The Argo Global Operating System:

Information and Resource Management Services Across an Exascale Machine

Swann Perarnau (ANL) and Martin Schulz (LLNL)

Argo GlobalOS Team

- **Argonne National Laboratory (ANL)**
  - Rajeev Thakur
  - Franck Cappello
  - Rinku Gupta
  - Swann Perarnau
- **Lawrence Livermore National Laboratory (LLNL)**
  - Martin Schulz
  - Tapasya Patki
  - Barry Rountree
- **University of Oregon**
  - Al Malony
  - Sameer Shende
  - Daniel Ellsworth
- **University of Chicago**
  - Hank Hoffman

Our Approach

Problem

Lack of infrastructure to share information and manage resources across entire machine.

Solution

Create a global operating system, as a collection of distributed services to:

- Provide new abstractions of the machine,
- Share information across software layers,
- Manage node configuration, power, failures, etc

A New Abstraction: Enclaves

Problem

At exascale, nodes and jobs are poor units of control over the system.

Solution

Introduce a new abstraction: groups of nodes configured and managed as a single entity.

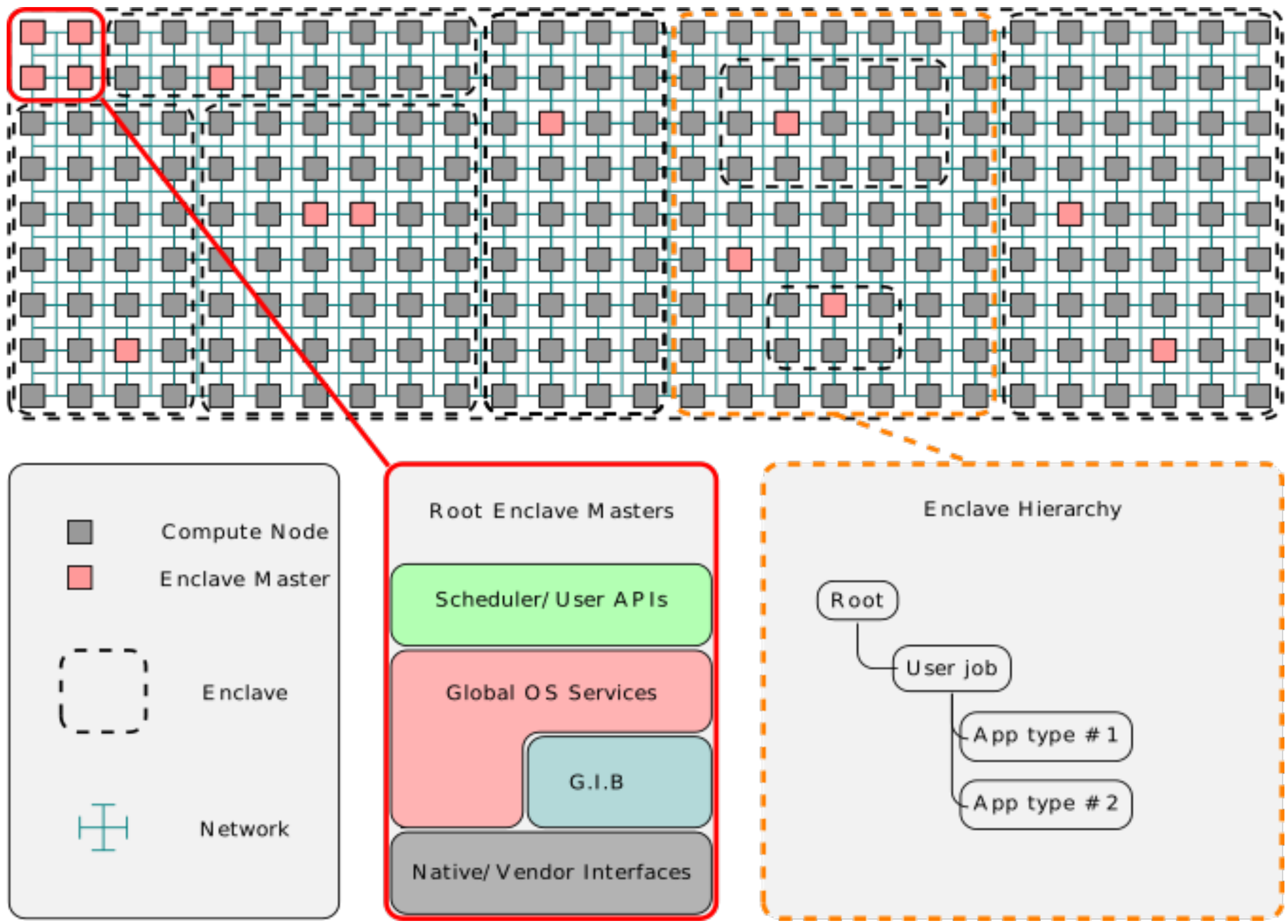
Enclaves are:

- Dynamic: nodes can move in and out of enclaves
- Recursive: enclaves can be subdivided

Enclave have:

- Dedicated Resources: master nodes only run system services
- A Hierarchy: root contains all resources, child enclaves are partitions of the system
- Distributed Control: master nodes are responsible for the management of their enclave

