Exceptional service in the national interest

Sandia National Laboratories

Power API for HPC: Standardizing Power Measurement and Control

Speaker: Stephen Olivier

Power API Team: James H. Laros III (Lead), Kevin Pedretti, Suzanne M. Kelly, Michael Levenhagen, David DeBonis, Stephen L. Olivier, Ryan E. Grant

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Images courtesy of: Oak Ridge National Laboratory and Cray Inc.
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. SAND NO. 2015-XXXX

Outline

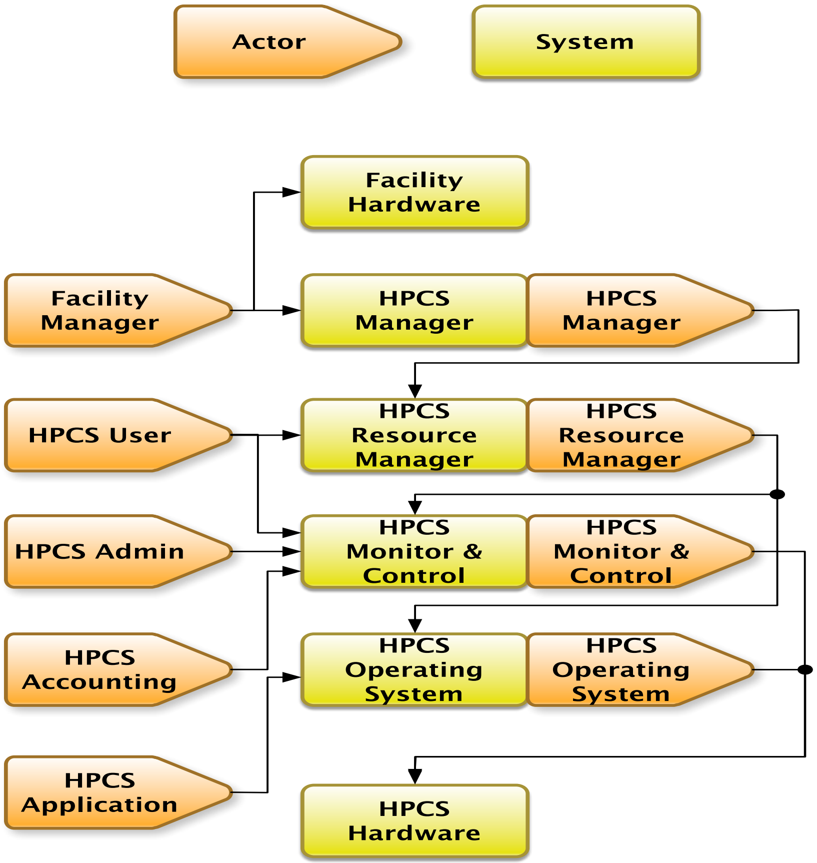
- What is the Power API?
- Overview of API Features and Interfaces
- Reference Implementation and Timeline

What is the Power API?

- The Power API is a comprehensive system software API for interfacing with power measurement and control hardware
- Designed to be comprehensive across many different levels of a data center
- Many different actors can interface with a single API to perform several different roles
- Encompasses facility level concerns down to low level software/hardware interfaces

What is the Power API?

- Broad scoped, portable API
- Multiple actors can interact with the system at different levels
- Each interaction represents an interface that is defined in the Power API



Example Use Cases

- Control power in a hardware overprovisioned system with a given MW power cap
- Accounting and prediction of power load for cooperation with power utilities
- Oversight entities wish to have long term historical power/ energy data for the platform
- Users wish to monitor their jobs on fine-grained scales to understand/improve power/energy consumption
- Enables studies of whole system power/energy consumption

