ACES and Cray Collaborate on Advanced Power Management for Trinity (and beyond)

Alliance for Computing at Extreme Scale (ACES)

Sandia National Laboratories and Los Alamos Laboratory

in collaboration with Cray Inc.

Presented by: Steve Martin, Cray Inc.

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND2015-1497 C

TEAM

- **Sandia:**

  James H. Laros III, Kevin Pedretti, Ryan E. Grant,

  Stephen L. Olivier, Michael Levenhagen, David DeBonis

- **Los Alamos:**

  Scott Pakin, Jonathan Woodring

- **Cray:**

  Steven Martin, Matthew Kappel, Paul Falde

** Host of others, see acknowledgments in paper*

"Cray has a long history working with the Sandia National Laboratories and a strong partnership around developing new technologies and advanced solutions. **The foundation for Cray's roadmap in Advanced Power Management was built on the pioneering research jointly conducted at Sandia Laboratories on the first Cray XT platform, Red Storm.** In addition, Cray wants to acknowledge Sandia's leadership role in driving vendors to use common API's for power management and control,"

- ***Peter Ungaro, President and CEO of Cray Inc.***

Background

2006

- Initial research on first Cray XT platform - Red Storm

2012:

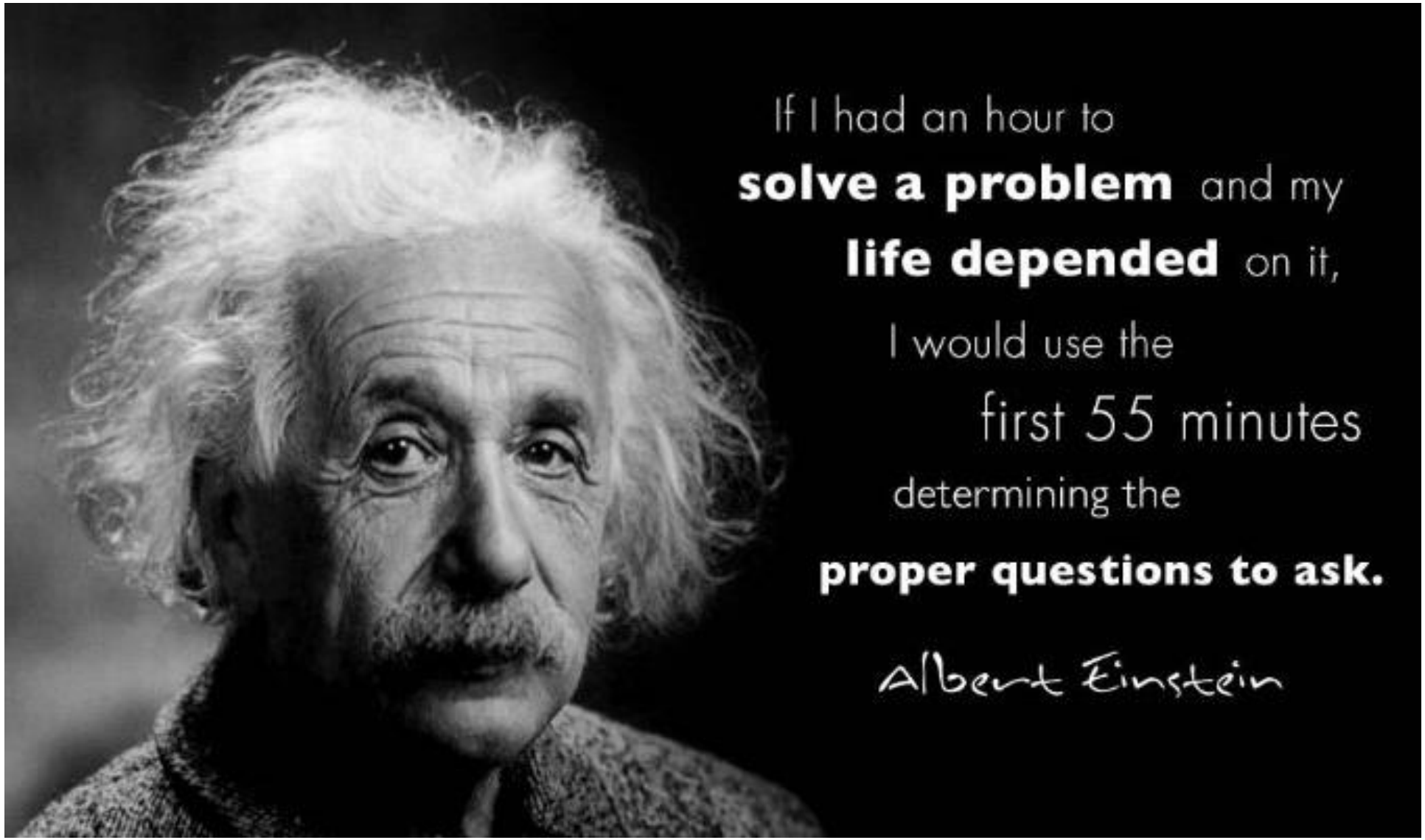
- Use case study conducted

2014

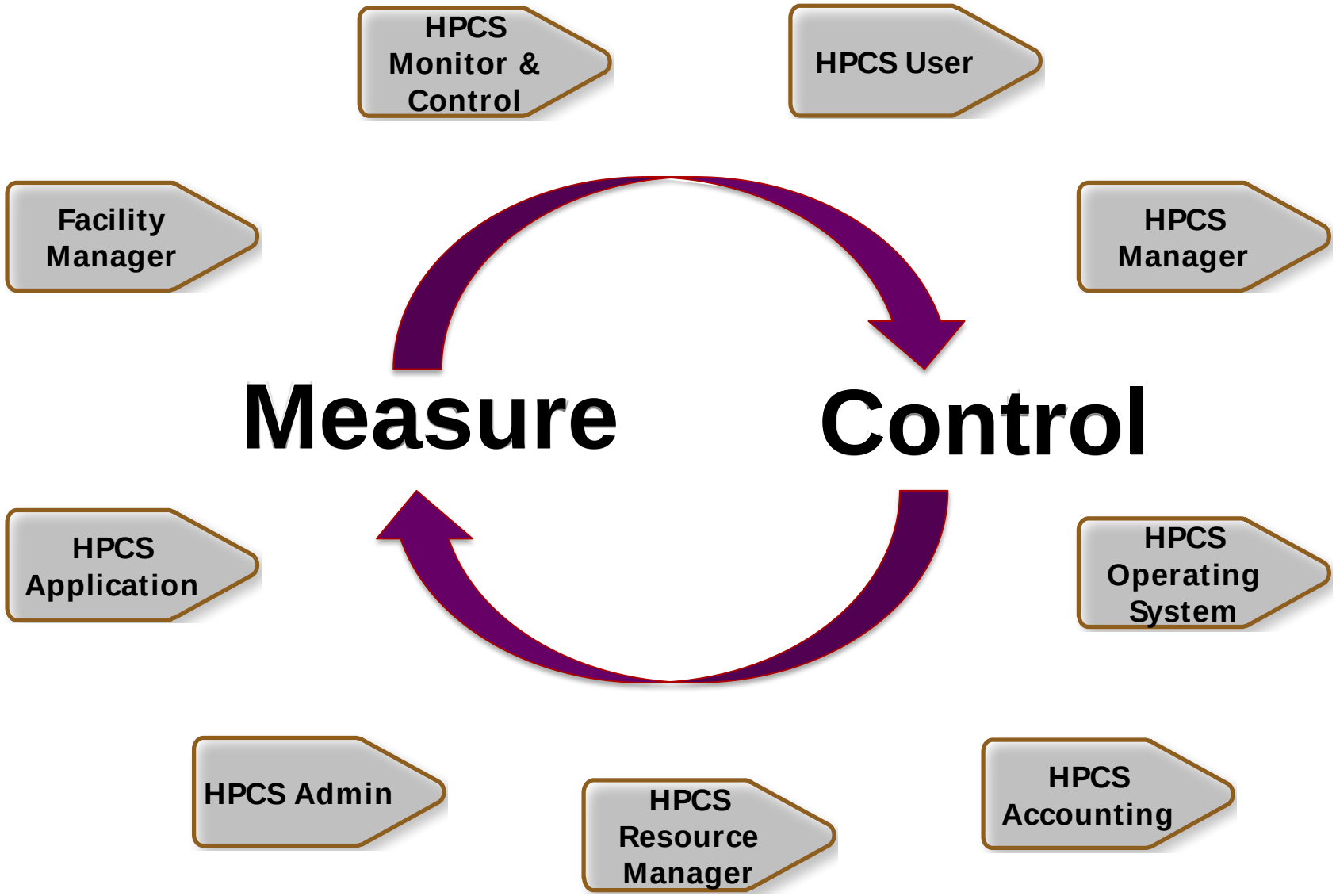
- Power API specification

2015

- Trinity Power NRE Project


If I had an hour to
solve a problem and my
life depended on it,
I would use the
first 55 minutes
determining the
proper questions to ask.
Albert Einstein

2006: Initial Research



2012: Use Case Study

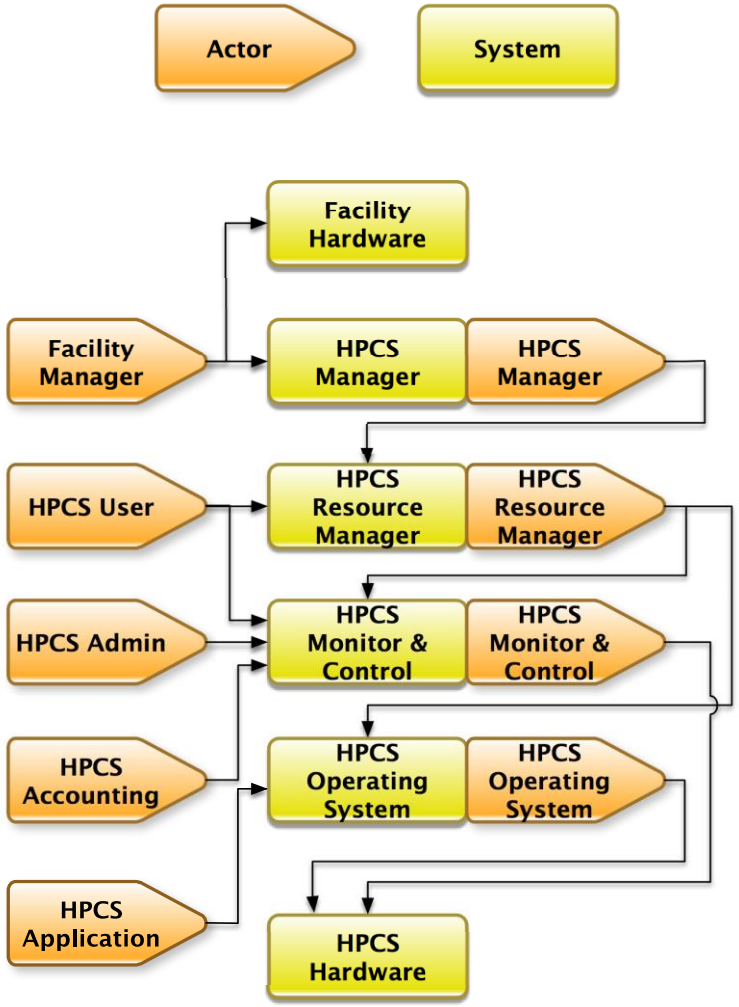
Diagram is the result of a UML study of the target space

- Goal: Define Scope, Roles and Interfaces

Arrows indicate interfaces or interaction between an Actor (Role) and System

- Each interaction represents an interface that is defined in the specification
- Specification is structured from the user or Role perspective