Enclaves Across the Machine



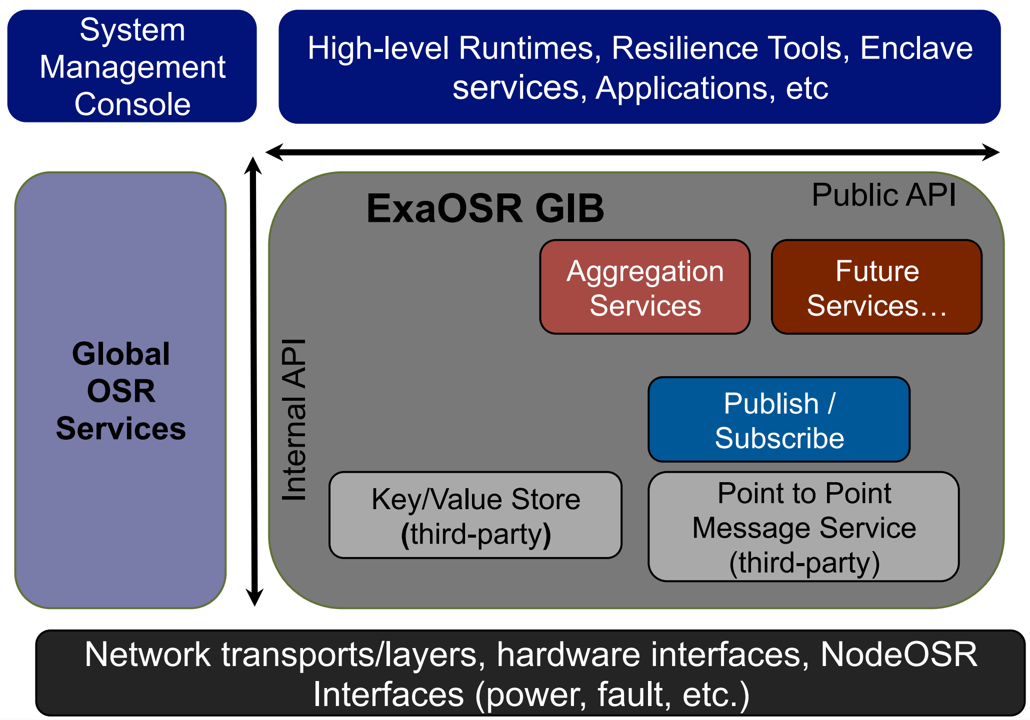
The Global Information Bus

Problem

Lack of infrastructure to share information across software layers on future architectures

Solution

Build a Global Information Bus framework with components such as a lightweight framework for sharing information across ARGO software layers supporting both event and control notification (BEACON) for introspection (faults, performance, energy), in situ analysis, and feedback (EXPOSÉ)



BEACON for Sharing Data in Argo

Design considerations

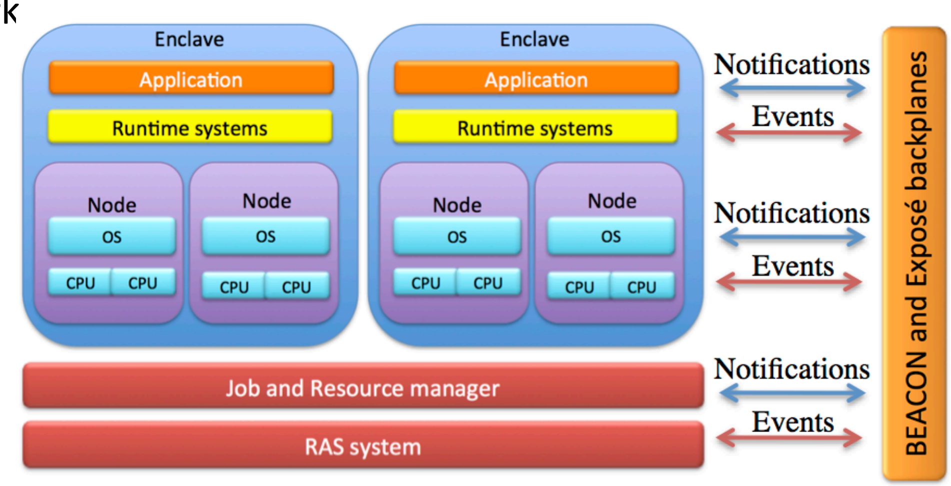
An Argo enclave contains application, runtime state, and dynamics. It is important to observe and share information about these aspects

- How do we enable introspection of performance and state within an enclave?
- How can information sharing and transport be made accessible to all layers? Information about the global state (performance, power, faults) of an exascale system is distributed across the machine and must be made accessible if it is to be actuated upon

Solution

Build a distributed system-wide *BEACON (Backplane for Event and Control Notification)*

- publish/subscribe (pub/sub) framework
- enclave-wide information access



BEACON for Sharing Data in Argo

Results

- BEACON: a pub/sub system responding to the needs (designed to be scalable, portable, resilient, fast –in memory-) and constraints (low overhead, low memory footprint) of HPC systems
- BEACON API allows to share data across different software on the systems and different layers of the HPC stack
- BEACON has been integrated with different components of the ARGO system (see demo for how BEACON works with EXPOSÉ to implement global power management)
- BEACON 1.0 API available
- BEACON implementations
  - Version 1.0 using TCP/IP networking protocol
  - Version 1.1 using Inotify (RAMDISK based I/O)
  - Version 1.2 using EVPATH (ongoing/under development)

Impact

- A central software piece in ARGO (see the demo on power management)
- A critical software piece that allowed prototyping other services (EXPOSE, etc.)

Future Work

- Explore BEACON implementation on AMQP
- Explore additional challenges and extensions: authentication, security, event logger, response management decision making

EXPOSÉ for In Situ Data Introspection and Analytics

Design considerations

- Fast, scalable, resilient system data introspection (aggregation, reduction/filtering, interpretation, and decision control) is critical for exascale system management and control
- A generic framework is needed for developing, integrating and running different data Introspection services
- Need to be “in situ”

Solution

The *EXPOSÉ (Exascale Performance and Observability Environment)* is a framework for:

- developing in situ data introspection and analysis services
- EXPOSÉ interfaces to BEACON for retrieving data on subscribed topics of interest and publishing results, control, and other data back to consumers
- Different services can be developed for specific purposes within the exascale environment, depending on the requirements of the problem

Results

- The EXPOSÉ service concept has been demonstrated in the global power management work
- Power data aggregation and processing occurs in the ERM and GRM as an EXPOSÉ service
- Introspection of node-level, enclave-level, and global performance data (captured by the TAU Performance System), its runtime analysis, and its visualization has been demonstrated
- Work is in progress on using the EVPath technology to implement the EXPOSÉ framework as service overlays and to align it with Hobbes activities

Impact

BEACON + EXPOSÉ will make it possible to develop scalable distributed services in exascale systems that are efficient, dynamic, retargetable, and responsive.