Roles

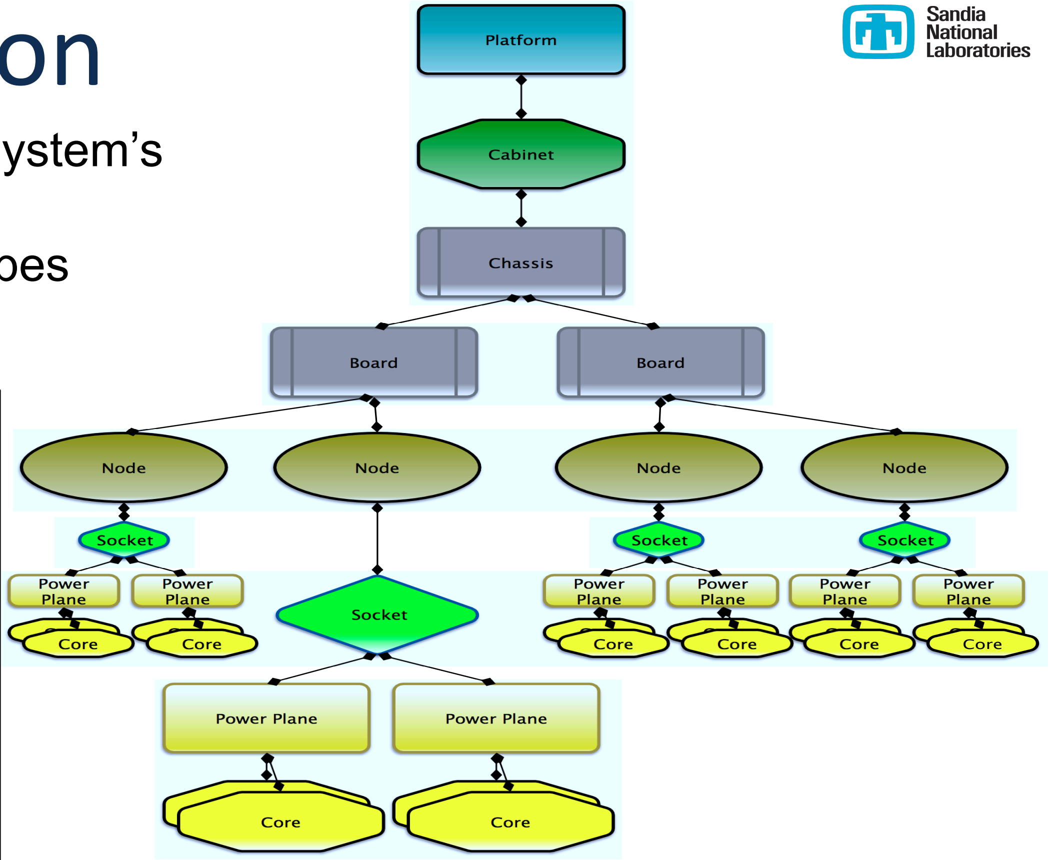
- Application – Application or application library executing on the compute resource; May include run-time components running in user space
- Monitor and Control -- Cluster management or Reliability Availability and Serviceability (RAS) systems, for example.
- Operating System -- Linux or specialized lightweight kernels and privileged portions of run-time systems.
- 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 -- Responsible for managing policy for the HPC platform.
- Accounting – Individual/software that produces reports for the platform.

System Description

Presents a navigable view of the system’s hardware components

- Can extend to custom object types
- Can be heterogeneous

```
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;
```



Example System Description

Common Functionality

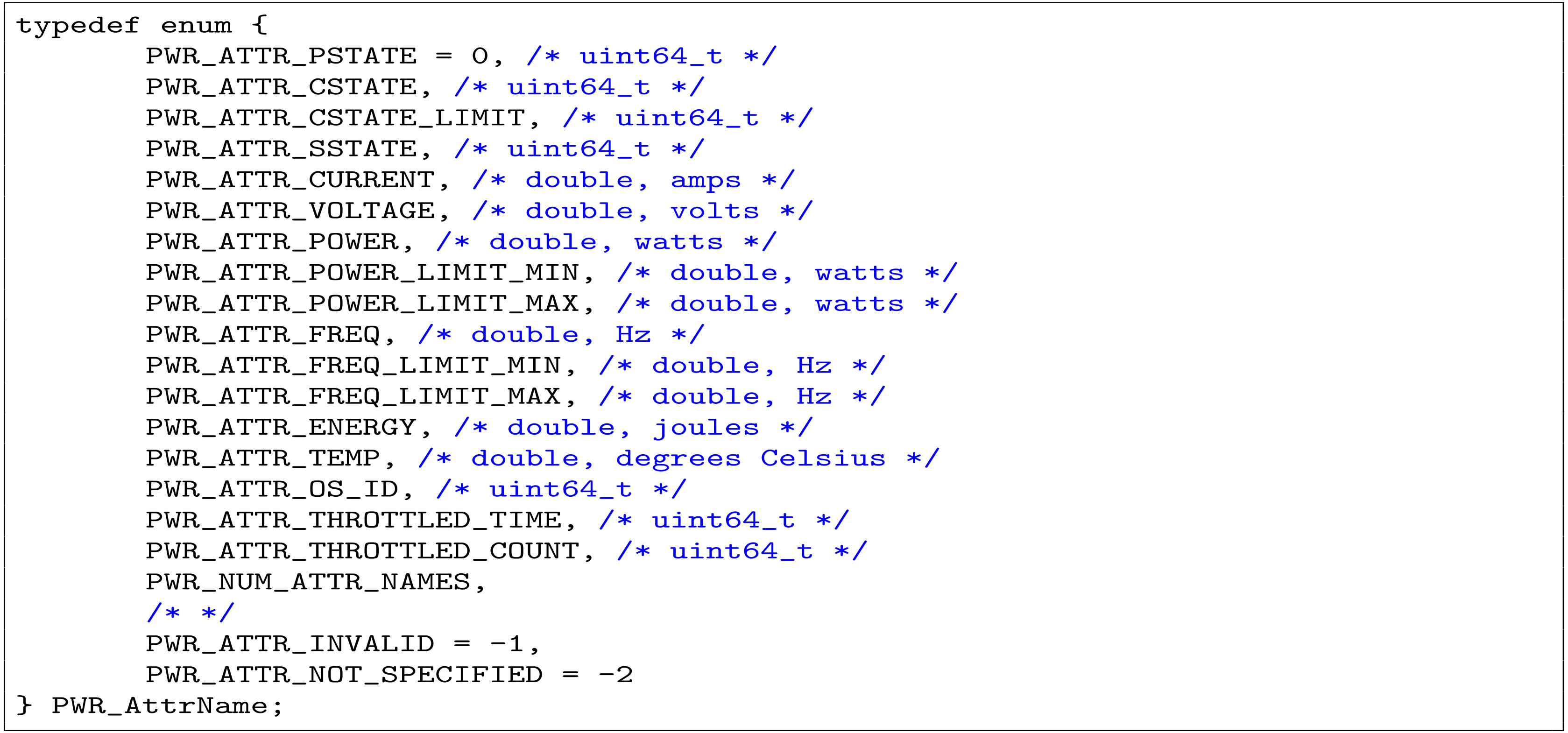
- Navigation across and grouping of objects in the system
- Attributes (e.g., power cap, voltage) for the objects can be accessed depending on role (e.g., user, app, OS, admin)
- Getters/setters enable basic measurement and control for the exposed object attributes
- Metadata interface provides information about quality, frequency, and other characteristics of measurement/control
- Statistics interface gathers data on one or more attributes for an object or group of objects over time

