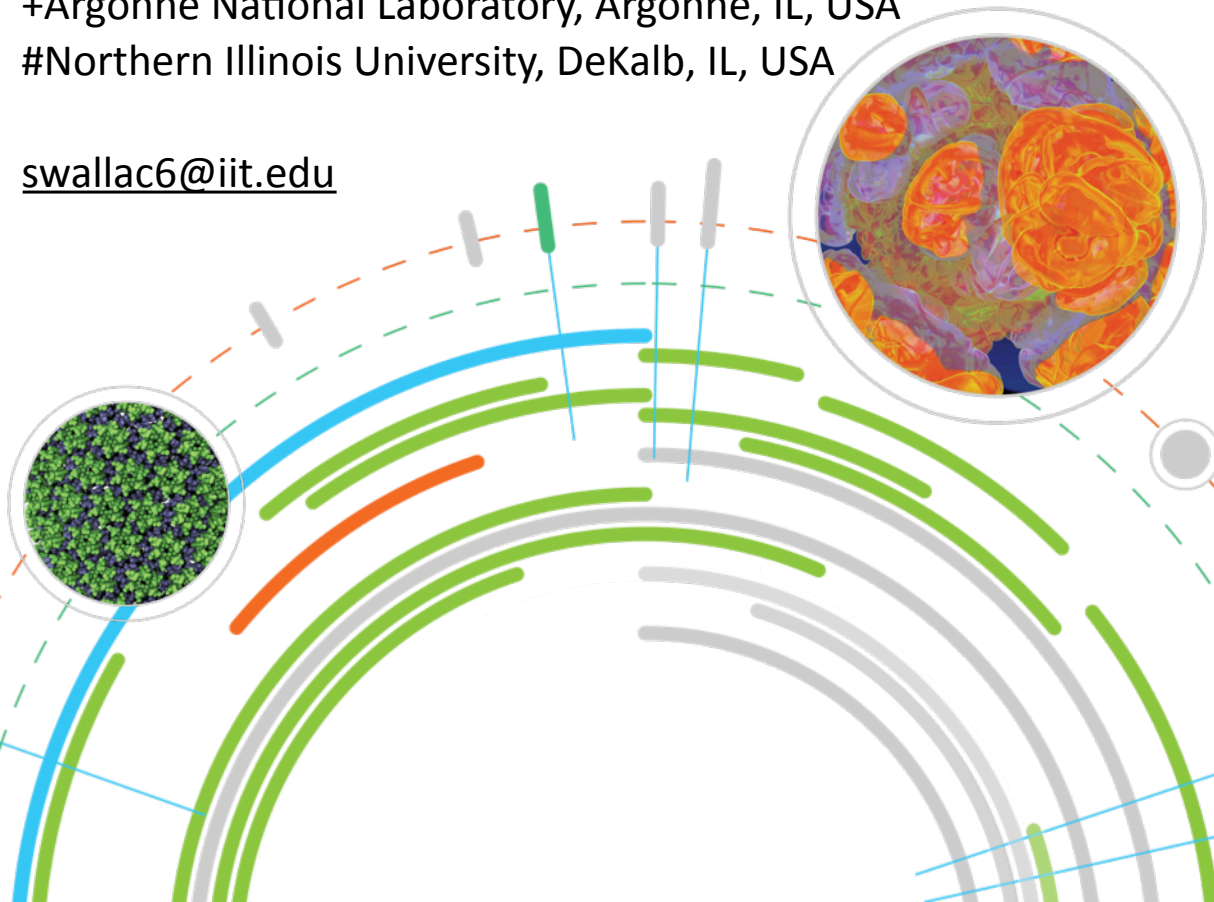
Comparison of Vendor Supplied Environmental Data Collection Mechanisms

Sean Wallace[*], Venkatram Vishwanath[+], Susan Coghlan[+], Zhiling Lan[*], Michael E. Papka[+#]

*Illinois Institute of Technology, Chicago, IL, USA
+Argonne National Laboratory, Argonne, IL, USA
#Northern Illinois University, DeKalb, IL, USA

swallac6@iit.edu



Argonne Leadership Computing Facility

Argonne NATIONAL LABORATORY

Motivation

- Systems becoming larger and more complex.
  - Makes it more difficult to get accurate picture of what "environmental" aspects (e.g., motherboard, CPU, GPU, hard disk and other peripherals temperature, voltage, current, fan speed, etc.) are like.
  - Putting accurate sensors in hardware is expensive, therefore hardware manufacturers do so sparingly and where only necessary.
- Lack of tools available to access and interpret environmental data across variety of systems.
  - Data like power consumption, temperature, etc. are some metrics which are largely not understood at system level.
- This data can be very useful!
  - Previous studies [1] showed possibility of electricity bill savings up to 23%.

[1] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems," in Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 60:1–60:11. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503264

