Power Data for HPC:
What is it?
How is it obtained?
What *use* is it?

Sean Wallace
Illinois Institute of Technology

HPC Power Management 2016

What is it?

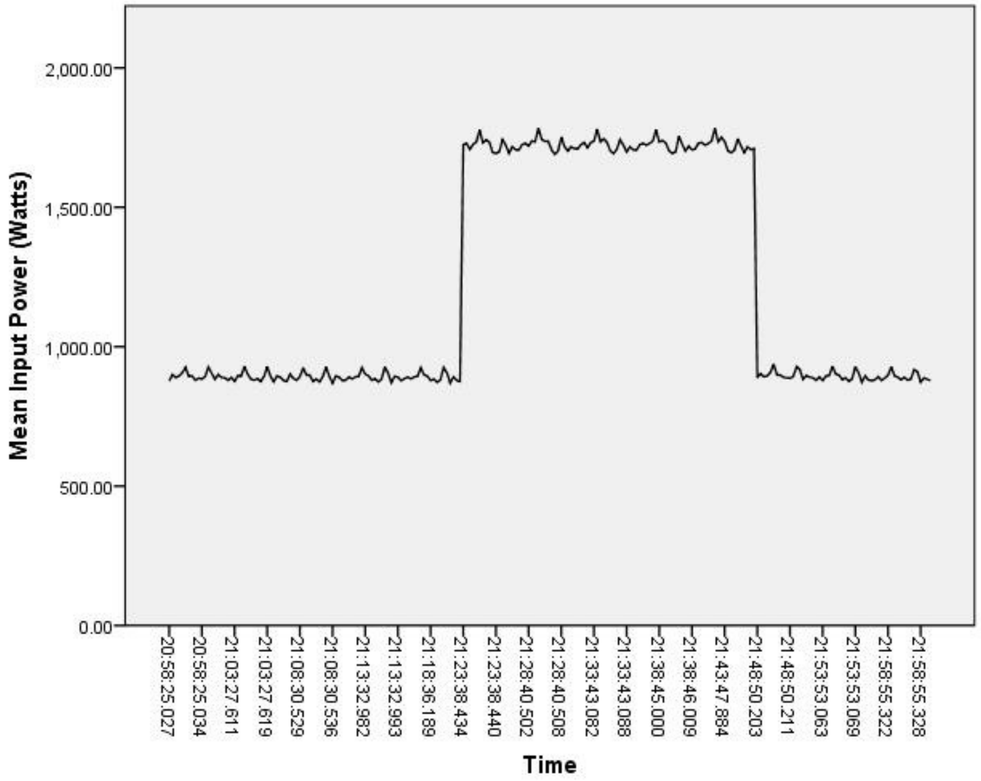
Who’s asking?