Navigation and Grouping of Objects

- Entry point into the system can depend on role
  - E.g., node level for an application and platform level for an admin
  - Functions are provided to navigate up to the parent object or down to child objects in in the hierarchy

- Many functions are provided to provide measurement and control of groups of objects
  - User of the API can create groups and perform set operations
  - Implementation may provide predefined groups for convenience

Attributes of Objects



```
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;
```

Accessing Attributes of Objects

MEASURE

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

CONTROL

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

Statistics Interface

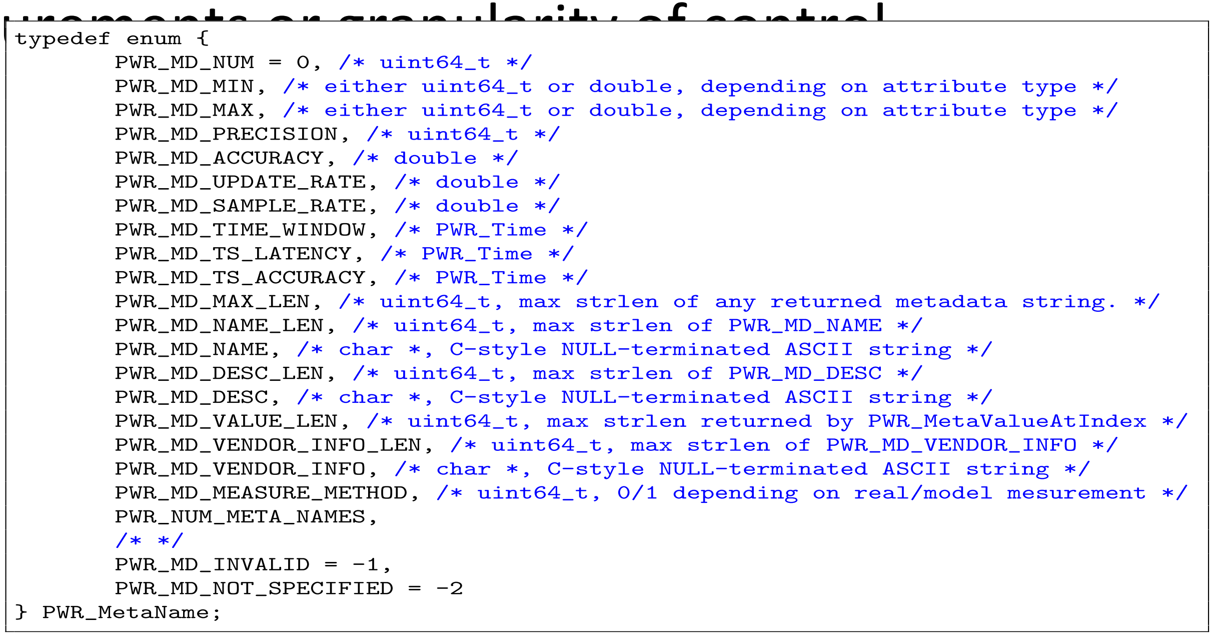
- While attribute getter functions return point values, the statistics interface gathers statistics from samples gathered over time