Outline

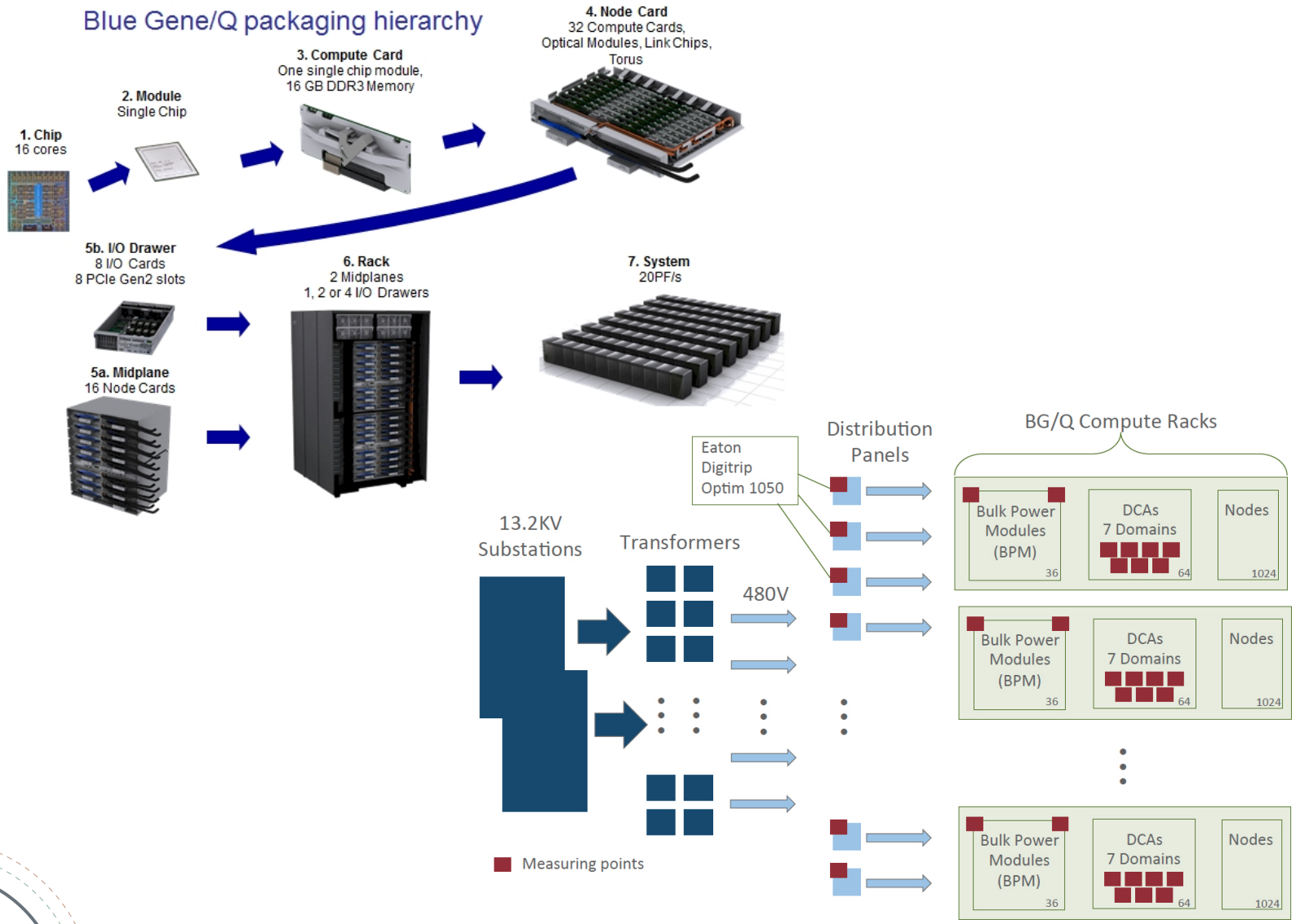
- Discussion and analysis of obtainable data from four major hardware platforms:
  - IBM Blue Gene/Q
  - Intel RAPL,
  - NVIDIA GPUs
  - Intel Xeon Phi
- With each:
  - How to collect data
  - How reliable data are
  - What frequency data can be obtained
  - Examples of data for benchmarks.
- MonEQ explanation and “how-to”

Vendor Supplied APIs

- Environmental analysis would not be possible without sensors in hardware.
- In addition to simply existing, also must have the ability to gather the information they provide.
- Good news! Every major vendor provides access to sensors:
  - Commonly accessed through low-level API, but can also be a daemon exposing a pseudo-file on virtual file system (Intel Xeon Phi) or direct access to registers (Intel processors).
- Unfortunately, no uniform way to access data across systems.
- Location of sensors determines what data is accessible.
  - Current hardware has variety here as well.
  - As such, not possible to gather exact same type of data between platforms.

	Xeon Phi	NVML	Blue Gene/Q	RAPL
Total Power				
Consumption (Watts)	✓	✓	✓	✓
Voltage	✗	✓	✓	✓
Current	✗	✓	✓	✗
PCI Express	✓	✗	✓	N/A
Main Memory	✗	✗	✓	✓
Temperature				
Die	✓	✓	✗	✗
DDR/GDDR	✓	✗	✗	✗
Device	✗	✓	✗	✓
Intake (Fan-In)	✓	✓	N/A	N/A
Exhaust (Fan-Out)	✓	✓	N/A	N/A
Main Memory				
Used	✓	✓	✓	✗
Free	✓	✓	✓	✗
Speed (kT/sec)	✓	✗	✓	✗
Frequency	✓	✗	✓	✗
Voltage	✓	✗	✓	✗
Clock Rate	✓	✓	✓	✗
Processor				
Voltage	✓	✗	✓	✓
Frequency	✓	✗	✓	✓
Clock Rate	✓	✓	✓	✓
Fans				
Speed (In RPM)	✓	✓	N/A	N/A
Limits				
Get/Set Power Limit	✓	✓	✗	✓