Notice that an Actor (Role) can also be a System



https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/UseCase-powapi.pdf

2014:Power API Specification

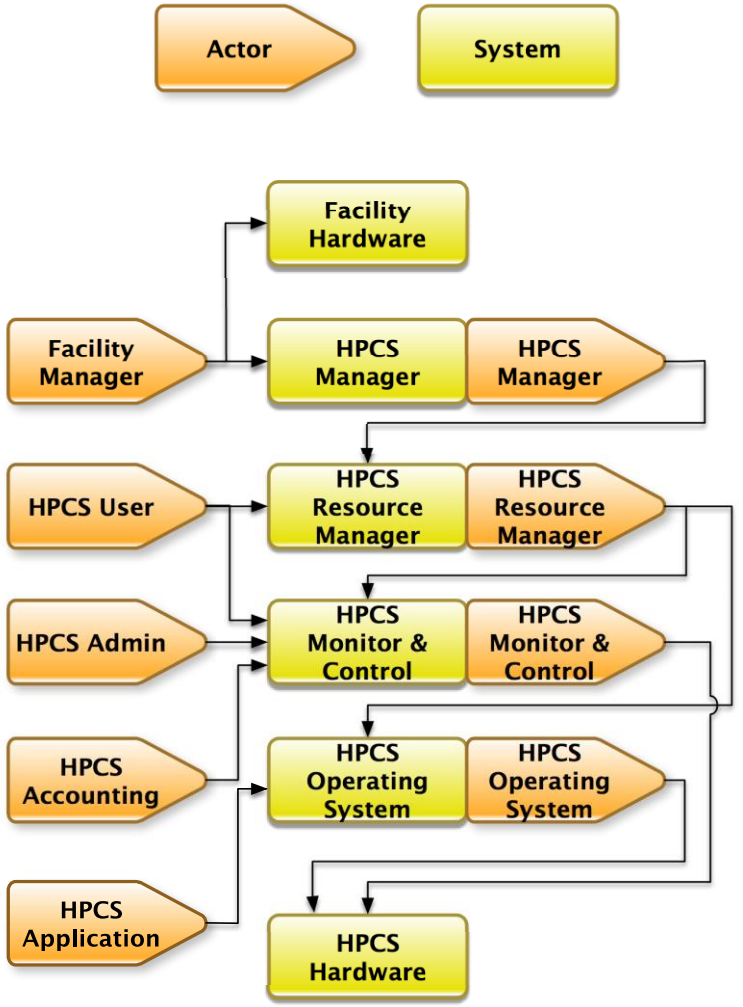
- Version 1.0, 1.1, 1.1a and 1.2 delivered
- Community needed a portable API for measuring and controlling power and energy
- Sandia developed Power API specification to fill this gap
- Provides portable power measurement and control interfaces
  - Covers full spectrum of facility to component
- First production implementation will be Trinity (ATS1)
- Continued (increasing) community involvement and influence

http://powerapi.sandia.gov

The *"High Performance Computing – Power Application Programming Interface Specification"* a.k.a. Power API

- Broad Scope
  - High-level: end user and applications
  - Low-level: hardware and operating system
- Roles (actors)
- Systems
- Interfaces
  - Roles interacting with Systems

```
typedef enum {
        PWR_ROLE_APP,   /* Application */
        PWR_ROLE_MC,    /* Monitor and Control */
        PWR_ROLE_OS,    /* Operating System */
        PWR_ROLE_USER,  /* User */
        PWR_ROLE_RM,    /* Resource Manager */
        PWR_ROLE_ADMIN, /* Administrator */
        PWR_ROLE_MGR,   /* HPCS Manager */
        PWR_ROLE_ACC    /* Accounting */
} PWR_Role;
```



Power API Goals

- Portability for the HPC community
  - Wouldn't it be nice to develop tools that worked on all your machines with little to no modification?
  - Same desire exists no matter what Role you play
- Forecast emerging needs of HPC community
  - As a group, inform the vendors of how we want to use systems now and in the future
  - Specification acts as a basis of collaboration
- Expose new capabilities developed by vendors and community
  - Leverage vendor and community innovations in this and related spaces
  - E.g. Geo and Redfish
- Most important, want something out there to throw stones at
  - Need a starting point!

What is the Power API?

- A comprehensive API for power MEASUREMENT and CONTROL of HPC platforms
  - Comprehensive = Facility to Component
  - API = Define the interface not the mechanism
  - HPC platforms = Facility (or datacenter) and all the platforms within

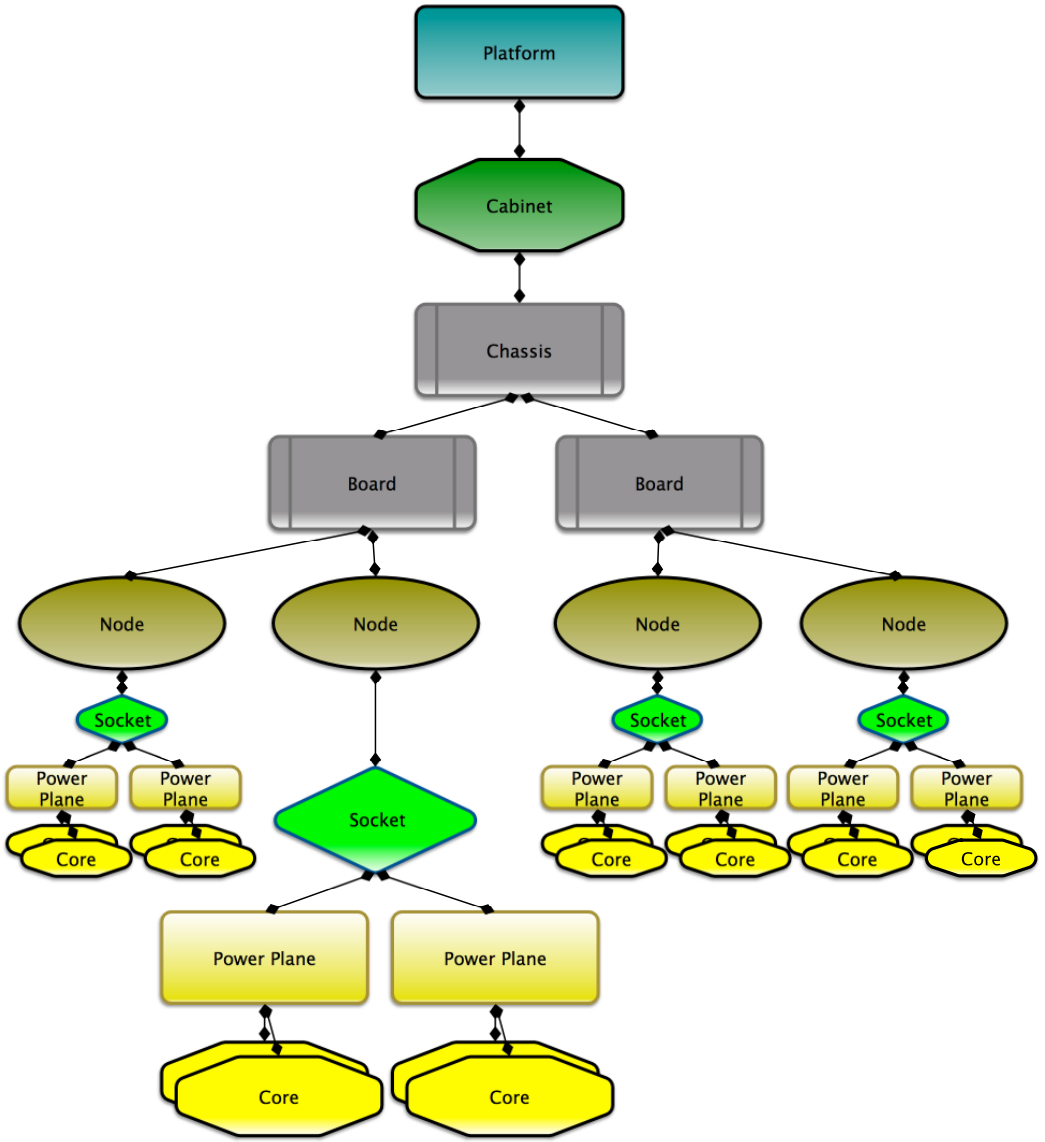
- Core (Common) among all “users” Includes:
  - Roles, Initialization, Navigation, Objects and Groups,
  - Attributes (Get/Set), Metadata and Statistics

