GEO Progress Updates
(Global Energy Optimization)

Project Lead: Jonathan Eastep, PhD & Principal Engineer
jonathan.m.eastep@intel.com

May 12, 2016
Recap of GEO Project Scope and Goals

- GEO is a runtime for energy optimization in HPC systems
  - Application-level: launches with and runs with the application
  - Global: coordinates DVFS / power alloc decisions across nodes
  - Open source: BSD 3-clause license
  - Scalable: tree-hierarchical control and telemetry aggregation
  - Extensible: plug-ins for extensions + out-of-the-box functionality

- Goals:
  - Report per-job (or per-application-phase) energy/perf profile info
  - Provide out-of-the-box functionality to unlock substantially more performance in power-limited systems
  - Provide open platform for research community to accelerate innovation in HPC system energy management
Reported initial public release of GEO on github

- **Package Name:** geopm (stands for GEO power management)
- **Release goals:**
  - Define GEO interfaces and publish user docs for community review
  - Nail down modular OO-design in C++11 (w/ C external interfaces)
  - Include solid autotools build system & gtest/gcov test infrastructure
  - Include support for basic static power management functionality
    - Example: Uniform Frequency Static mode

- **Non-Goals:**
  - Code / feature-completeness
    - No dynamic power management yet (runtime was still under construction)
    - No support for extensibility via plug-ins yet
Status Update on Implementation (Current)

Completed a significant new geopm release

- **Release goals:**
  - Achieve functional correctness of runtime for dynamic power mgmt
  - Provide plug-in frameworks for extending GEO in two dimensions:
    - Add new energy management strategies
    - Add support for new target hardware platforms
  - Provide an out-of-the-box plug-in for a key US DOE use-case:
    - Goal: maximize application performance within a job power bound
    - Approach: dynamically reallocate power to speed up nodes on critical path
  - Provide developer documentation and additional user documentation

- **Non-Goals:**
  - Production quality test coverage (much testing included, more needed)
  - Benchmark and regression test infrastructure (work in progress)
  - Tuned-up power balancer plug-in (results not yet optimized)
Goal: develop GEO for deployment on Aurora in 2018
  - Note: earlier intercepts probable on other Phi or Xeon systems

Scope:
  - Work with Argonne/Cray to integrate GEO into Aurora software stack
  - Nail down key use-cases for GEO & user incentives for running it
  - Explore power-aware scheduler functions in Cobalt Job Mgmt Suite

Status:
  - [COMPLETE] Define GEO design and integration architecture
  - [NEXT STEP] Bring up test cluster at Argonne for integration work
  - [NEXT STEP] Demo GEO running on KNL cluster (proxy for Aurora)
Status Update on Collaborations: LLNL

- **Goal:** work toward deploying GEO on LLNL production systems
- **Scope:**
  - Develop high-performance safe userspace interfaces to power/perf monitors and controls (build on msr-safe)
  - Study /enhance GEO scalability on LLNL catalyst test cluster
  - Explore integrating Conductor energy mgmt technology into GEO
- **Status:**
  - [COMPLETE] msr-safe enhancements for performance
  - [NEXT STEP] Work with LLNL and Cray and attempt to get msr-safe adopted in OpenHPC and SLES/RHEL Linux distros
  - [NEXT STEP] Begin GEO scaling work
  - [NEXT STEP] Begin exploring Conductor integration
Status Update on Collaborations: Sandia

- **Goal:** work toward compatibility between Sandia Power API and GEO APIs and explore integration feasibility

- **Scope:**
  - Exchange feedback to influence future API versions, simplify wrapping
  - Explore feasibility of having GEO provide some of the control and monitoring functionality specified in Sandia API

- **Status:**
  - [COMPLETE] GEO team to modify application API for simpler wrapping
  - [COMPLETE] GEO team to suggest changes to Sandia application API for compatibility
  - [NEXT STEP] Sandia working to incorporate feedback on application API into a future version of the spec
  - [NEXT STEP] Exchange feedback on design of interfaces between Workload Managers and Job-Level Energy Managers like GEO
## Project Information

<table>
<thead>
<tr>
<th>Type</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email contact</td>
<td><a href="mailto:jonathan.m.eastep@intel.com">jonathan.m.eastep@intel.com</a></td>
</tr>
<tr>
<td>Project page url</td>
<td>geopm.github.io/geopm</td>
</tr>
<tr>
<td>Project repo url</td>
<td>github.com/geopm</td>
</tr>
<tr>
<td>Release notes url</td>
<td>github.com/geopm/geopm/releases/tag/v0.1.0</td>
</tr>
<tr>
<td>EE HPC WG Webinar on GEO hyperlinks</td>
<td>slides and audio</td>
</tr>
</tbody>
</table>
GEO Team Acknowledgements

GEO Core Team (Intel)
- Fede Ardanaz
- Chris Cantalupo
- Jonathan Eastep
- Stephanie Labasan
- Kelly Livingston
- Steve Sylvester
- Reza Zamani
- ... and hiring!