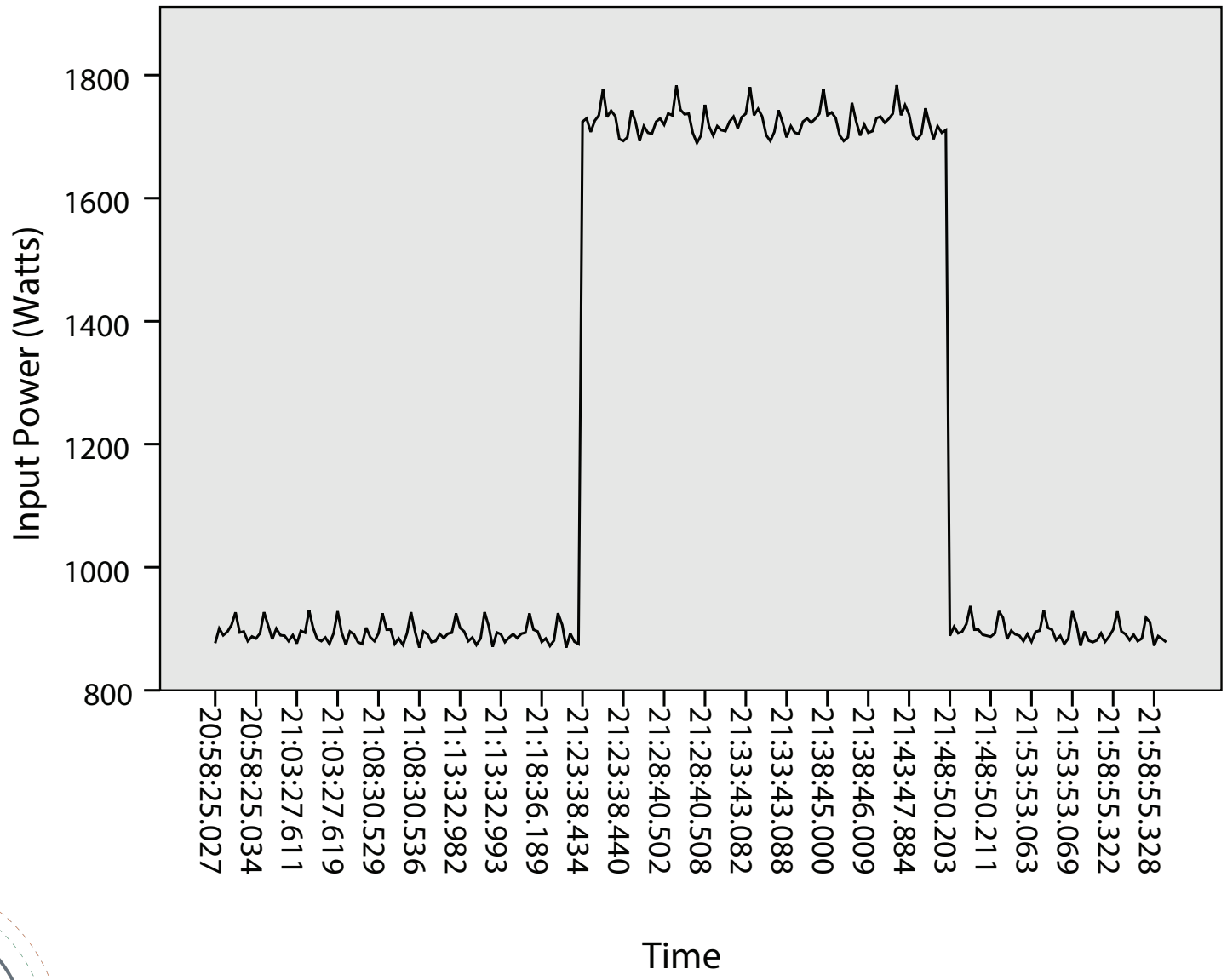
IBM Blue Gene/Q



IBM Blue Gene/Q - Environmental Database

- Blue Gene systems have environmental monitoring capabilities that periodically sample and gather environmental data from various sensors.
- This information along with timestamp and location is stored in IBM DB2 relational database—commonly referred to as the environmental database.
- Sensors are found in service cards, node boards, compute nodes, link chips, bulk power modules (BPMs), and the coolant environment.
- Depending on sensor, can be temperature, coolant flow and pressure, fan speed, voltage, and current.
- Collected at relatively long polling intervals (about 4 minutes on average).