- High-Level Common
  - Higher level of abstraction but still potentially common among multiple Roles

- Role/System Specific
  - Higher level abstraction specific to how Role interfaces with system

System Description

```
typedef enum {
        PWR_OBJ_PLATFORM,
        PWR_OBJ_CABINET,
        PWR_OBJ_CHASSIS,
        PWR_OBJ_BOARD,
        PWR_OBJ_NODE,
        PWR_OBJ_SOCKET,
        PWR_OBJ_CORE,
        PWR_OBJ_POWER_PLANE,
        PWR_OBJ_MEM,
        PWR_OBJ_NIC,
        PWR_OBJ_INVALID
} PWR_ObjType;
```



Foundation: Measurement and Control

- Monitor:

```
int PWR_ObjAttrGetValue( PWR_Obj object,
                         PWR_AttrName attr,
                         void* value,
                         PWR_Time* ts );
```

- Control:

```
int PWR_ObjAttrSetValue( PWR_Obj object,
                         PWR_AttrName attr,
                         const void* value );
```

```
typedef enum {
    PWR_ATTR_PSTATE = 0, /* uint64_t */
    PWR_ATTR_CSTATE, /* uint64_t */
    PWR_ATTR_CSTATE_LIMIT, /* uint64_t */
    PWR_ATTR_SSTATE, /* uint64_t */
    PWR_ATTR_POWER, /* double, Watts */
    PWR_ATTR_CURRENT, /* double, Amps */
    PWR_ATTR_VOLTAGE, /* double, Voltage */
    PWR_ATTR_MAX_POWER, /* double, Watts */
    PWR_ATTR_MIN_POWER, /* double, Watts */
    PWR_ATTR_FREQ, /* double, Hz */
    PWR_ATTR_ENERGY, /* double, Joules */
    PWR_ATTR_TEMP, /* double, Celsius */
    PWR_ATTR_OS_ID, /* uint64_t */
    PWR_ATTR_NUM_ATTRS,
    PWR_ATTR_INVALID = PWR_ATTR_NUM_ATTRS,
} PWR_AttrName;
```

Cray PM on XC system

- First released in 2013
  - System Management Workstation (SMW) 7.0.UP03
  - Cray Linux Environment (CLE) 5.0.UP03
- Power Management Database (PMDB)
- System Power Capping
- PM Counters /sys/cray/pm_counters
- Resource Utilization Reporting (RUR)

- Online documentation:
  - http://docs.cray.com/books/S-0043-7204/S-0043-7204.pdf

Cray Advanced Platform Monitoring and Control (CAPMC)

- Cray Advanced ~~Power~~ Platform Monitoring and Control
  - CAPMC, much more than just power
- Released in the fall of 2014
  - SMW 7.2.UP02 and CLE 5.2.UP02
  - New features in upcoming release
- Enabling Workload Managers (WLM)
  - Secure, authenticated, off-smw, monitoring and control interfaces
  - Supported by major WLM partners on XC systems

- CAPMC online documentation:
  - http://docs.cray.com/books/S-2553-11/S-2553-11.pdf

Trinity NRE: Three Areas of Focus

1. Cray: Power Management Database interface (PMDB)
   - Python (coming in Version 2.0 of Power API specification)
   - Implementation of PowerAPI in Python providing access to Cray's PMDB

2. Cray: Compute node interface
   - Node level C interface
   - Implementation in C on Cray Compute Nodes for Trinity systems

3. Adaptive: Power Aware Scheduling
   - Focusing on features that will enable use cases
   - Exercise capabilities implemented as part of #2 and potentially #1 above

Trinity NRE: PowerAPI access to PMDB

```
# INIT
myPwrCntxt = pwr.Cntxt(pwr.CntxtType.DEFAULT, pwr.Role.RM, "Default")
myPwrObj = myPwrCntxt.GetObjByName(objName)
attrName = pwr.AttrName.POWER

# Get Point in Time
measInfo = myPwrObj.AttrGetValue(attrName)
measurementValue = measInfo.value
measurementTime = measInfo.timestamp

# Average over time single object
attrStat = pwr.AttrStat.AVG
endTime = Time(time.time()) # current time.
timePeriod = timePeriod(start=(endTime-3600.0), end=endTime) # one hour.
statInfo = myPwrObj.GetStat(attrName, attrStat, timePeriod)
statisticValue = statInfo.value
statisticTimePeriod = statInfo.timestamp
```

PowerAPI in Python example code

Trinity NRE: Compute Node

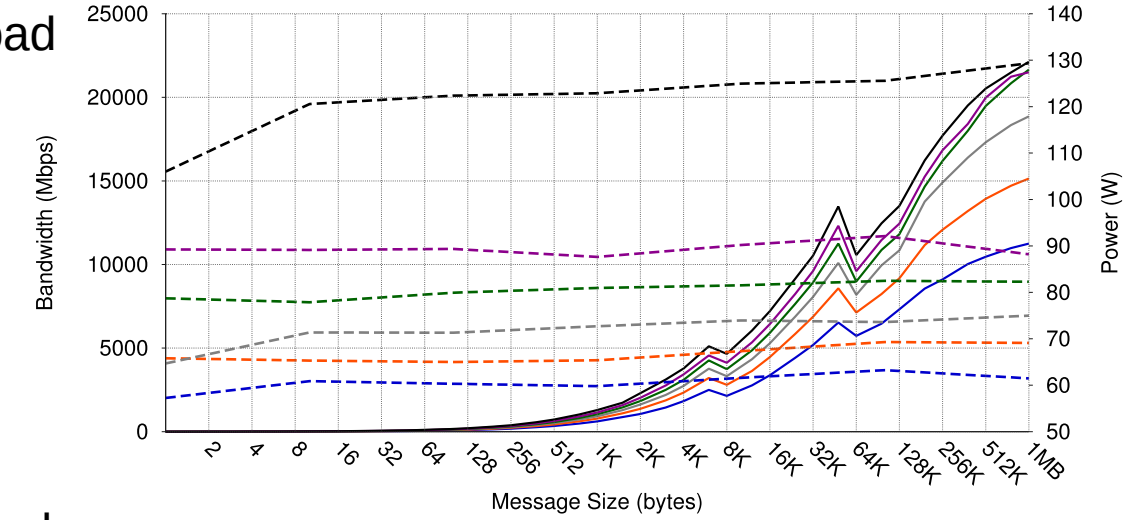
PowerAPI C example code

```
PWR_Cntxt myPwrCntxt;
PWR_Obj myPwrObj;
double orig_freq, io_freq;
PWR_Time my_ts;

/* Init context, and entry point... w/out error checking/handling */
rc = PWR_CntxtInit(PWR_CNTXT_DEFAULT, PWR_ROLE_RM, "foo", &myPwrCntxt);
rc = PWR_CntxtGetEntryPoint(myPwrCntxt, &myPwrObj);
…

/* Get the current frequency for myPwrObj */
rc = PWR_ObjAttrGetValue(myPwrObj, PWR_ATTR_FREQ, &orig_freq, &my_ts);
…
/* Set new frequency */
io_freq = MY_IO_FREQ;
rc = PWR_ObjAttrSetValue(myPwrObj, PWR_ATTR_FREQ, &io_freq);
```