Collaborators (Intel)
- Tryggve Fossum
- Al Gara
- Richard Greco
- David Lombard
- Ram Nagappan
- Mike Patterson
Backup Slides
GEO Capabilities

- Comprehend and mitigate dynamic load imbalance by globally coordinating frequency and power allocations across nodes.
- Leverage application-awareness and learning to recognize patterns in application (phases), then exploit patterns to optimize decisions.
- React to phase changes at aggressive time scales (low milliseconds) and rapidly redistribute limited power to performance-critical resources.
- Tackle the scale challenges prior techniques have swept under the rug to enable holistic joint optimization of power policy across the job.
Recap of GEO Integration Architecture

- User Interface (Work w/ job queues)
- Admin Interface (Work w/ WLMs)
- Workload Manager
  - Power-Aware Sched (Work w/ Intel to implement)
  - Workload Mngr Interface (Work w/ WLMs)
  - Application Power Mngr = GEO
  - PCU RAPL and Perf Counter Interfaces (Work w/ Intel to enhance)

Owner
- 3rd parties
- Intel GEO team
- Intel PM Arch team

Initially required; later optional if not tuning SW knobs

Intel Corporation
GEO Hierarchical Architecture

GEO manages job to a power budget and globally coordinates frequency & power allocation decisions

Scaling challenge is addressed via tree-hierarchical design & hierarchical policy

- Each agent owns sub-problem: decide how to divide/balance power among children
- Power/perf telemetry is scalably aggregated so network traffic is minimal
- Tuning is globally optimized despite distributed tuning: achieved through Hierarchical-POMDP learning techniques

GEO tree runs in 1 reserved core per CN

- Leaf & non-leaf agents run in these cores
- Enables fast reaction times, deep analysis
- Overhead is negligible in manycore chips
- Designing for minimal memory footprint
Zoom-In on Leaf Agent

Power budgeting inside the processor:
- Spatio-Temporal Energy Scheduling (phase-adaptively allocate power among RAPL power domains)
Auto-Tuner Prototype Results Summary

Speedup from Auto-Tuner at ISO Power

Speedup derives from two factors: correcting load imbalance across nodes and node-local spatio-temporal energy scheduling optimizations exploiting phases.

Bars represent average results over a range of assumptions about how much power the job is allocated and how much load imbalance is present.

Experimental setup carefully emulates large-cluster load imbalance on a small cluster.

Results collected while running on Xeon hardware (not simulation).
Redfish Update for EE HPC

DMTF Scalable Platforms Management Forum
May 2016
Disclaimer

• The information in this presentation represents a snapshot of work in progress within the DMTF.

• This information is subject to change without notice. The standard specifications remain the normative reference for all information.

• For additional information, see the Distributed Management Task Force (DMTF) website.
Scalable Platforms Management Forum

- Created in September 2014 – now 27 member companies
- Co-Chairs: Jeff Autor (HPE), Paul Vancil (Dell)
- Promoters: Broadcom Limited, Cisco, Dell, EMC, Emerson Network Power, Ericsson AB, Hewlett Packard Enterprise, Inspur, Intel, Lenovo, Microsoft, Supermicro, VMWare
- Supporters: AMI, Fujitsu, Huawei, IBM, Insyde Software, Mellanox, Microsemi, NetApp, Oracle, OSIsoft, Qualcomm, Quanta, Seagate, Western Digital
- **Charter:** Create and publish an open industry-standard specification and schema that meets the expectations of Cloud and Web-based IT professionals for scalable platform hardware management utilizing existing tool chains as well as being usable by personnel with minimal experience.
- **Alliance Partnerships**
  - OpenCompute Project
  - UEFI - Collaborating on Firmware Update and Host Interface work
  - SNIA – Collaborating on Storage modeling / alignment between SSM and Redfish
Redfish Specification