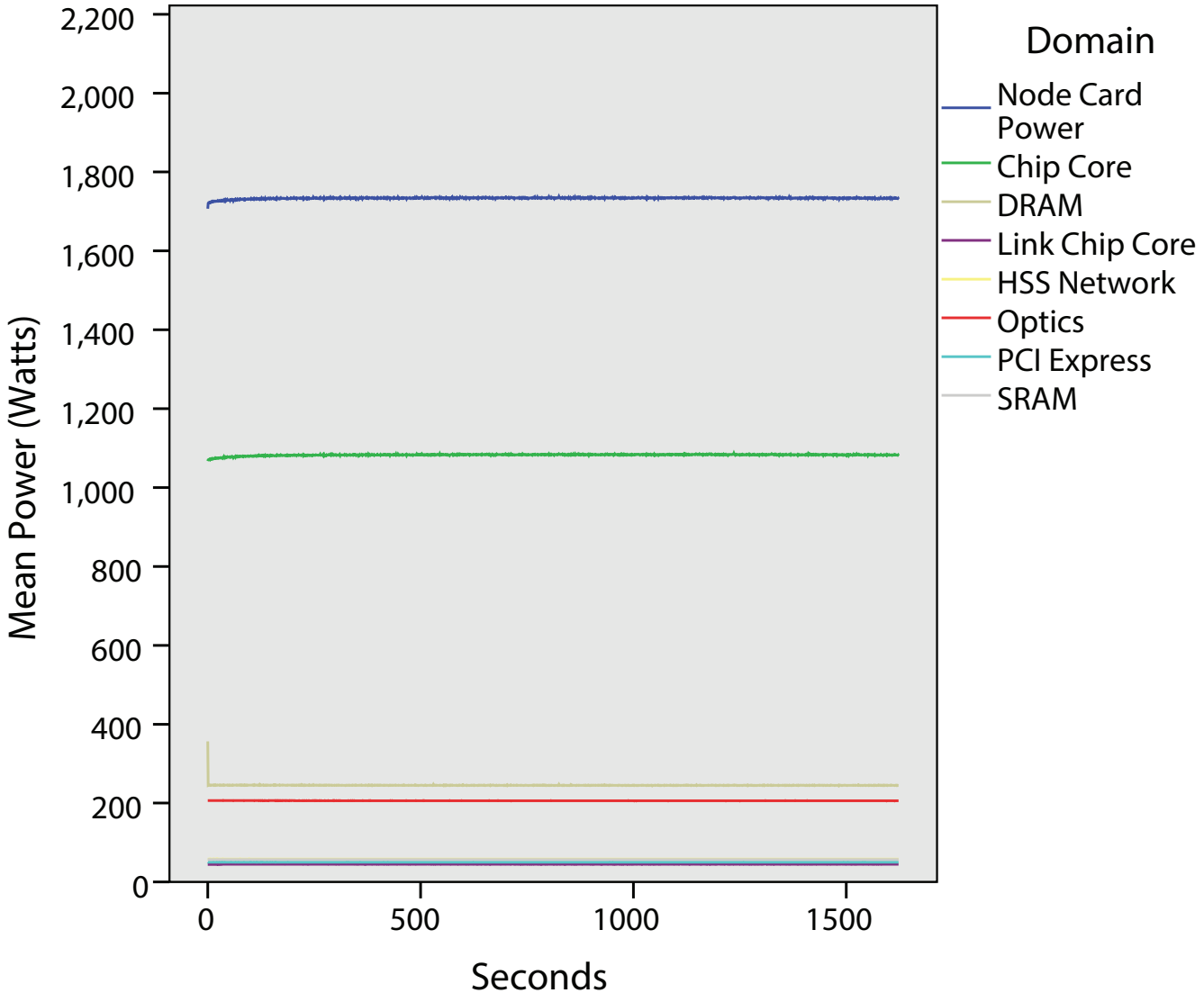
IBM Blue Gene/Q - Environmental Database



IBM Blue Gene/Q - EMON

- In addition to environmental database, IBM also provides interfaces in form of environmental monitoring API called EMON.
- Allows access to power consumption data from code running on compute nodes at much faster rate than environmental database.
- Data from EMON is *total* power from the oldest generation of power data.
- Collection is done at the node card (32 compute nodes) level for each of 7 “domains”.

IBM Blue Gene/Q - EMON



Intel RAPL

- As of the Sandy Bridge architecture, Intel has provided the "Running Average Power Limit" (RAPL) interface.
- Originally designed to provide a way to keep processors inside of a given power limit over a sliding window of time, but can also be used to calculate power consumption over time.
- Circuitry of chip is capable of providing estimated energy consumption based on hardware counters.
- Intel model-specific registers (MSRs) are implemented within x86 instruction sets to allow for access and modification of parameters.

Domain	Description
Package (PKG)	Whole CPU package.
Power Plane 0 (PP0)	Processor cores.
Power Plane 1 (PP1)	The power plane of a specific device in the encore (such as an integrated GPU-not useful in server platforms).
DRAM	Sum of socket's DIMM power(s).