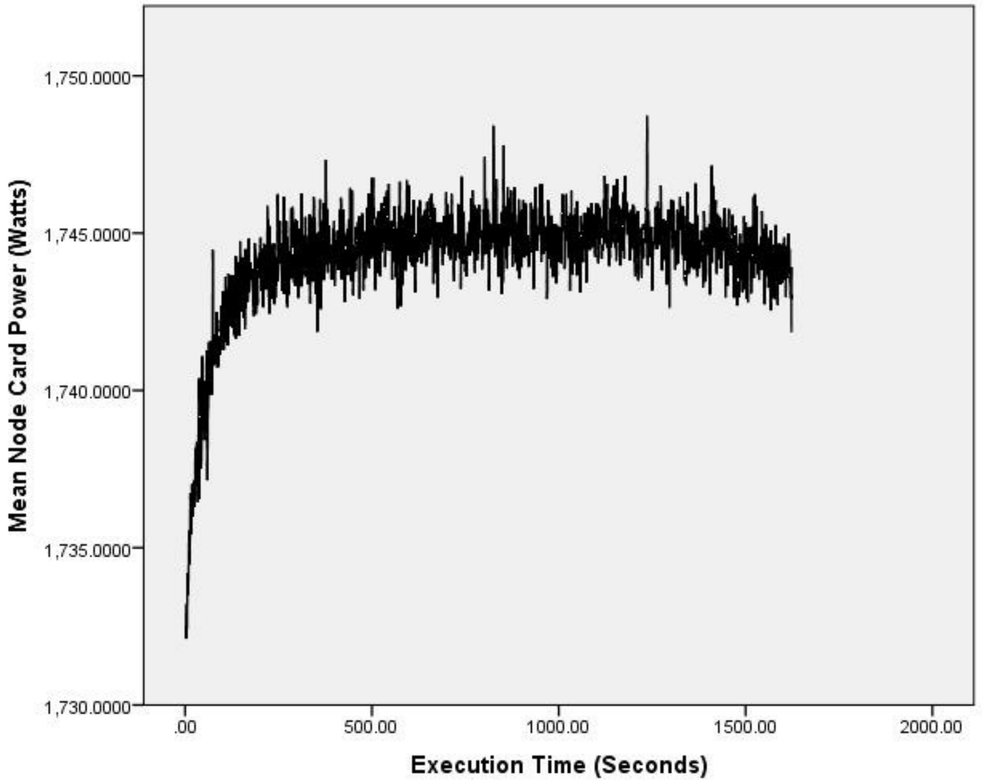
- To the pragmatist (scientist?):
  - Voltage and current at a particular time
- To the lab director:
  - That thing I would like to use less of
- To the application developer/end user:
  - What do I care!? Just make my application run faster!

What is it?

Not the same to everyone!

One size fits all?