- RESTful interface over HTTPS in JSON format based on OData v4
- Usable by client applications and browser-based GUls
- A secure, multi-node capable replacement for previous interfaces
- Schema-backed human-readable output
- Covers popular use cases and customer requirements
- Intended to meet OCP Remote Machine Management requirements
Redfish v1.0 Specification & Schema

Retrieve “IPMI class” data
• Basic server identification and asset info
• Health state
• Temperature sensors and fans
• Power supply, power consumption and thresholds

Discover
• Service endpoint (network-based discovery)
• System topology (rack/chassis/server/node)

Basic I/O infrastructure data
• Host NIC MAC address(es) for LOM devices
• Simple hard drive status / fault reporting

Security
• Session-based, leverages HTTPS

Perform Common Actions
• Reboot / power cycle server
• Change boot order / device
• Set power thresholds

Access and Notification
• Serial console access via SSH
• Event notification method(s)
• Logging method(s)

BMC infrastructure
• View / configure BMC network settings
• Manage local BMC user accounts
Redfish releases

- v1.00 Released August 2015
  - Specification and Schema files
- v1.01 Errata Release November 2015
  - Clarifications to specification, corrected errors in schemas
- v1.10 Schema release November 2015
  - Additions to ComputerSystem, Chassis
- 2016.1 Release – NEW (April / May 2016)
  - New schemas for BIOS, Memory, Storage
  - Will correct schema naming issues (all schemas will be revised)
  - Clarifications to specification – errata release v1.0.2
- Releases planned for Schema and Specification
  - 2016.2 - Summer 2016 (July/August)
  - 2016.3 - Fall 2016 (November)
SPMF Work in Progress

• Significant expansion to data model coverage
  • PCIe devices
  • Storage subsystems
  • Network Adapters / Controllers
  • DIMM / NV-DIMM inventory
• “Task Force” sub-groups created to tackle specific topics
  • Host (OS) Interface to Redfish – working with DMTF PMCI
  • Firmware Update – working with UEFI and DMTF PMCI
  • Storage – working with SNIA
  • Privilege Mapping
• “Integration recipe” target for Redfish implementations
  • Strong desire for an OCP HW Management conforming property list
  • Other groups welcome to suggest target recipes

www.dmtf.org
Redfish Ecosystem – Tool Development underway

Github public repository
• Coming soon!

Client Library
• Common utility support functions
  • Discovery, Enumeration, etc.
  • Event subscription
• Typical tasks
  • Power on/off/reboot
  • Gather thermal data
• Languages under consideration
  • Python
  • Java
  • PowerShell
  • Other possibilities…

Command Line Utility
• Similar to IPMItool
• Designed for end users
• Calls Client library

Conformance Test Suite
• Schema-aware tool for testing
• Checklist for vendors and customers
• Avoid spec interpretation conflicts

Schema Dev Tools
• CSDL Validator
• CSDL to JSON-Schema converter
Redfish Resource Explorer

- Browser-based Educational tool part of the DMTF web site for Redfish
- Explore “mockups” of the Redfish data model
- Navigate via links through the model to various resources
- Text descriptions are taken directly from the schema files for consistency

http://redfish.dmtf.org
More information and Providing Feedback

- Download Specification and Schema: [http://www.dmtf.org/redfish](http://www.dmtf.org/redfish)
- Redfish Developer Information Site: [http://redfish.dmtf.org](http://redfish.dmtf.org)
- BrightTalk webinars: [https://www.dmtf.org/education/webinars](https://www.dmtf.org/education/webinars)
  - Introduction to Redfish (25min)
  - Redfish Data Model Deep Dive (55min)
  - Modeling the Redfish Way (60min)
- Provide feedback through the DMTF feedback portal, on both published specification and “Work in Progress”: [http://www.dmtf.org/standards/feedback](http://www.dmtf.org/standards/feedback)
- Coming Soon – public User Group / Forum
- Join the SPMF
  - By Joining the DMTF and SPMF, you can shape the standard
  - [http://www.dmtf.org/join/spmf](http://www.dmtf.org/join/spmf)
Q&A & Discussion

Redfish

www.dmtf.org
Introduction to the Redfish data model

- All resources linked from a Service Entry point (root)
  - Always located at URL: /redfish/v1/
- Major resource types structured in ‘collections’ to allow for standalone, multi-node, or aggregated rack-level systems
  - Additional related resources fan out from members within these collections