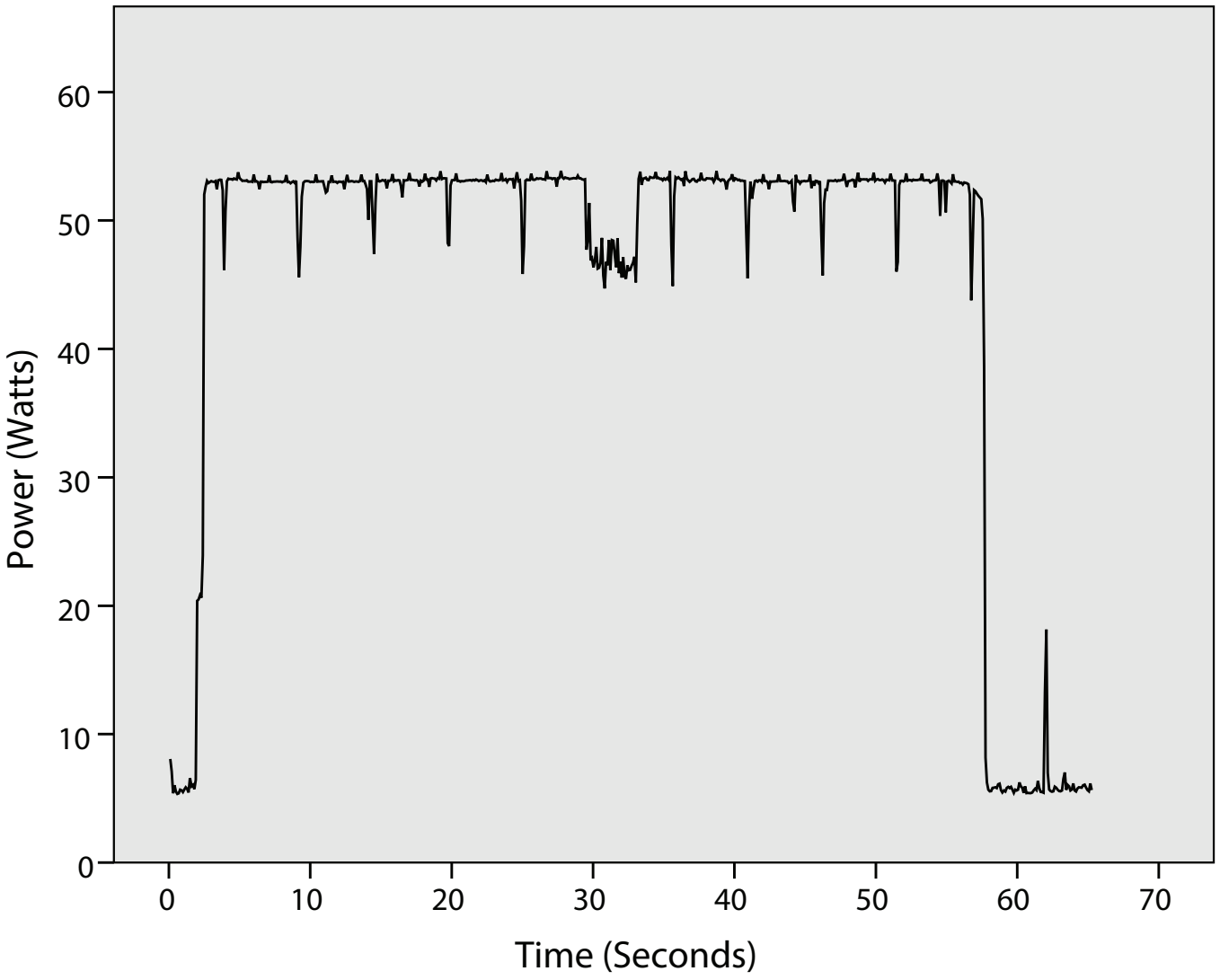
Intel RAPL

- Access to MSRs requires elevated access to the hardware, typically something only the kernel can do.
- As a result, a kernel driver is necessary to access these registers in this way.
  - As of Linux 3.14 these kernel drivers have been included and are accessible via the `perf_event` (perf) interface.
- Short of having a supported kernel, only way to access is to use Linux MSR driver which exports MSR access to userspace.
- Once built and loaded, it creases a character device for each logical processor under `/dev/cpu/*/msr`.
- Number of limitations:
  - Collected metrics are for whole socket. Therefore, not possible to collected data for individual cores.
  - DRAM memory measurements do not distinguish between channels.

Intel RAPL

- In terms of accuracy:
  - Generally concluded that updates are not accurate enough for short-term energy measurements with updates happening within the range of ±50,000 cycles.
  - However, few updates deviate beyond 100,000 cycles making RAPL interface relatively accurate for data collection at about 60ms.
  - Registers can “overfill” if they are not read frequently enough, so a sampling of more than about 60 seconds will result in erroneous data.

