation based on MPICH
  - We have been working with them closely and share a common code base. They send us code patches that we incorporate.

- Similar interactions with Cray on MPI for the Cray systems and with Intel for MPI on Intel platforms

- As a result, the majority of the largest machines in the Top500 run MPICH
  - ***Eight of the top ten machines in the Nov. 2012 Top500 list use MPICH exclusively***
  - We are working with Fujitsu and the University of Tokyo to port MPICH to the K computer
  - Tianhe-2 and Blue Waters also run MPICH

One-Sided Communication

- MPI-3 has added significant new features for one-sided communication, which make it useful for implementing high-level programming models, libraries, and applications

- In addition to supporting all these new features in MPICH 3.0, we have published papers on how to implement them efficiently and on using MPI for shared memory programming within a node (MPI+MPI)
  - "Leveraging MPI's One-Sided Communication Interface for Shared-Memory Programming", EuroMPI 2012
  - "MPI+MPI: A New, Hybrid Approach to Parallel Programming with MPI Plus Shared Memory Computing", *Computing* (journal), to appear
  - "An Implementation and Evaluation of the MPI 3.0 One-Sided Communication Interface," submitted to *Concurrency and Computation: Practice and Experience*

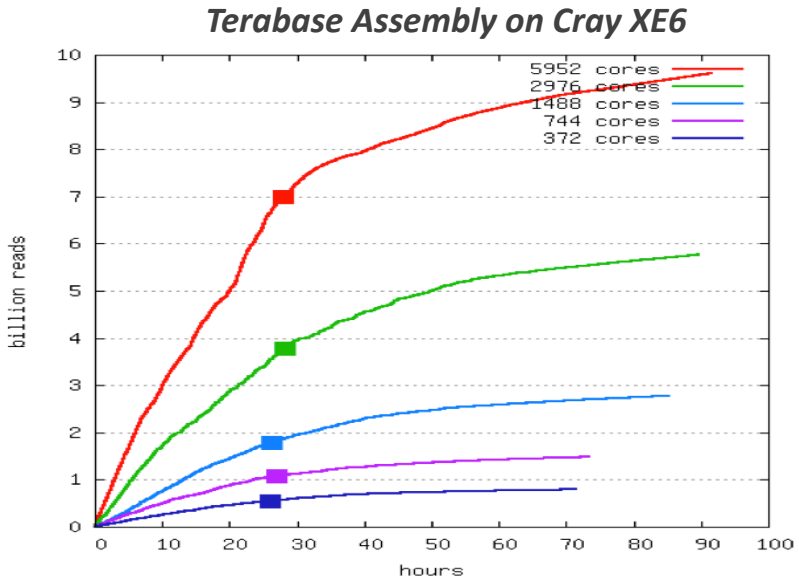
Active Messages in MPI

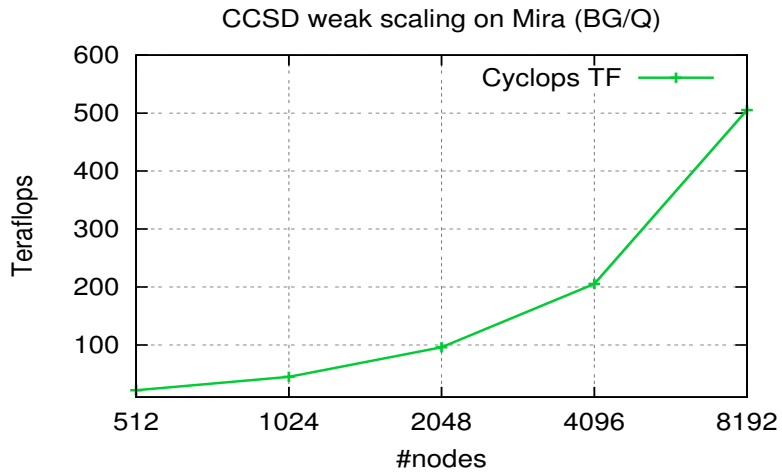
- MPI does not directly support active messages
- Nonetheless, they are useful for implementing higher-level programming models, such as PGAS and Charm++
- Together with Xin Zhao and Bill Gropp at UIUC, we investigated approaches for supporting active messages within the context of MPI
- This work was accepted for publication at CCGrid 2013
  - "Towards Asynchronous, MPI-Interoperable Active Messages", CCGrid 2013