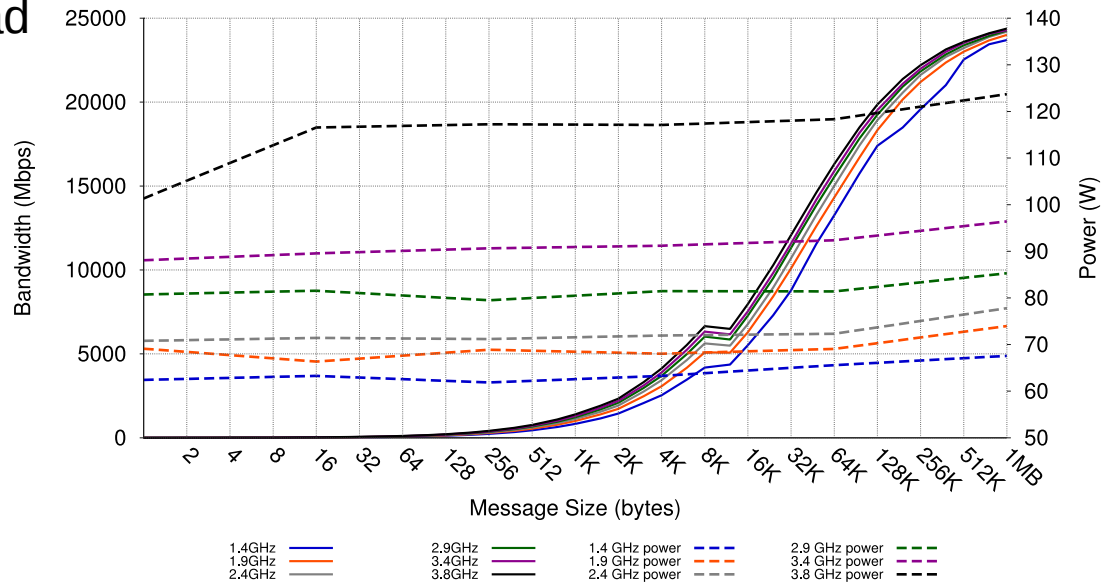
Trinity NRE: Compute Node

- Leverage potential power savings using application hints
- Onload (CPU involved) – communication performance is effected by processor frequency changes – Ignore hint
- Offload – communication performance maintained in addition to significant power savings
- Application (or run-time) provides the hint
- Underlying hardware can choose to leverage hints based on circumstances

Onload



Offload



Trinity NRE: Power Aware Scheduling

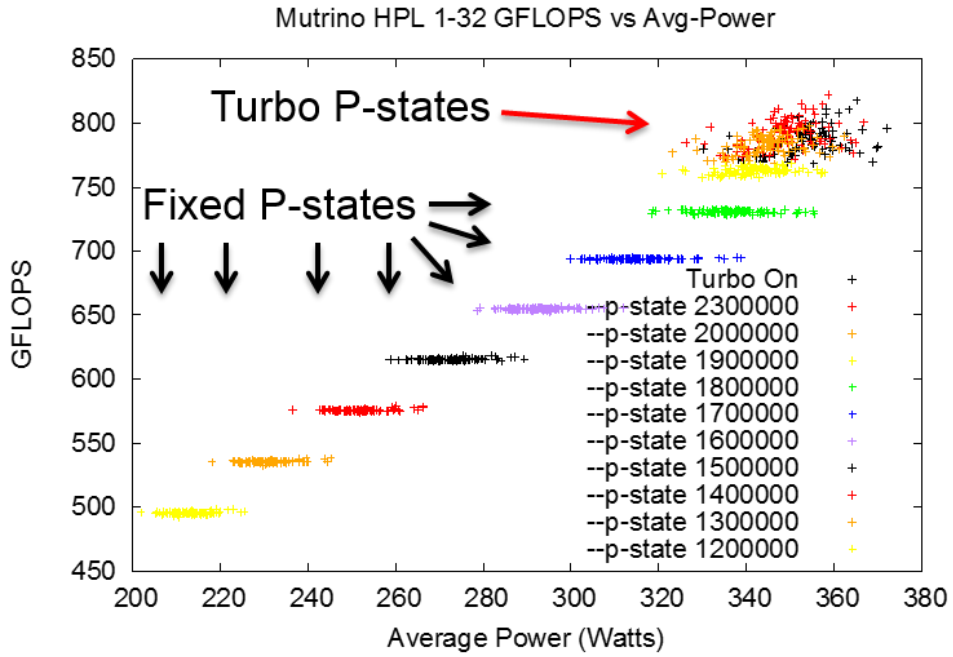
Power capping appealing due to max power draw "guarantee"

- **Recapture (*WallPlate – Pcap*) headroom and reallocate elsewhere**
- **Potentially useful for doing power aware scheduling**