- Can specify min, max, average, standard deviation
  - Vendors may extend to support other statistics

- 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

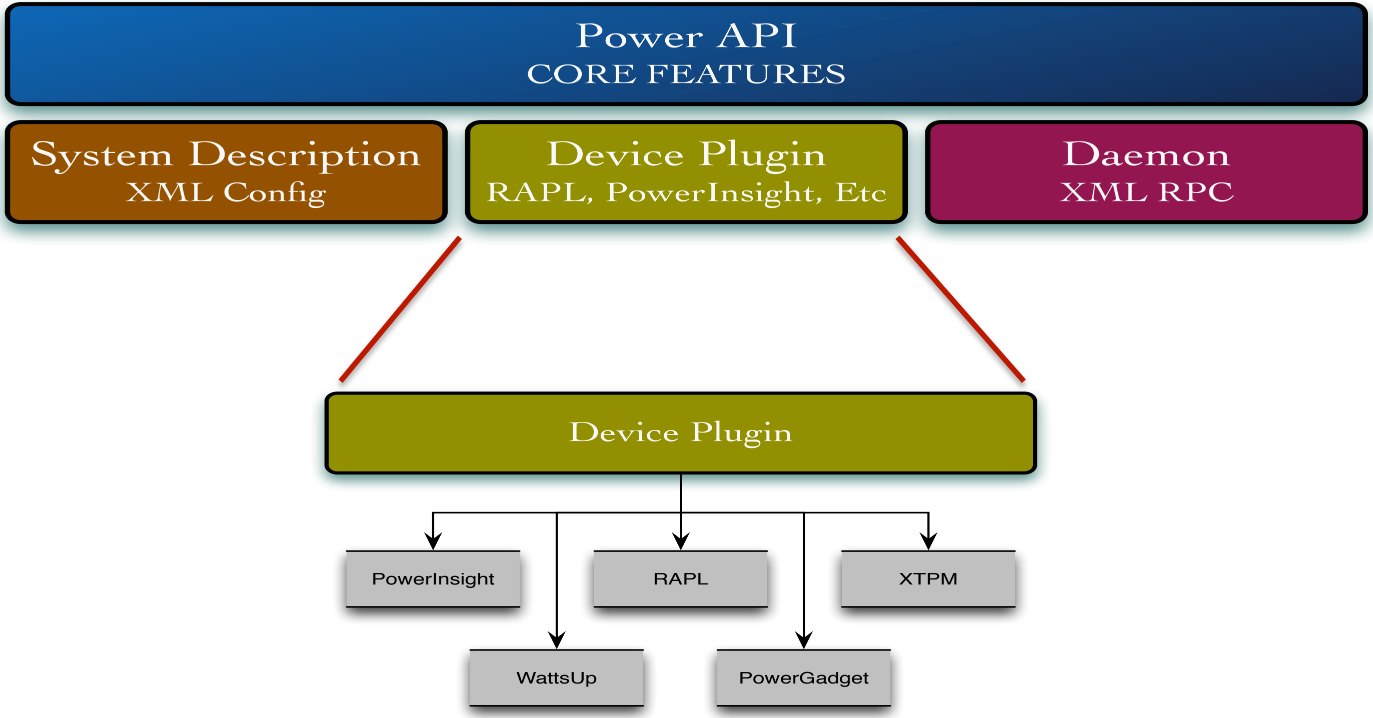
Metadata Interface

- Allows querying and, in some cases manipulation, of characteristics of objects and their attributes, e.g. quality of measurements or granularity of control



```
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_MetaValueAtIndex */
        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 mesurement */
        PWR_NUM_META_NAMES,
        /* */
        PWR_MD_INVALID = -1,
        PWR_MD_NOT_SPECIFIED = -2
} PWR_MetaName;
```

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
- Aug. 2014: Specification v1.0 release (http://powerapi.sandia.gov/)
- 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
- Oct. 2015: Specification v1.1a release (http://powerapi.sandia.gov/)

Who is Behind PowerAPI?

IBM

NREL
NATIONAL RENEWABLE ENERGY LABORATORY

Los Alamos
NATIONAL LABORATORY
EST. 1943

intel®

Lawrence Livermore
National Laboratory

PENGUIN COMPUTING
EXPERTS IN HIGH PERFORMANCE COMPUTING SOLUTIONS

NERSC

Adaptive
COMPUTING

Sandia
National
Laboratories

AMD

UNM

hp

CRAY
THE SUPERCOMPUTER COMPANY

E E H P C
ENERGY EFFICIENT HIGH PERFORMANCE COMPUTING

<Your logo here!>

Thank you

Power API

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

Online Survey

Sandia National Laboratories

http://bit.ly/sc15bof

Your feedback is appreciated!