- **ComputerSystem**: properties expected from an OS console
  - Items needed to run the “computer”
  - Roughly a logical view of a computer system as seen from the OS

- **Chassis**: properties needed to locate the unit with your hands
  - Items needed to identify, install or service the “computer”
  - Roughly a physical view of a computer system as seen by a human

- **Managers**: properties needed to perform administrative functions
  - aka: the systems management subsystem (BMC)
Resource map (highlights)

/redfish/v1/
Root Resource
Links to all content

/redfish/v1/Managers
Collection of Managers
BMC functionality

/redfish/v1/Managers/<id>
BMC
System Manager operations

/redfish/v1/Systems
Collection of Systems
“Logical” view of the system

/redfish/v1/Systems/<id>
Server Information
Model #, Serial #, Boot Order, NIC MAC, status, etc.

/redfish/v1/Chassis
Collection of Chassis
“Physical” view of the system

/redfish/v1/Chassis/<id>
Chassis
Chassis global physical asset info

/jsonschemas

/eventservice

/sessionService

/accountService

/manager

/bmc

/disk

/networkinterface

/power

/thermal

/logs
UPDATE

High Performance Computing Power Application Programming Interface (API) Specification

James H. Laros III
Sandia National Laboratories
http://powerapi.sandia.gov
Version 1.2 Released February 2016
- Changes were predominately a result of our collaboration with Cray on the Trinity Power APM NRE

Version 1.3 *to-be-released* May 2016
- Statistics interface
- New Object (HT)
- New Attribute (GOV)

Version 2.0 *release-TBD* 2016 Python bindings

Trinity APM NRE
- Cray and Adaptive collaborations

Discussions with Geo

Discussions with Redfish
Trinity APM NRE

- **APM** – Advanced Power Management
- **NRE** – Non-recurring Engineering
  - Investment part of the NNSA Advanced Technology System platform acquisition strategy
- 3 General Areas:
  - **Cray: Power Management Database (PMDB) interface**
    - Provide access to Cray’s PMDB
    - Core and selected higher-level interfaces
    - Implemented in Python – to-be released in Version 2.0 of specification
  - **Cray: Compute node interface**
    - Node level C implementation
    - Core and selected higher-level interfaces
  - **Adaptive Computing: Power Aware Scheduling**
    - Exercise certain aspects of Cray’s implementation
    - Use case driven - scheduling within power constraints

*All capabilities will be implemented on the production Trinity Platform*
Power API and Geo Alignment

- **Power API Today:**
  - Application “Hints” Interface
    - Recognize or hint about application regions
      - E.g. compute, IO, parallel, serial

- **Geo Today:**
  - Labeling regions which allows
    - I’m repeating a previous region
    - Time a region
    - Evolving or “learning” about regions

- **Power API Tomorrow:**
  - Evolve the Hints interface (or create new interfaces) to align with and expose Geo capabilities

- **Good example of portability across runtime layers**
  - Geo as the underlying runtime
  - Portable to “other” runtimes with similar capabilities
Power API and Redfish Alignment

- **Power API Today: core interfaces**
  - “Common” set of interfaces for power measurement and control

- **Redfish Today:**
  - Provides out-of-band abstracted interface to the underlying implementation
    - Replaces IPMI-over-LAN
  - Inband later? (see Redfish talk)

- **Power API Tomorrow:**
  - Tools built using Power API interfaces could be portably used on systems that implement Redfish
  - Tools remain portable to other systems
  - Also to other Role->System pairs that are not applicable to Redfish

- **Another example of enabling portability**
  - This time using an emerging DMTF standard underneath
Going Forward

- How do we move forward?
- What “standards” model to apply?
- Regular calls?
  - Frequency?
- Face to Face meetings
  - Frequency?
- Important to have broad community participation which includes vendor representatives
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
Overview

- This will be a 10,000 foot view
  - The specification is necessarily broad in scope
  - Covering the specification in detail takes many hours
- A bit of history
- Collaboration from the start
- Important core principles
- Some higher level concepts
- Moving forward
Who is Behind PowerAPI?