P-state control appealing due to predictable performance, power unknown

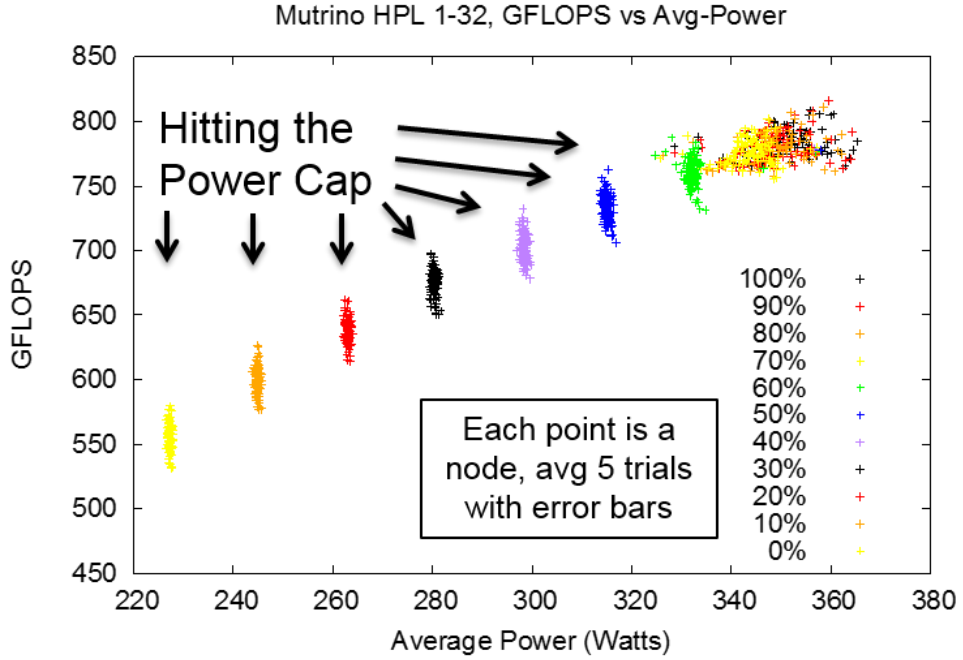
P-State Sweep

Fixed Performance, Variable Power



Power Cap Sweep

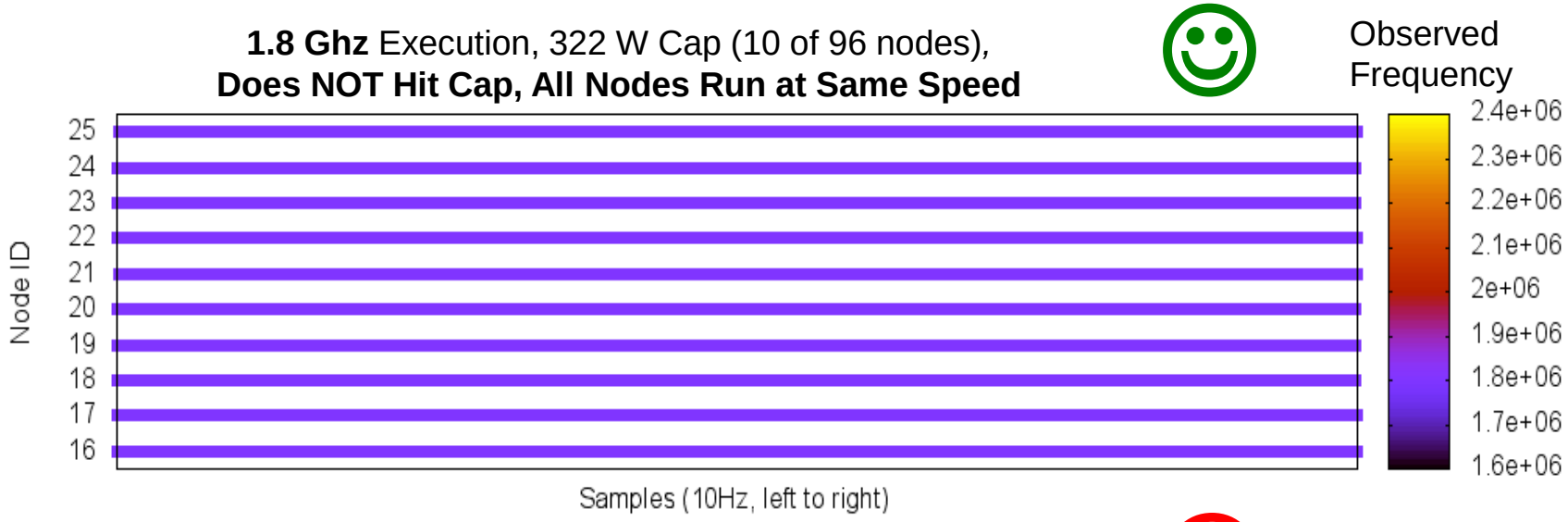
Fixed Power, Variable Performance



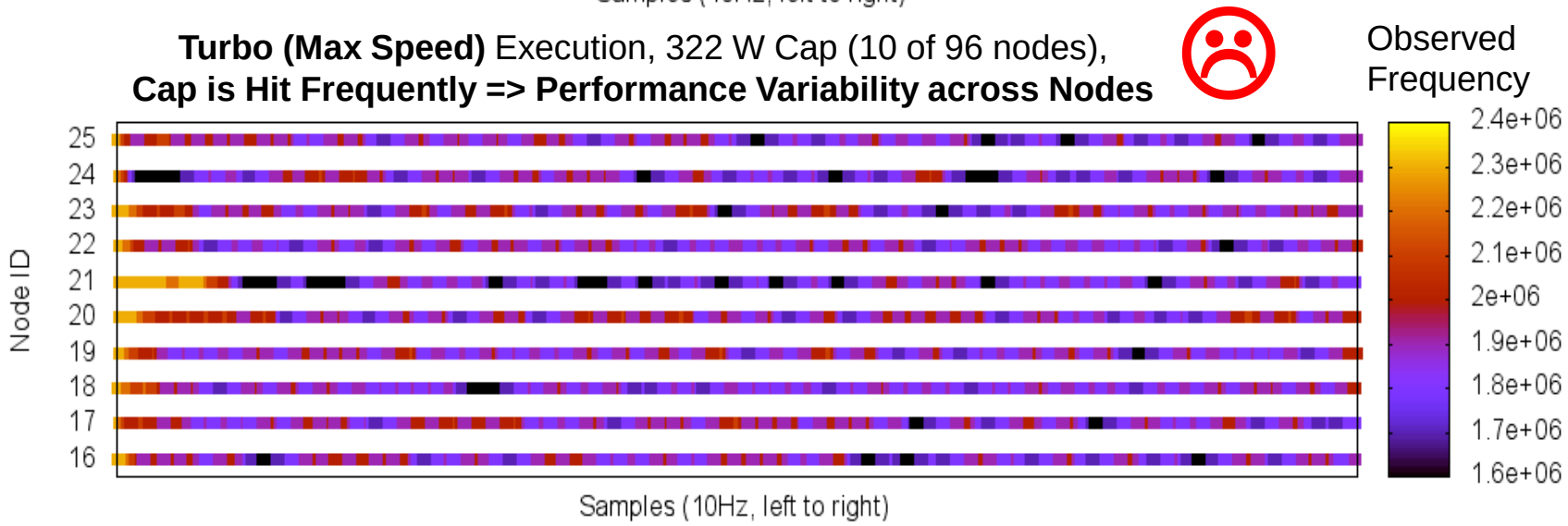
Trinity NRE: Power Aware Scheduling

- S3D CPU 0 Frequency over time for 10 nodes
- Hitting Power Cap Leads to Frequency Oscillation
- Performance variability is a problem for bulk synchronous applications
  - Only as fast as the slowest node