Tools Interface

- The new MPI-3 tools interface, commonly known as the "MPI_T" interface, has been completely implemented in MPICH
- All internal configuration in MPICH that was previously controlled exclusively by UNIX environment variables is now also accessible programmatically through MPI_T control variables
  - E.g., thresholds for selecting different algorithms for collective communication as well as options for runtime debugging
- Through MPI_T, we have also exposed several new performance variables, primarily vending statistics from MPI message-matching queues and low-level communication data structures
- We have used these new performance variables to great effect for studying the performance of NAMD/Charm++ implemented over MPI
  - a publication on this work is under preparation

Applications





- **Terascale Genome Assembly**
  - Graph assembly problem that deals with finding an Hamiltonian path in a fuzzy graph with erroneous edges (or non-edges)
  - Highly communication intensive, with (seemingly) random global communication
  - Genome assembly at this scale (2.3TB on Cray XE6) has, for the first time, allowed scientists to study multiorganism genome colonies of completely or partially unknown species
- **Cyclops Tensor Framework (Chemistry)**
  - Fundamental component of quantum chemistry for coupled-cluster methods
  - Supersedes existing algorithms and software for parallel tensor contractions
  - Enabled quantum simulations of 250 electrons in 1000 orbitals (no point-group symmetry) on Argonne Mira
    - Order of magnitude larger scale problem than anything that has been previously done

Other Projects in our Group (1)

- Pavan Balaji's DOE Early Career Award
  - "Exploring Efficient Data Movement Strategies for Exascale Systems with Heterogeneous Memory"
- "Analysis of Topology-Dependent MPI Performance on Gemini Networks"
  - EuroMPI 2013 paper
- "Enabling MPI Interoperability Through Flexible Communication Endpoints"
  - EuroMPI 2013 paper
- "Analysis and Acceleration of an Asynchronous Message-driven Runtime System over MPI on IBM Blue Gene/Q and Cray XE6"
  - Under preparation

Other Projects in our Group (2)

- Several recent papers on integrating MPI and GPU communication
  - "On the Efficacy of GPU-Integrated MPI for Scientific Applications," HPDC 2013
  - "Synchronization and Ordering Semantics in Hybrid MPI+GPU Programming," AsHES Workshop at IPDPS 2013
  - "Enabling Fast, Noncontiguous GPU Data Movement in Hybrid MPI+GPU Environments," Cluster 2012
  - "MPI-ACC: An Integrated and Extensible Approach to Data Movement in Accelerator-Based Systems," HPCC 2012

- Virtualizing GPUs
  - "pVOCL: Power-Aware Dynamic Placement and Migration in Virtualized GPU Environments," ICDCS 2013
  - "VOCL: An Optimized Environment for Transparent Virtualiza- tion of Graphics Processing Units," InPar 2012

New Exascale OS/R Research (Argo Project)

Introduction

- Earlier this year, DOE issued a call for proposals for research and development of a prototype addressing the challenges of Exascale Operating and Runtime Systems
- Four key challenges that proposals must address
  - Energy efficient hardware and software design constrained to a power envelope of 20 Megawatts (MW)
  - Managing unprecedented parallelism, especially at the node level of exascale systems
  - Managing deep memory hierarchies and multi-stage storage systems under severe energy constraints
  - Resilience in the presence of predicted high rates of hard and soft faults

Introduction contd.