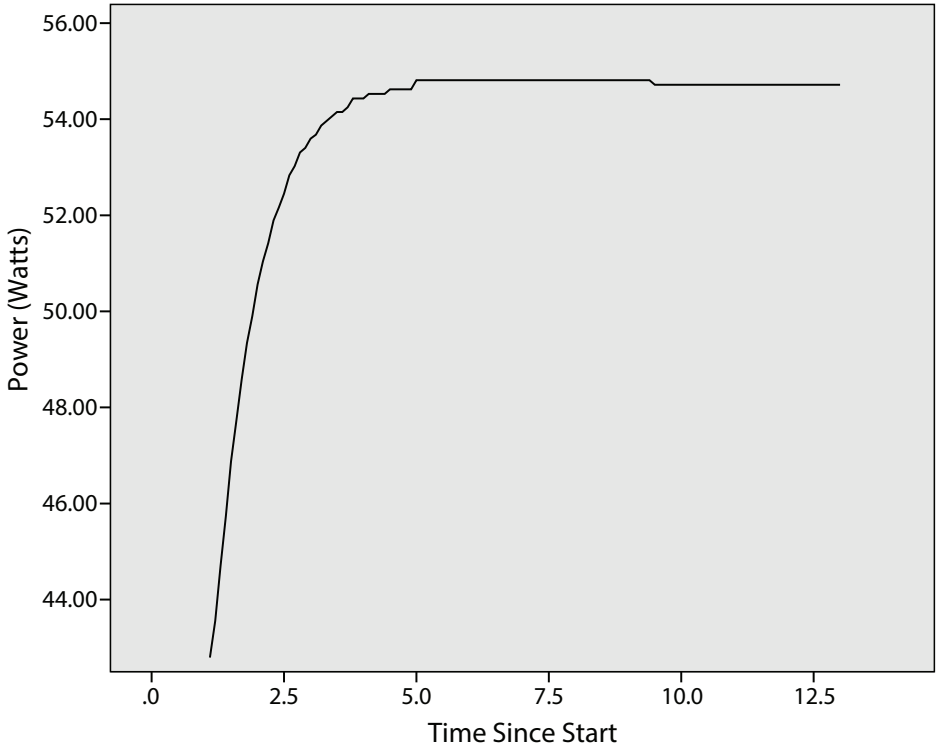
Intel RAPL



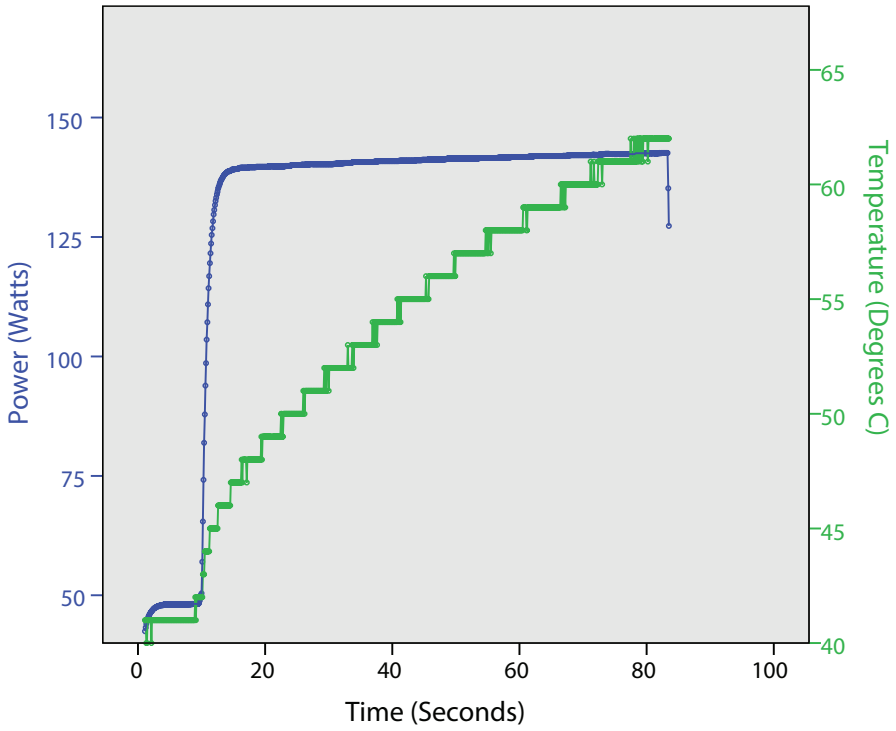
NVIDIA Management Library

- A C-based API which allows for the monitoring and configuration of NVIDIA GPUs.
- Only supported on Kepler and newer architecture (e.g., K20, K40).
- Only one call for power data collection: `nvmlDeviceGetPowerUsage()`.
- Reported accuracy by NVIDIA is ±5W with an update time of about 60ms.
- Power consumption is for entire board including memory.