James H. Laros III, David DeBonis, Ryan Grant, Suzanne M. Kelly, Michael Levenhagen, Stephen Olivier, Kevin Pedretti
A UML-ish Approach

- 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
- Cite use case document
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
    - More about Roles later

- 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!
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
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

Considers all users of HPC platform - people and programs

- **Core (Common)**
  - 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
Roles

PWR_Role

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;
 Roles

- **Application** – Application or application library executing on the compute resource; May include run-time components running in user space
- **Monitor and Control** -- Cluster management or Reliability Availability and Serviceability (RAS) systems, for example.
- **Operating System** -- Linux or specialized lightweight kernels and privileged portions of run-time systems. Privilege escalation layer.
- **User** -- The end user of the HPC platform.
- **Resource Manager** – Can include work load managers, schedulers, allocators and even portions of run-time systems that manage resources.
- **Administrator** – System administrator or day-to-day platform manager.
- **HPCS Manager** -- Individual(s) responsible for managing policy for the HPC platform, often through scheduling policy. Implements facility parameters.
- **Accounting** -- Individual or software that produces reports of metrics for the HPC platform.
System Description

Presents a navigable view of the system’s hardware components (objects) upon initialization
• Can extend to custom object types
• Can be heterogeneous

PWR_ObjType

typedef enum {
    PWR_OBJ_PLATFORM = 0,
    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_NUM_OBJ_TYPES,
    /* */
    PWR_OBJ_INVALID = -1,
    PWR_OBJ_NOT_SPECIFIED = -2
} PWR_ObjType;
Objects

- **Objects and Groups**
  - Objects represent components of a system
    - Lots of flexibility in what a component is
  - System Description is the organization of these objects to represent the system
  - Representation may be dependent on Role (and other considerations)
  - Attributes (later) are associated with objects
  - Groups can be implementation (predefined) or user defined (long or short lived)

- **Navigate the system description provided upon Initialization**
  - Entry point may depend on Role
    - Node entry point for Application
    - Platform entry point for Administrator
  - Navigate up (parent object), down (child objects)
  - Navigation can also be thought of as Discovery
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_CURRENT, /* double, amps */
    PWR_ATTR_VOLTAGE, /* double, volts */
    PWR_ATTR_POWER, /* double, watts */
    PWR_ATTR_POWER_LIMIT_MIN, /* double, watts */
    PWR_ATTR_POWER_LIMIT_MAX, /* double, watts */
    PWR_ATTR_FREQ, /* double, Hz */
    PWR_ATTR_FREQ_LIMIT_MIN, /* double, Hz */
    PWR_ATTR_FREQ_LIMIT_MAX, /* double, Hz */
    PWR_ATTR_ENERGY, /* double, joules */
    PWR_ATTR_TEMP, /* double, degrees Celsius */
    PWR_ATTR_OS_ID, /* uint64_t */
    PWR_ATTR_THROTTLED_TIME, /* uint64_t */
    PWR_ATTR_THROTTLED_COUNT, /* uint64_t */
    PWR_NUM_ATTR_NAMES,
    /* */
    PWR_ATTR_INVALID = -1,
    PWR_ATTR_NOT_SPECIFIED = -2
} PWR_AttrName;
Attribute Interface

MEASURE

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

CONTROL

```c
int PWR_ObjAttrSetValue( PWR_Obj object,
    PWR_AttrName attr,
    void* buf);
```

Symmetric calls available for operating on groups of objects
Attributes: Common Functionality

- **Attributes** (measure and control) of objects and groups of objects
  - Access dependent on Role and other implementation specific considerations
- **Get and Set operations** enable basic measurement and control for the exposed object attributes (and groups of objects)
- Attributes can represent generic measurement and control features
  - Power, Voltage, Current, Frequency
- Architecture specific features are permissible at the lowest levels
  - Pstate, Cstate may not be meaningful for all architectures
- An attribute, like power, can represent an instrumentation point or a summation of underlying instrumentation points
  - Power attribute of a CPU object
  - Power attribute of a Node object
typedef enum {
    PWR_MD_NUM = 0, /* uint64_t */
    PWR_MD_MIN, /* either uint64_t or double, depending on attribute type */
    PWR_MD_MAX, /* either uint64_t or double, depending on attribute type */
    PWR_MD_PRECISION, /* uint64_t */
    PWR_MD_ACCURACY, /* double */
    PWR_MD_UPDATE_RATE, /* double */
    PWR_MD_SAMPLE_RATE, /* double */
    PWR_MD_TIME_WINDOW, /* PWR_Time */
    PWR_MD_TS_LATENCY, /* PWR_Time */
    PWR_MD_TS_ACCURACY, /* PWR_Time */
    PWR_MD_MAX_LEN, /* uint64_t, max strlen of any returned metadata string. */
    PWR_MD_NAME_LEN, /* uint64_t, max strlen of PWR_MD_NAME */
    PWR_MD_NAME, /* char *, C-style NULL-terminated ASCII string */
    PWR_MD_DESC_LEN, /* uint64_t, max strlen of PWR_MD_DESC */
    PWR_MD_DESC, /* char *, C-style NULL-terminated ASCII string */
    PWR_MD_VALUE_LEN, /* uint64_t, max strlen returned by PWR_MetaValValueAtIndex */
    PWR_MD_VENDOR_INFO_LEN, /* uint64_t, max strlen of PWR_MD_VENDOR_INFO */
    PWR_MD_VENDOR_INFO, /* char *, C-style NULL-terminated ASCII string */
    PWR_MD_MEASURE_METHOD, /* uint64_t, 0/1 depending on real/model measurement */
    PWR_NUM_META_NAMES,
    /* */
    PWR_MD_INVALID = -1,
    PWR_MD_NOT_SPECIFIED = -2
} PWR_MetaName;
Metadata: Common Functionality