- In addition, the proposed research must address all of the following seven classes of capabilities
  - *Power management*
  - *Support for dynamic programming environments*
  - *Programmability and tuning support*
  - *Resilience*
  - *Heterogeneity*
  - *Memory management*
  - *Global optimization*

Exascale OS/R

- Two proposals have been selected for funding
- One led by Argonne, called **Argo**
  - PI: Pete Beckman
  - Partners include LLNL, PNNL, UTK, Univ of Oregon, UChicago, Boston Univ
- Another led by Sandia, called **Hobbes**
  - PI: Ron Brightwell
  - Partners include ORNL, LBNL, Indiana Univ, and others
- Three-year projects starting August 2013

Jason's ship: Argo

In Greek mythology, Argo (which means "swift") was the ship used by Jason in his quest for the Golden Fleece

Argo Project Participants

PI: Pete Beckman, ANL

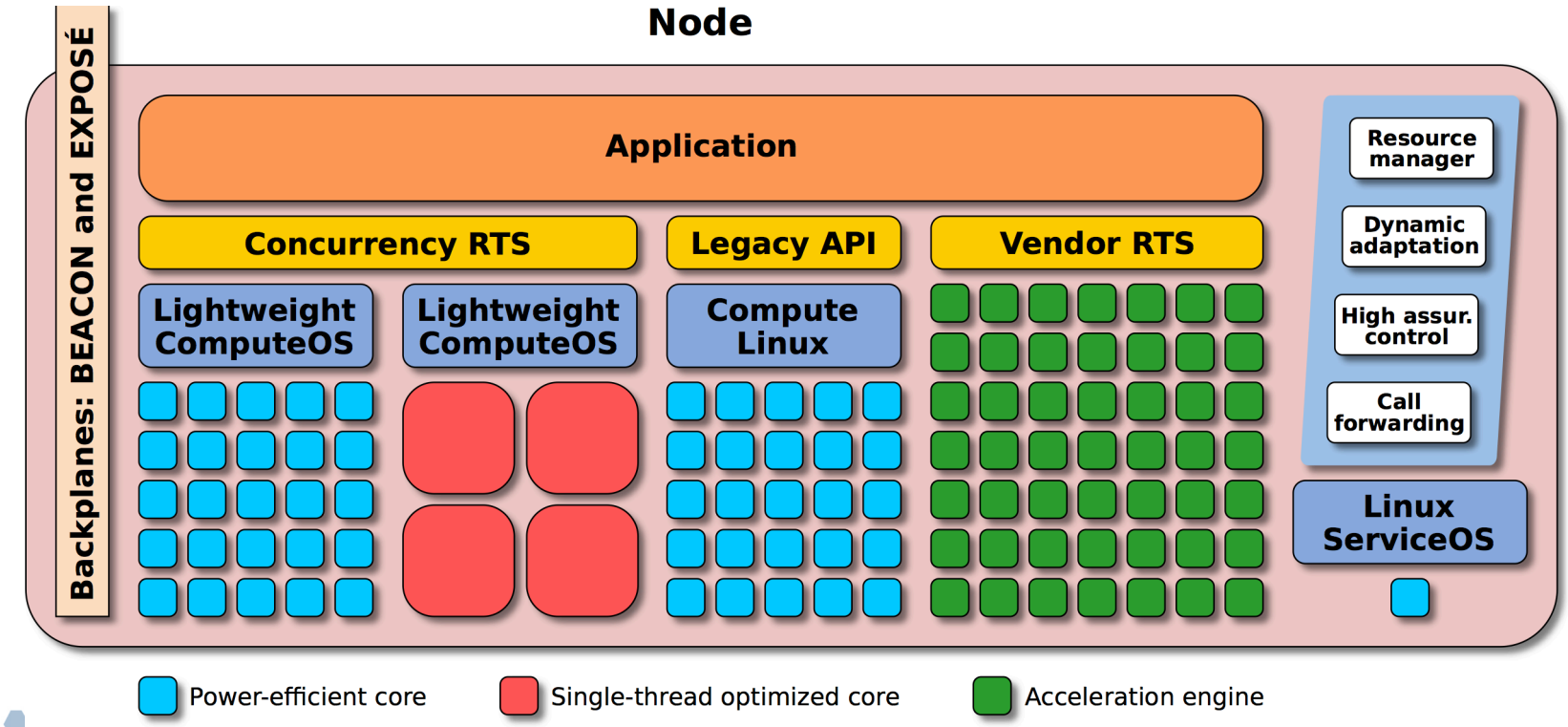
ANL:	Marc Snir, Pavan Balaji, Rinku Gupta, Kamil Iskra, Rajeev Thakur, Kazutomo Yoshii
BU:	Jonathan Appavoo, Orran Krieger
LLNL:	Maya Gokhale, Edgar Leon, Barry Rountree, Martin Schulz, Brian Van Essen
PNNL:	Sriram Krishnamoorthy, Roberto Gioiosa, David Callahan
UC:	Henry Hoffmann
UIUC:	Laxmikant Kale, Eric Bohm, Ramprasad Venkataraman
UO:	Allen Malony, Sameer Shende, Kevin Huck
UTK:	Jack Dongarra, George Bosilca