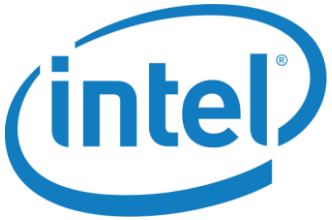
1.8 Ghz Execution, 322 W Cap (10 of 96 nodes),
Does NOT Hit Cap, All Nodes Run at Same Speed



Turbo (Max Speed) Execution, 322 W Cap (10 of 96 nodes),
Cap is Hit Frequently => Performance Variability across Nodes



Who is Behind PowerAPI?



<Your logo here!>

Thank you – Questions?

http://powerapi.sandia.gov/

Acknowledgments:

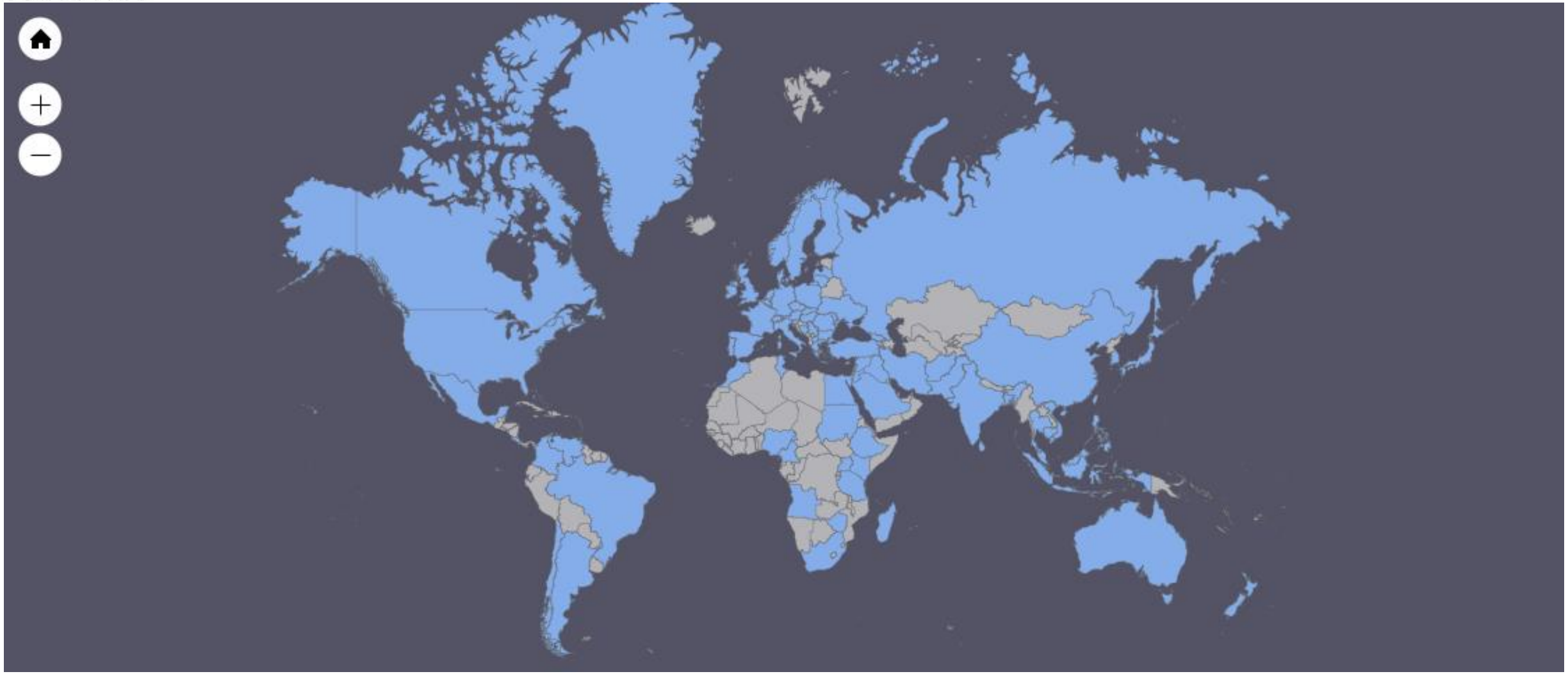
This work was funded through the Computational Systems and Software Environment sub-program of the Advanced Simulation and Computing Program funded by the National Nuclear Security Administration