Enclave and Resource Management

Problem

- Lack of infrastructure to manage resources at scale
- Need to experiment with new policies and ideas

Solution

- Provide enclaves management across the system
- Build collection of services on top of enclave management to deal with power, failures, node configuration, workflow support, etc

Approach

- Use the Global Information Bus for transport, low-level data management (key-value store)
- Reuse industry-proven open-source software (provisioning, node management)
- Make new services easy to create and integrate

Tracking Enclaves

Problem

- Maintain information on enclaves, their nodes, the hierarchy
- Provide easy access to this information to all other services
- Allow extensions (new services) to add information to enclaves

Solution

- Service built on top of key-value store
- Enclave/node UUIDs mapped to a list of arbitrary key-values
- New enclave inherits the list of its parent in the tree

Results

- Successfully deployed on Chameleon to track 20 nodes
- Used as part of integration demo

Future Work

- Scalability evaluation
- Limit modification of enclave information to enclave masters

Orchestration

Problem

- Boot and provision nodes based on enclave information
- Update enclave information from outside sources
- Provide interface between external services (batch scheduler) and GlobalOS

Solution

- Collection of clients to vendor and GlobalOS interfaces
- Send nodes through a pipeline of client operations
- Maintain ephemeral information during pipeline execution
- Build extensive list of clients and operations, let admins build machine-dependent pipelines

Provides

- OpenStack clients for low-level operations (power, network, system images)
- Ansible/Salt clients for provisioning

Future Work

- Better integration with event and control infrastructure
- Better extensibility

Fault Monitoring

Problem

- Applications and services are not aware of on-going faults/failures in the system

Solution

- Distributed service to monitor hardware and software faults
- Use enclave masters to aggregate/analyze events
- Allow application to subscribe to events

Impact

- Adapt checkpoint frequency of applications using machine knowledge and fault events
  - Service detects when system is in degraded state
  - Increase temporarily checkpoint frequency of applications
  - Use system knowledge to tune detection and checkpoint frequency increases
- Based on simulations, between 13% and 50% reduction of wasted times.

Future Work

- Integration with additional fault reporting interfaces
- Experiments on real application deployments

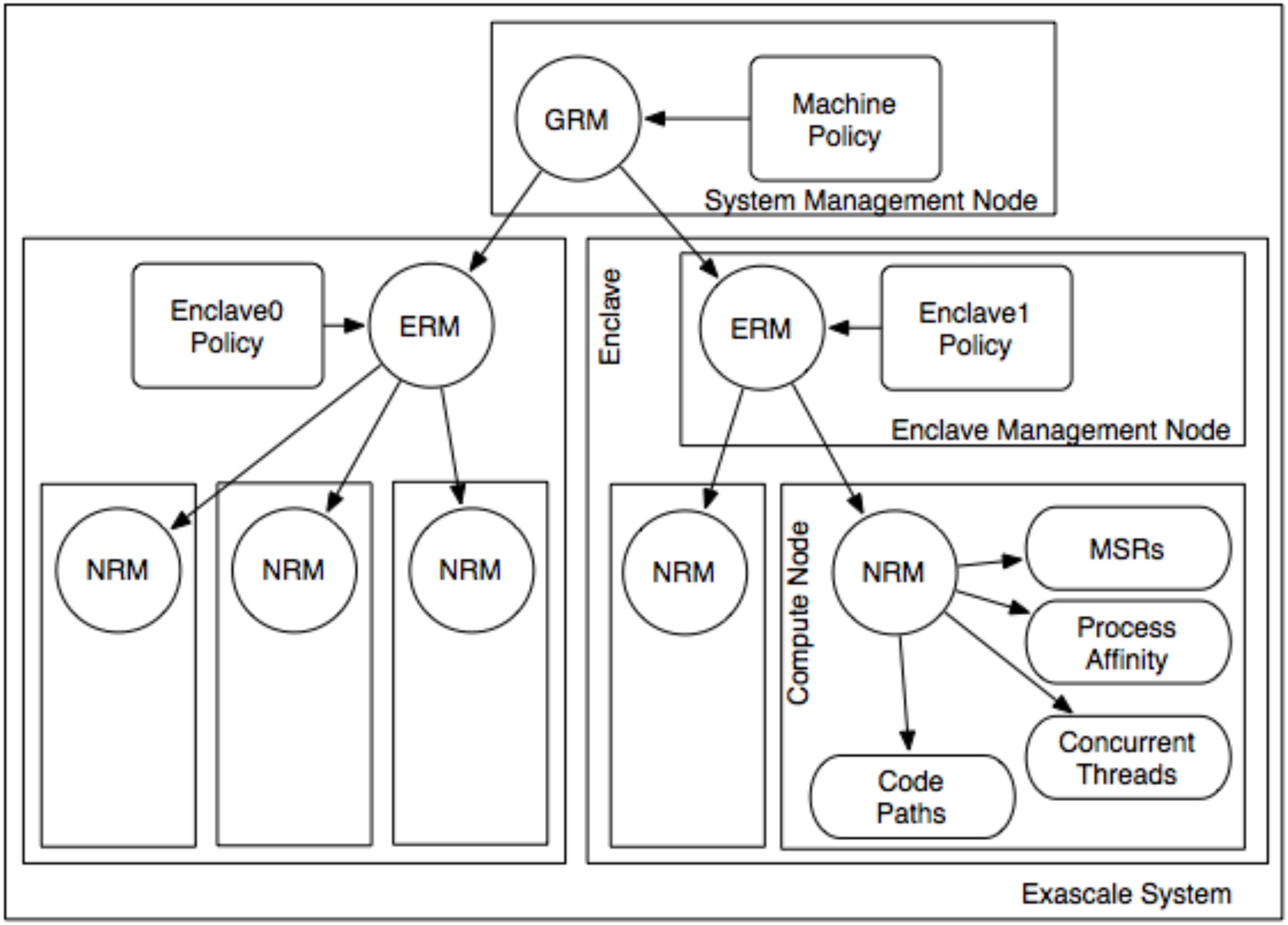
System-Wide Power Optimization

- **Power is a limited resource**
  - Power is THE limiting factor on the way to exascale
  - We will no longer be able to fully power all components
  - Consequence: Over-provisioned system with power caps