- **Metadata** interface provides information about quality, frequency, and other characteristics associated with attributes of objects
  - Can be specific for a particular attribute/object pair
    - All power sensors might not provide the same accuracy
  - Frequency of sample collection can help determine usefulness of data
  - Can also, in some cases, set metadata
    - Potentially to change how a device responds
Statistics Interface

PWR_AttrStat

typedef enum {
    PWR_ATTR_STAT_MIN = 0,
    PWR_ATTR_STAT_MAX,
    PWR_ATTR_STAT_AVG,
    PWR_ATTR_STAT_STDEV,
    PWR_ATTR_STAT_CV,
    PWR_NUM_ATTR_STATS,
    /* */
    PWR_ATTR_STAT_INVALID = -1,
    PWR_ATTR_STAT_NOT_SPECIFIED = -2
} PWR_AttrStat;
Statistics: Common Functionality

- **Statistics** interface gathers data on one or more attributes for an object or group of objects
  - Real time or historic statistics
    - Historic implies data retention (database for example)
  - Min, Max, Average, Standard Deviation, Coefficient of Variation
  - Reduction operation available
  - User provided statistic function on the to-do list

- Provides functions to...
  - Start, stop, and reset statistics gathering
  - Get the calculated value(s) for the object or group of objects
  - Reduce the values calculated for objects in a group into a single value
High Level Interfaces by Role (1)

Administrator:
- Apply Hard Power Limits based on Facility parameters
  - Bounds Power Aware Scheduling

Accounting:
- Power/Energy Application profiling based on historic information
  - Feeds into Power Aware Scheduling
High Level Interfaces by Role (2)

Resource Manager
- Power Aware Scheduling
  - HPC Tetris

User
- Understands application Power and Energy Characteristics and Phases

Application
- Provide application hints based on profiling
One Implementation Across Multiple Interfaces

- Resource Mgr
- Acct Mgr
- Administrator
- User

Decision based on what ROLE is asking

Single API Implementation

Decision based on where data exists

- Monitor and Control
- Resource Mgr

- Database
- Database
Reference Implementation

Available online and open source: http://github.com/pwrapi
Power API Timeline

- 2013: Use case document prepared by SNL and NREL and reviewed by partners

- July 2014: Draft specification review meeting with cross-vendor panel of experts


- Sept. 2014: Day-long community launch meeting with labs, industry, academia

- Jan. 2015: Prototype implementation release

- June 2015: Reference implementation release (http://github.com/pwrapi)

- Aug. 2015: Specification v1.1 release

81 Countries as of October 2015
Trinity ATS-1 NRE: Advanced Power Management

- DOE NNSA’s Advanced Technology System (ATS-1)
  - >19,000 Nodes, <10MW

- Introduced the concept of funding Non-Recurring Engineering (NRE) projects to advance important technologies in conjunction with platform procurement

- Cray contracted to address Advanced Power Management
  - Implement portions of the Power API at scale

Targeting two areas of the API:

1. Implement interface to Power Management Database
   - Extend specification to include Python
   - Monitor and Control in our diagram

2. Compute node implementation (native C)
Going Forward

- How do we move forward?
- What “standards” model to apply?
- Regular calls?
  - Frequency?
- Face to Face meetings
  - Frequency?
- Important to have broad community participation which includes vendor representatives
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