Argo Project Components

- Node level OS/R
- Runtime System for Massive Concurrency
- Global Optimizations
- Event and Control Backplane
- Goal-Based Introspective Autonomic Optimizer

Node Level OS/R

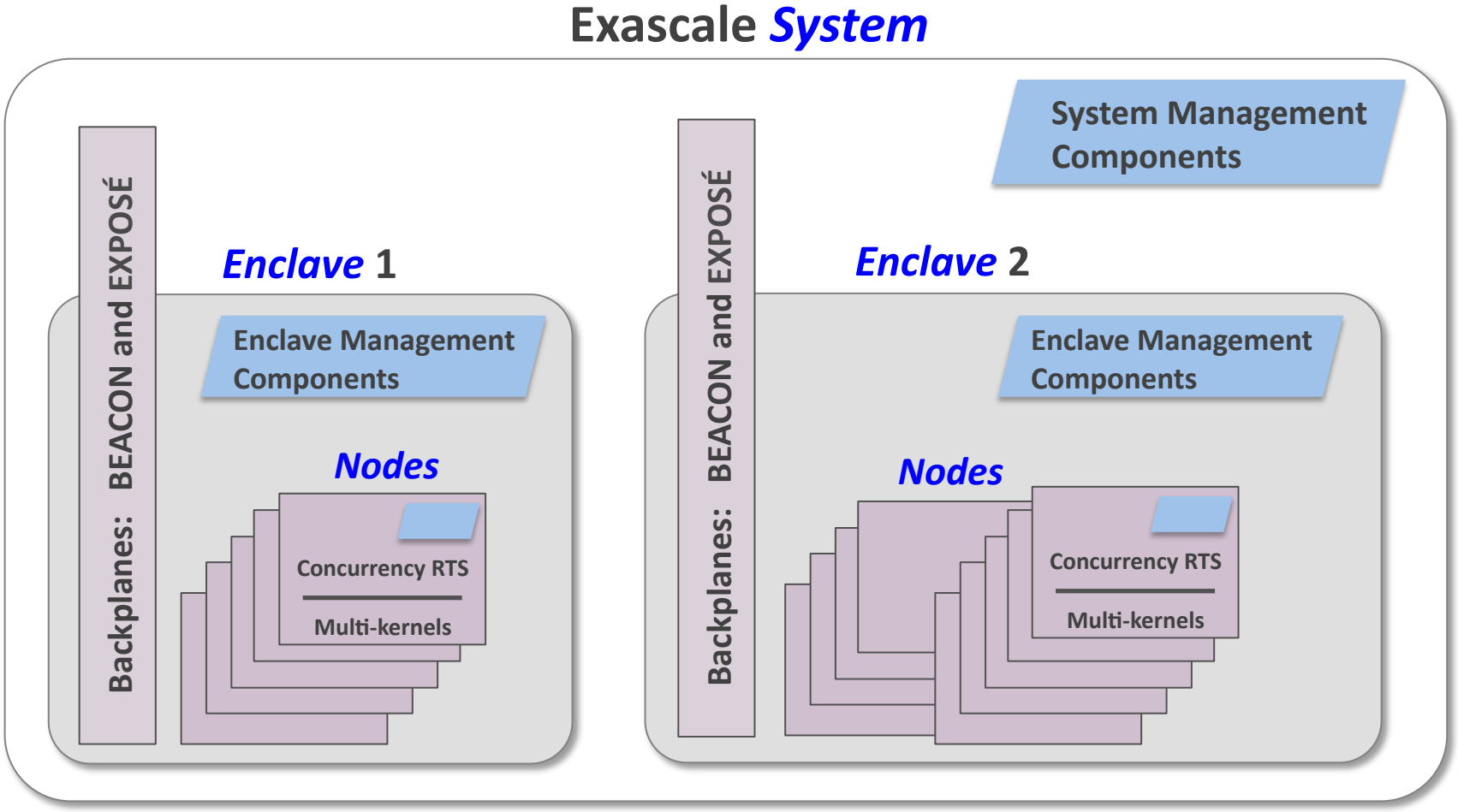
- OS specialization: run multiple different OS kernels simultaneously for different types of cores and workloads
- Advanced memory management for different memory types
- ANL, LLNL, PNNL, BU