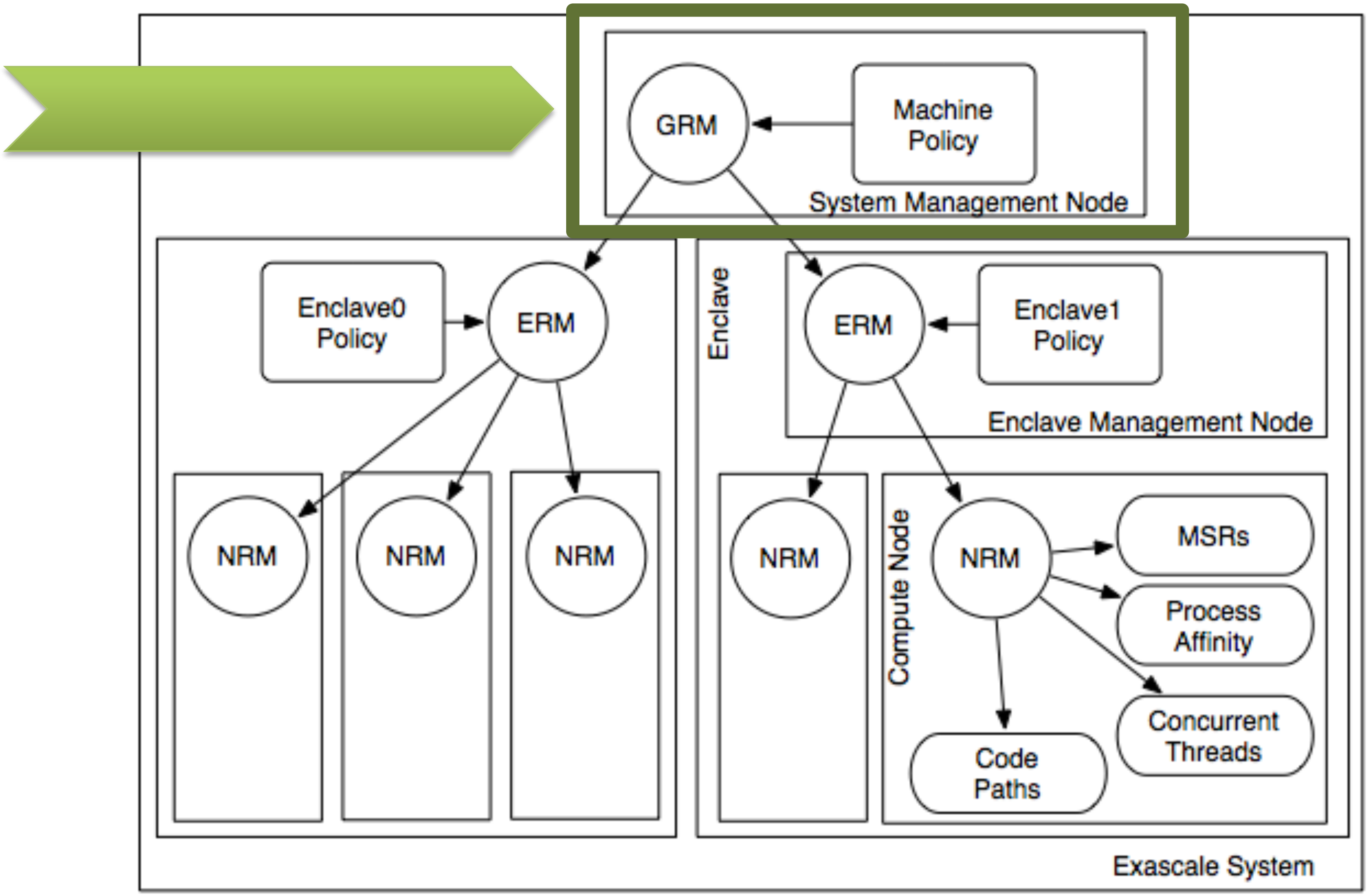
- **Power must be managed as a global resource**
  - The system power cap must be divided among jobs
  - Static division results in power fragmentation
  - Impact of load imbalance and manufacturing variability

- **Dynamic management can utilize open resources**
  - Part of the Global OS
  - Guided by user and administrator settings
  - At multiple levels of the system hierarchy