NVIDIA Management Library



NOOP Workload

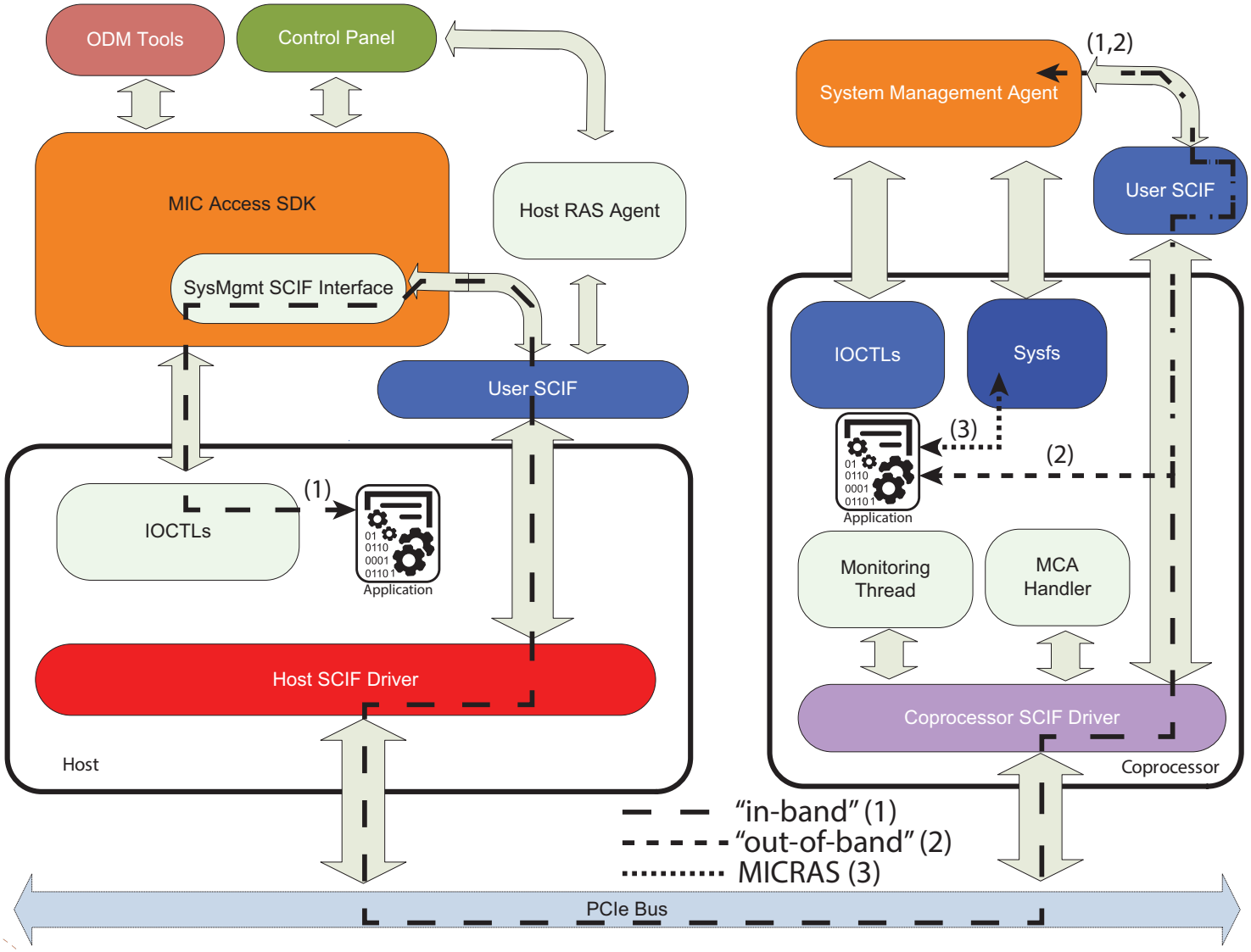


Vector Add Workload

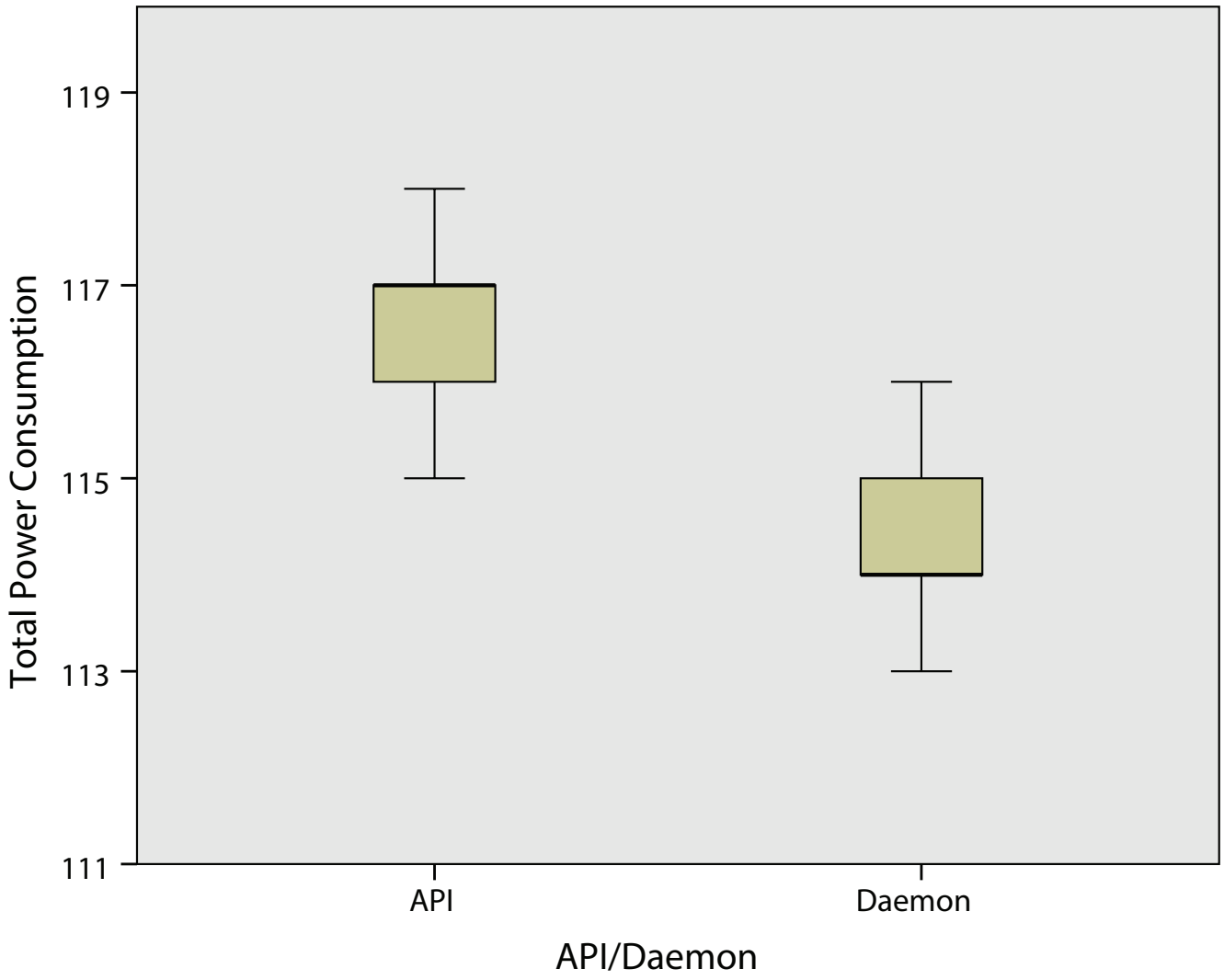
Intel Xeon Phi

- Two ways to collect data on host side:
  - In-band - uses symmetric communication interface (SCIF). Enables communication between host and device as well as device to device. Primary goal to provide uniform API for all communication across PCI Express buses. All drivers expose same interface on host and Xeon Phi, allows for software to execute where most appropriate.
  - Out-of-band - starts with same capabilities in coprocessor, but then sends information to Xeon Phi's System Management Controller (SMC). Then responds to queries from platform's Baseboard Management Controller (BMC) using intelligent platform management bus (IPMB) protocol.
- MICRAS daemon is a tool which runs on both the host and device platforms.
  - On host, allows for the configuration of the device, logging of errors, and other common administrative utilities.
  - On device, this daemon exposes access to environmental data through pseudo-files mounted on a virtual file system.
  - To read data, just read the file and parse data.