Runtime System for Massive Concurrency

- Lightweight Threads and Tasks
  - Stackable Scheduler with Pluggable Strategies
  - Pluggable Task Graph Engine
- Data Movement for Dynamic Execution Environments
  - Active Task Invocation and Memory-Driven Active Messages
  - Dynamic Cache and Memory Access for Put/Get Communication
- ANL, UIUC, UTK, PNNL

Global Optimizations



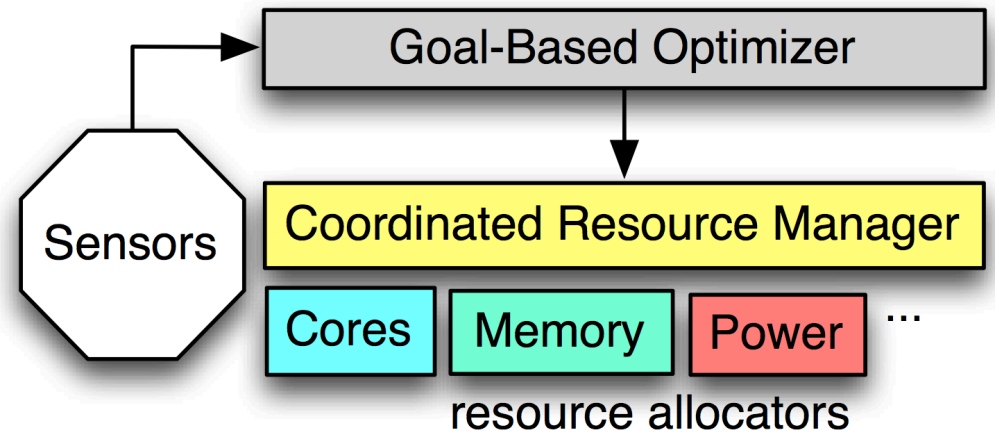
Enclave services, System services, Cross-cutting services

Event and Control Backplane

- Backplane for event and control notification (BEACON)
  - UO and LLNL
- Dynamic Performance Observation and Introspection (EXPOSÉ)
  - UO and LLNL

Goal-Based Introspective Autonomic Optimizer

- Led by Hank Hoffmann, Univ of Chicago
- Based on the SElf-awarE Computing (SEEC) model developed by Hank
- SEEC is designed to address the challenge of programming modern and future computer systems that must meet conflicting goals (e.g. high performance with low energy consumption)



We are hiring!

- We have funding in both MPI and Argo projects to hire postdocs, students, or programmers

- Talk to me if you are interested