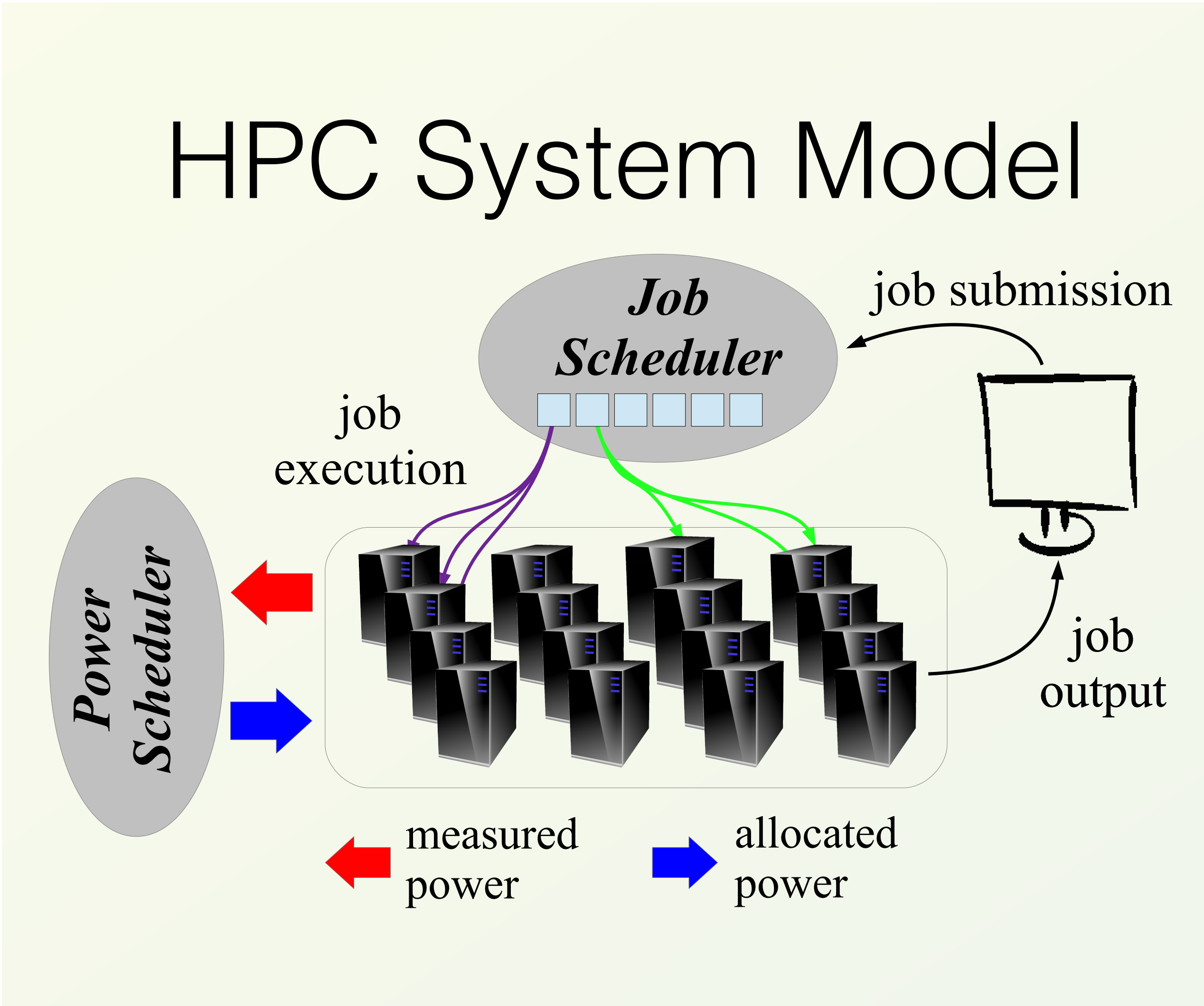
Hierarchical Power Management



Hierarchical Power Management

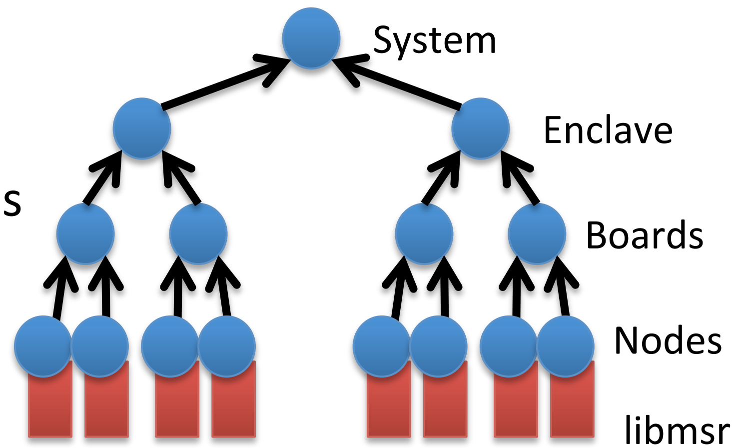


PowSched: A Global Application Agnostic Tuner



POWsched: The Basic Idea

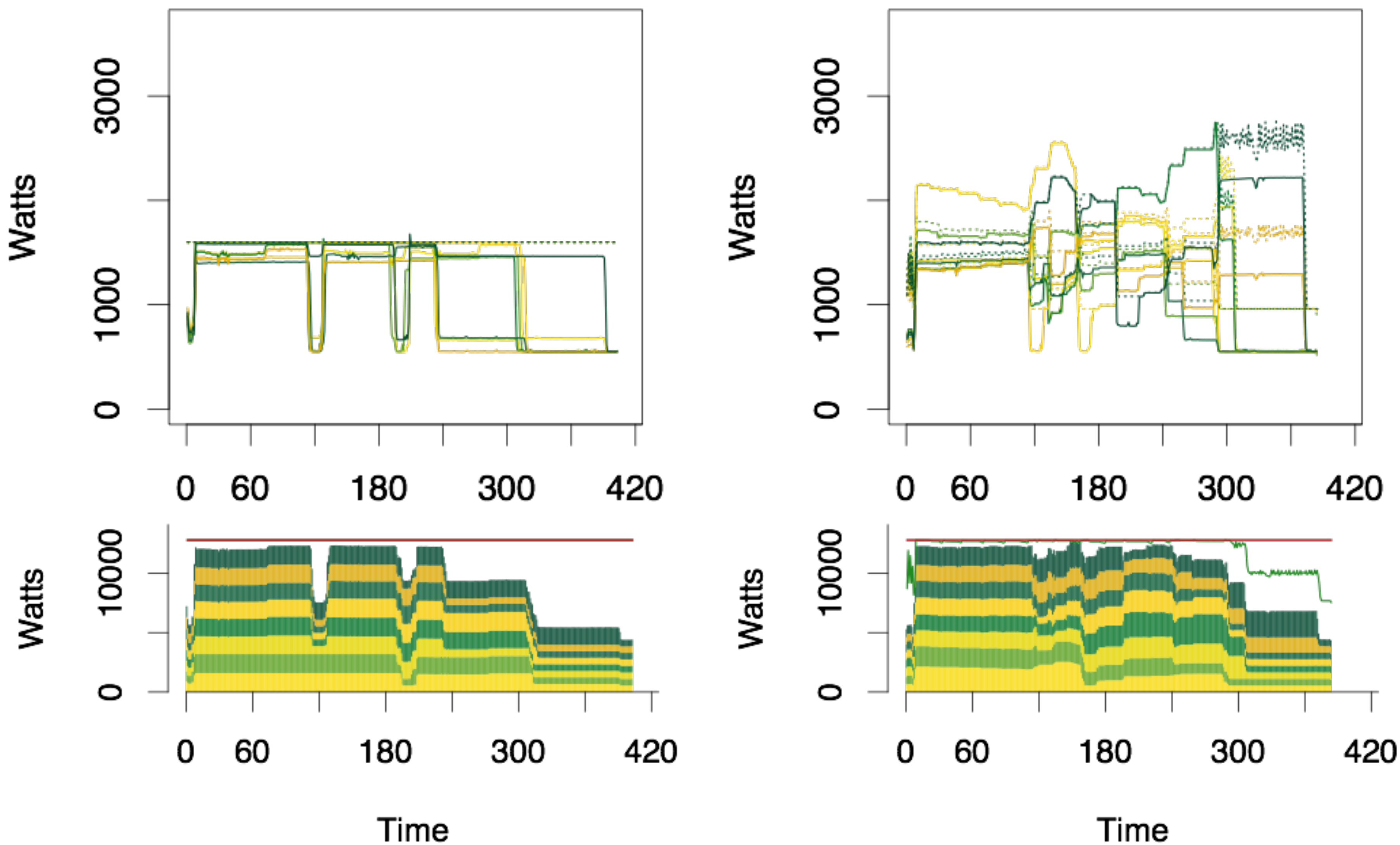
- **Monitor power usage of all applications**
  - OS level power monitors
  - Scalable aggregation
    - In a hierarchy representing applications
    - In a hierarchy representing the system