Intel Xeon Phi



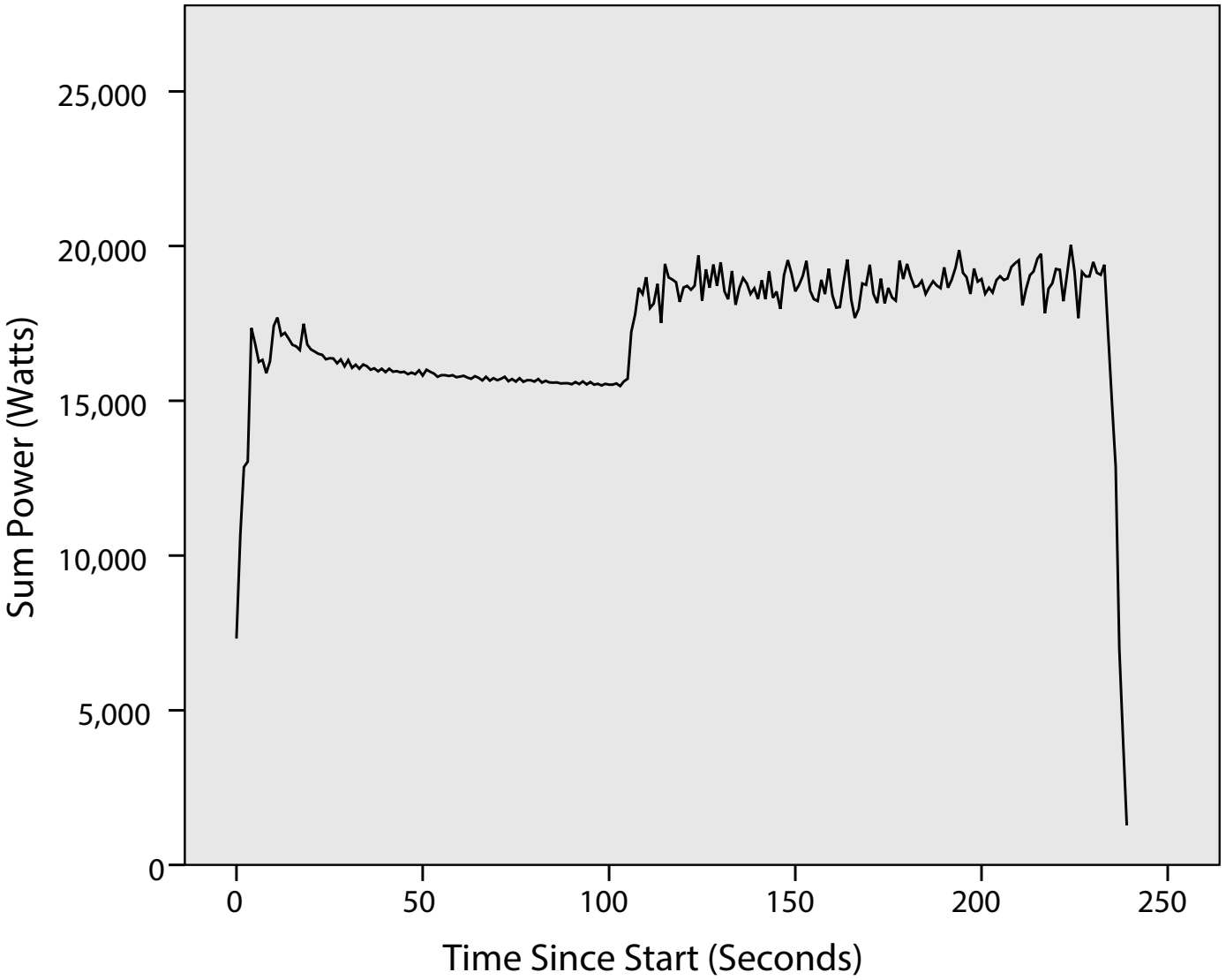
Intel Xeon Phi



Intel Xeon Phi - Tradeoffs in Approaches

- While slight, there is a significant difference between SysMgmt API and MICRAS daemon.
  - When API call is made to the lower-level library to gather data, it must travel across the SCIF to the device where user libraries call kernel functions to access registers containing data.
  - Code that wasn't already execution on device before API call must run, collect, and return, hence power difference.
- Data collected by daemon is only accessible by the portion of code running on the device.
  - Results in unavoidable overhead associated with any data collection as any collection performed must be down during the execution fo the application which is running.

Intel Xeon Phi



MonEQ

- Wanting to address limitations in other tools as well as in data collection mechanisms, we designed and developed MonEQ.
- In default mode, MonEQ pulls data from selected environmental collection interface at quickest polling interval possible for the given hardware.
- Registers to receive SIGALRM signal at polling interval. When delivered, MonEQ calls down to the appropriate interface and records data.
- Extended in this work to access data from all hardware mentioned, just link with the appropriate library.
- Supports more complex features like tagging specific areas of code.

Simple MonEQ Example

```
int status, myrank, numtasks;

status = MPI_Init(&argc, &argv);

MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

/* Setup Power */
status = MonEQ_Initialize();


/* User code */


/* Finalize Power */
status = MonEQ_Finalize();

MPI_Finalize();
```

MonEQ Overhead

- Designed to be robust without weighing down application.
- In general, overhead is dictated by number of devices being profiled.
  - Most expensive operations performed when the application isn't running (i.e., before and after execution).
- Memory overhead is essentially constant with respect to scale.

	32 Nodes	512 Nodes	1024 Nodes
Application Runtime	202.78	202.73	202.74
Initialization	0.0027	0.0032	0.0033
Finalize	0.1510	0.1550	0.3347
Collection	0.3871	0.3871	0.3871
Total	0.5409	0.5455	0.7251

Conclusions

- In many cases, the same environmental data isn't available between two different hardware platforms.
- Methods for collection can vary substantially.
- Only data point in common for all hardware discussed is total power.
- Single greatest issue which is practically impossible to eliminate is the collection overhead.
- Number of (simple) improvements can be made:
  - Stated limitations of data and the collection of data.
  - If data is to be used to compare platforms, unification is necessary.

Questions?