Los Alamos NATIONAL LABORATORY EST.1943
Adaptive COMPUTING

Sandia National Laboratories
CRAY THE SUPERCOMPUTER COMPANY

BACKUP

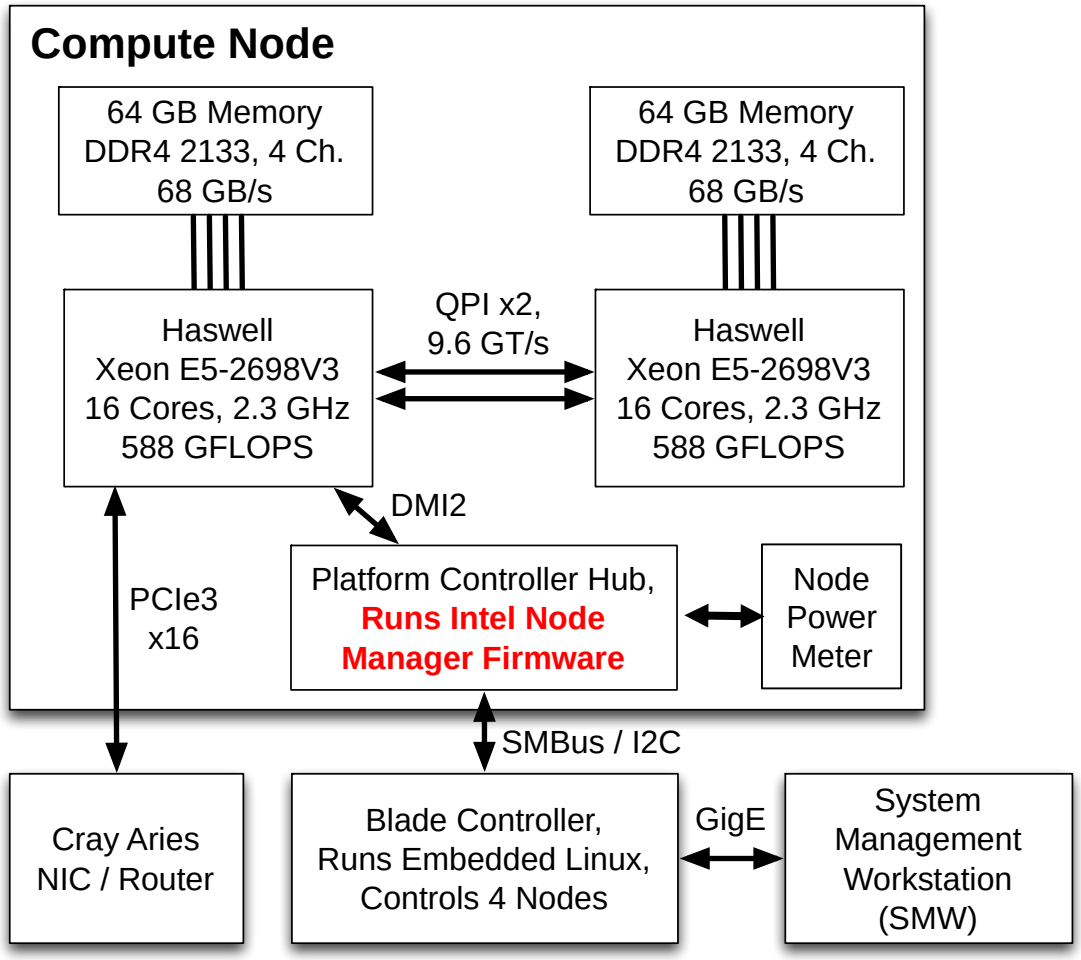
96 Countries



http://powerapi.sandia.gov

Node-Level Power Capping

Trinity "Haswell" Compute Node

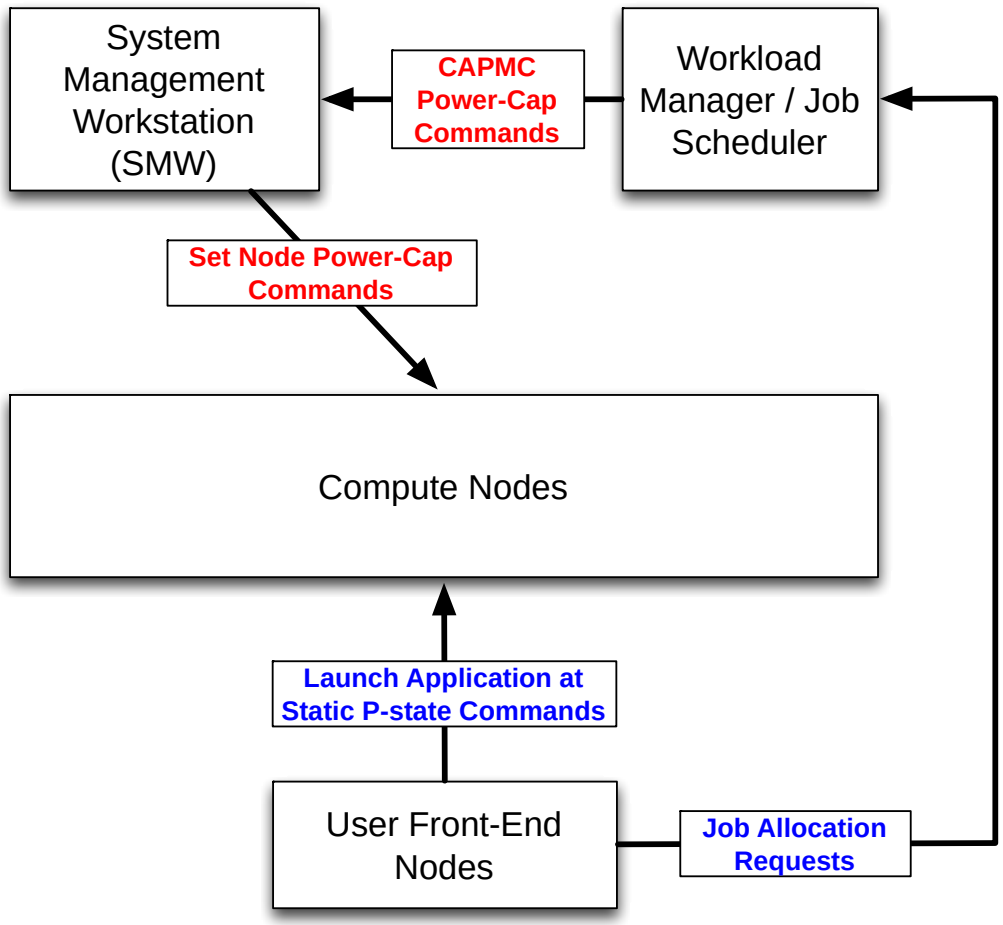


- Blade controller receives power cap command from management network
- Blade controller relays command to Intel PCH
- PCH runs Intel's Node Manager firmware, which enforces the power cap
- Cray node-level power meter provides feedback to Node Manager
- Node manager makes use of each socket's RAPL
- Power capping algorithm not disclosed by Intel

Specification: Intel Intelligent Power Node Manager 3.0, March 2015, Doc# 32200-001US

Job/System-Level Power Capping

Cray XC40 Power Management Architecture



- A single management workstation controls system, the SMW
- Node-level power caps set from SMW, distributed to compute nodes via out-of-band management network
- Admins use `xtpmaction` command to set power caps manually
- Workload managers use Cray's CAPMC web API to set power caps
- Users may launch their job at a fixed p-state, default is P0 (turbo on)

See: Monitoring and Managing Power Consumption on the Cray XC System, April 2015, S-0043-7203

Trinity NRE: PowerAPI access to PMDB

```
# INIT
myPwrCntxt = pwr.Cntxt(pwr.CntxtType.DEFAULT, pwr.Role.RM, "Default")
myPwrObj = myPwrCntxt.GetObjByName(objName)
attrName = pwr.AttrName.POWER

# Point in Time
measInfo = myPwrObj.AttrGetValue(attrName)
measurementValue = measInfo.value
measurementTime = measInfo.timestamp

# Maximum overTime Multiple objects
attrStat = pwr.AttrStat.MAX
statInfo = myPwrObj.GetStat(attrName, attrStat, timePeriod)
statisticValue = statInfo.value
statisticTimePeriod = statInfo.timestamp
statisticPwrObj = statInfo.obj
```

PowerAPI in Python example code