- **Detect “wasted” power**
  - Applications that don’t use their allocation
- **Detect “needed” power**
  - Applications that run against their power limit
- **Shift “wasted” power to applications that need it**

[Ellsworth et al., HPDC 2015]

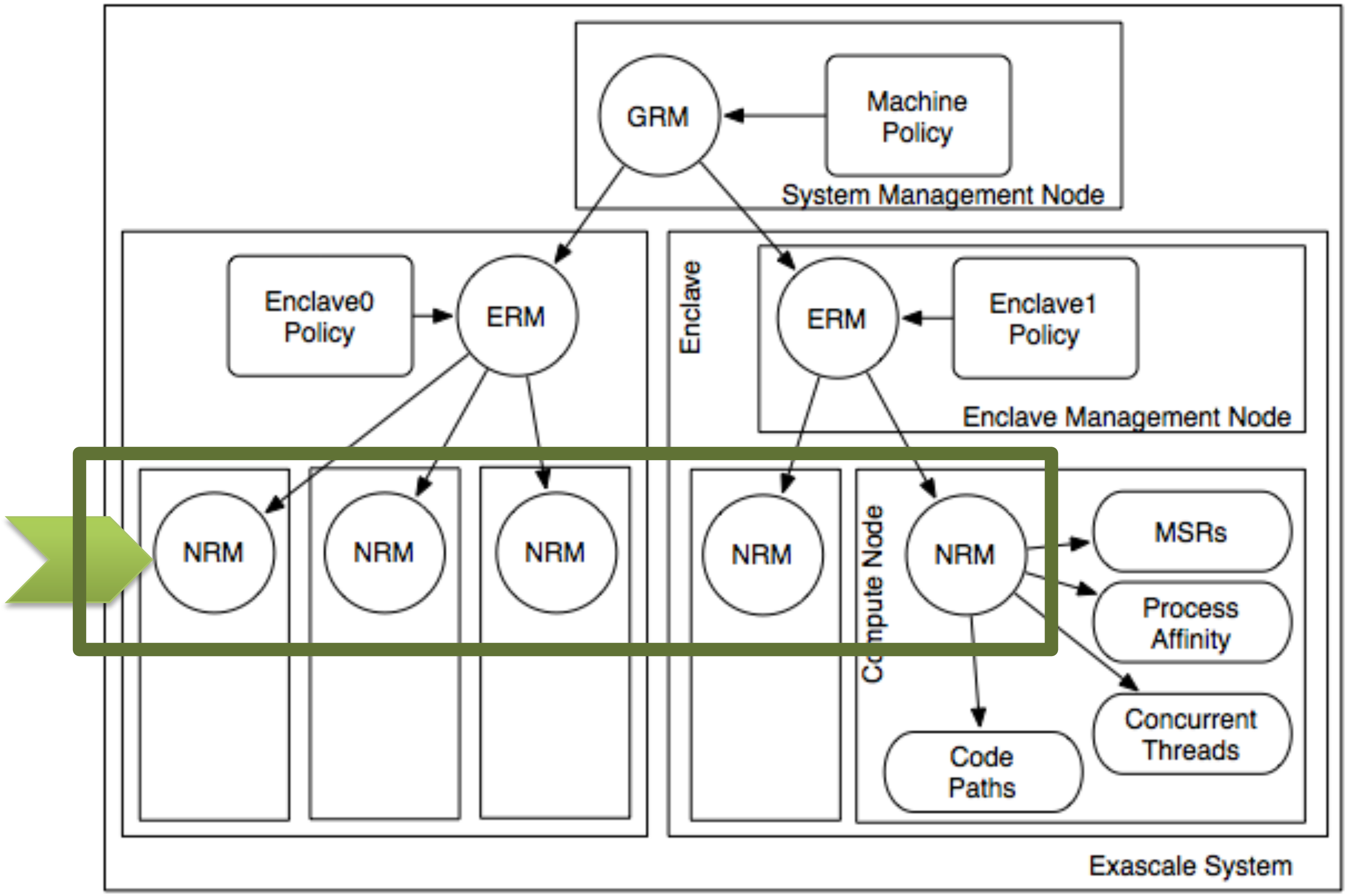
POWsched Results: 50W Static and Dynamic



POWsched Results: Static vs Dynamic

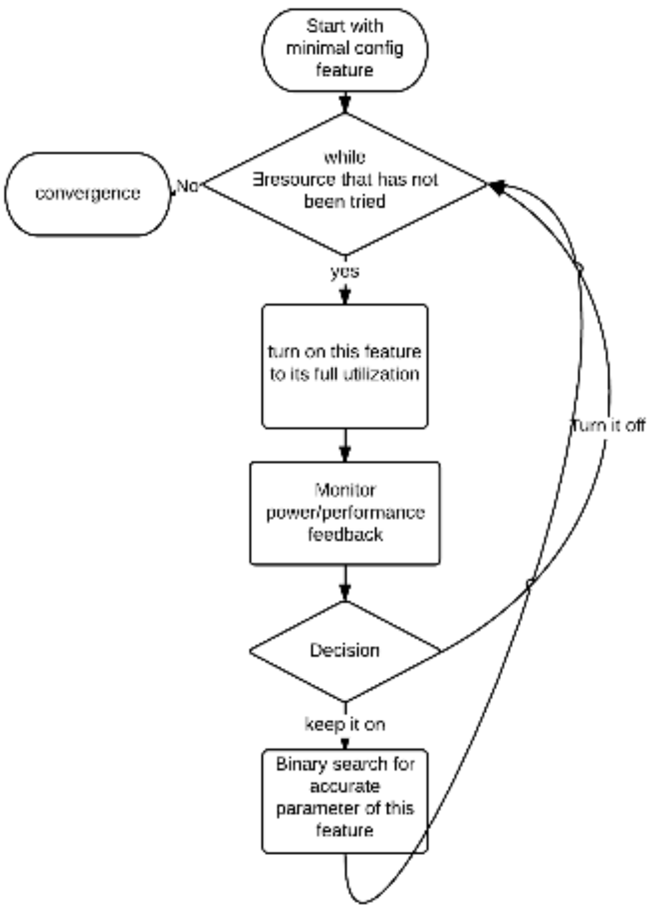
Experiment	Runtime	Stddev	Improvement
115W static	278.26	9.57	
115W dynamic	276.24	4.84	0.7%
90W static	284.63	3.20	
90W dynamic	277.13	5.04	2.6%
70W static	323.83	4.90	
70W dynamic	278.02	4.97	14.1%
50W static	407.21	18.00	
50W dynamic	371.92	13.23	8.7%

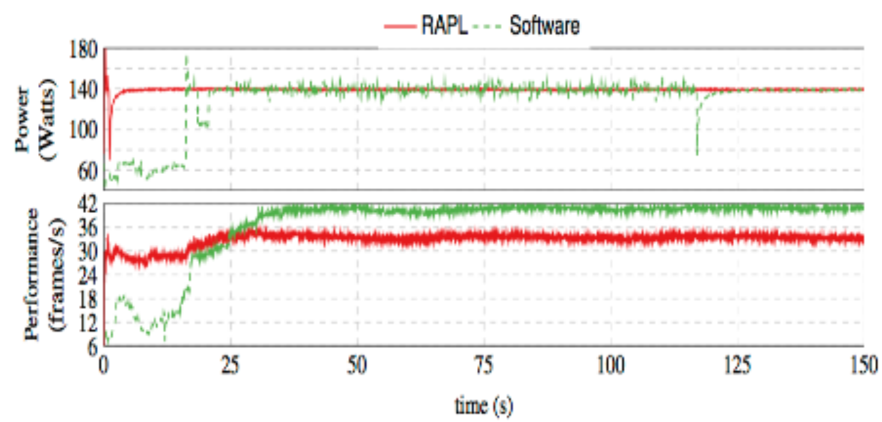
ARGO Global OS: A Hierarchical Solution



PUPiL: Optimizing Node Performance Under a Power Cap

Combines software *decision tree* for performance optimization with hardware power management (e.g. Intel RAPL)

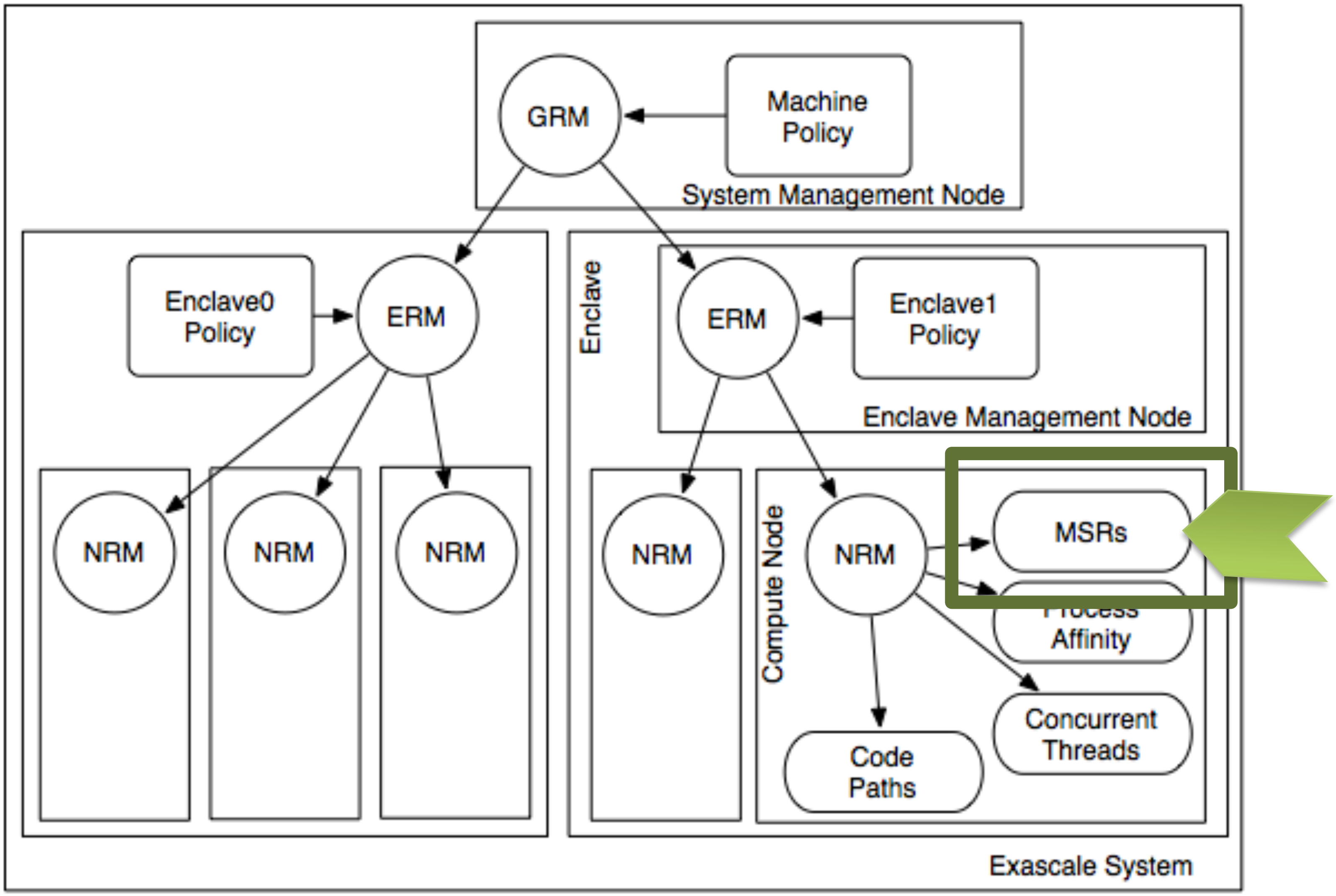




Delivers the performance of software machine learning with the responsiveness of hardware power capping

[Zhang and Hoffmann, ASPLOS 2016]

ARGO Global OS: A Hierarchical Solution



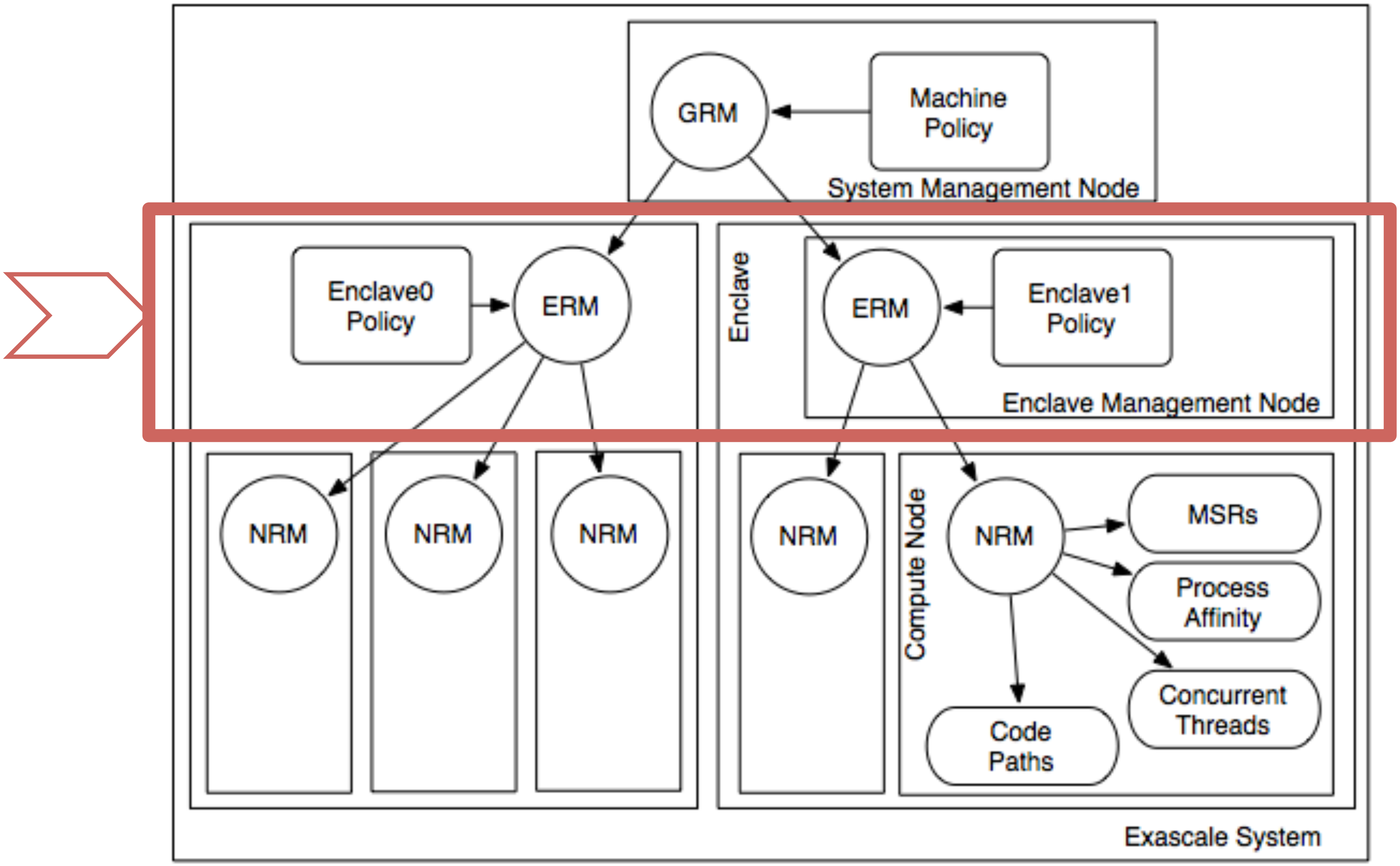
NodeOS Interfaces for RAPL Control

- **Power measurement and control requires low-level access**
  - On most chips implemented as model specific registers
  - Full access to them for every user is dangerous
  - Need extra access control

- **MSR-Safe and Libmsr**
  - MSR-safe: kernel module to whitelist certain registers
    - https://github.com/LLNL/msr-safe
  - LibMSR: Portable library encapsulating MSR semantics
    - https://github.com/LLNL/libmsr

- **Used in production**
  - Installed on several NNSA Tri-lab systems
  - Working with Redhat and Intel to integrate this into their stack

ARGO Global OS: A Hierarchical Solution



Job-Level Power Optimization

- **Adding application knowledge can increase accuracy**
  - Phase information
  - Communication behavior / critical path
  - Configuration and tuning parameters

- **An Enclave Level Power Manager is needed but raises questions**
  - How to integrate with node and global power managers?
  - How to determine suitable resource usage?

- **Conductor as one option for an ERM: Conductor**
  - Active on-node configuration management
  - Critical path detection and active power shifting
  - Integration with GRM through job power caps

- **Power aware scheduling and resource management**

[Conductor: Marathe et al., ISC 2016]

Global OS - Summary

- **Global OS extends node local services for a true exascale OS**
- **Infrastructure**
  – Enclave management
  – Communication structures
  – Support for in-situ vis
  – Fault monitoring
- **Dynamic global power management**
  – Arranged in a hierarchical fashion
    - Transparent to the application, system-wide
    - Cooperative within the application
    - Local optimization at node level
  – Prototype for transparent power shifting promisisng