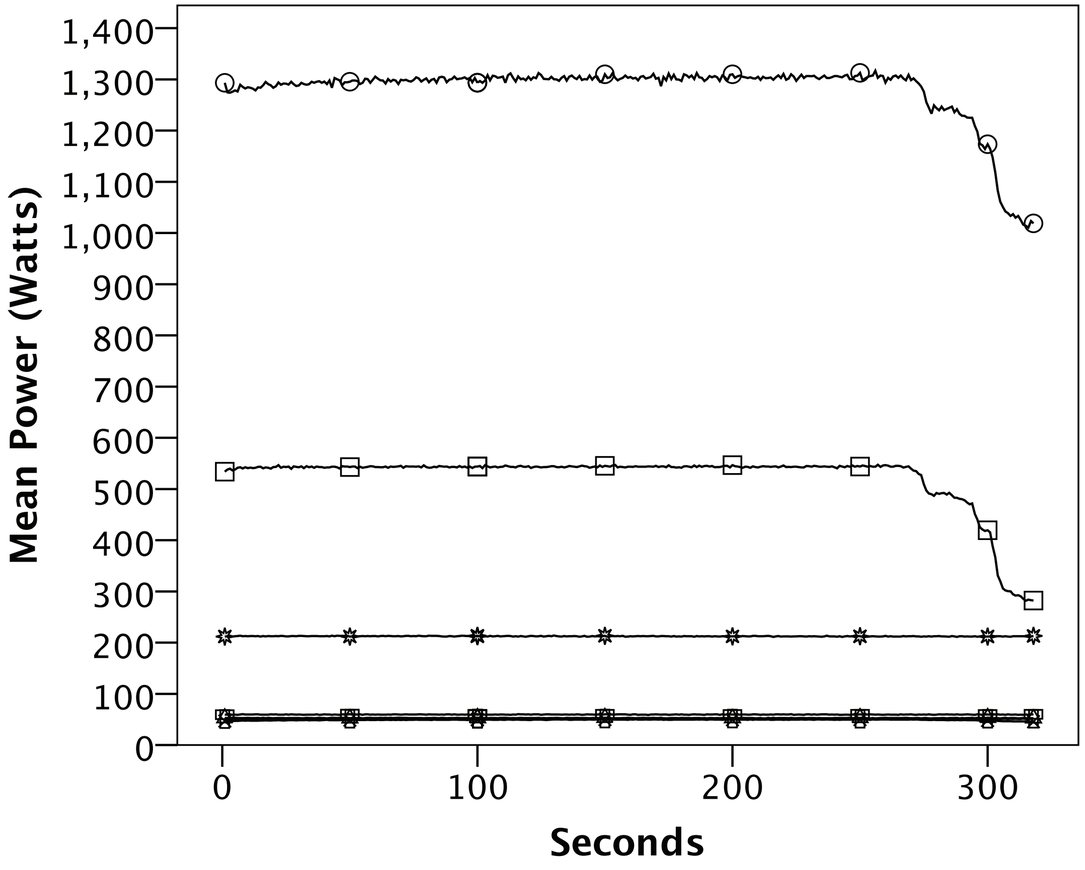


Granularity makes a big difference!

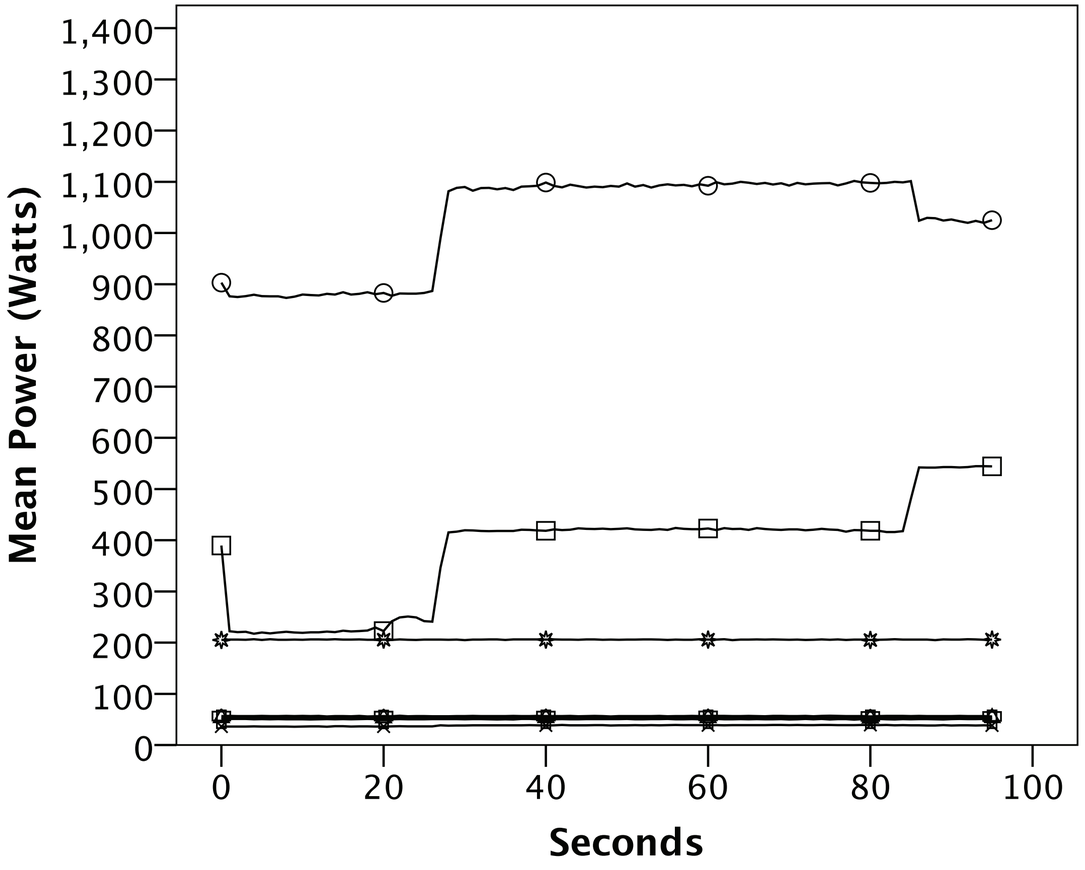
What is it?

Not the same to everyone!
Not one size fits all!

Two different applications?



DGEMM



XSBench

Application makes a big difference!

What is it?

Not the same to everyone!
Not one size fits all!
Not the same for all applications!

.
.
.

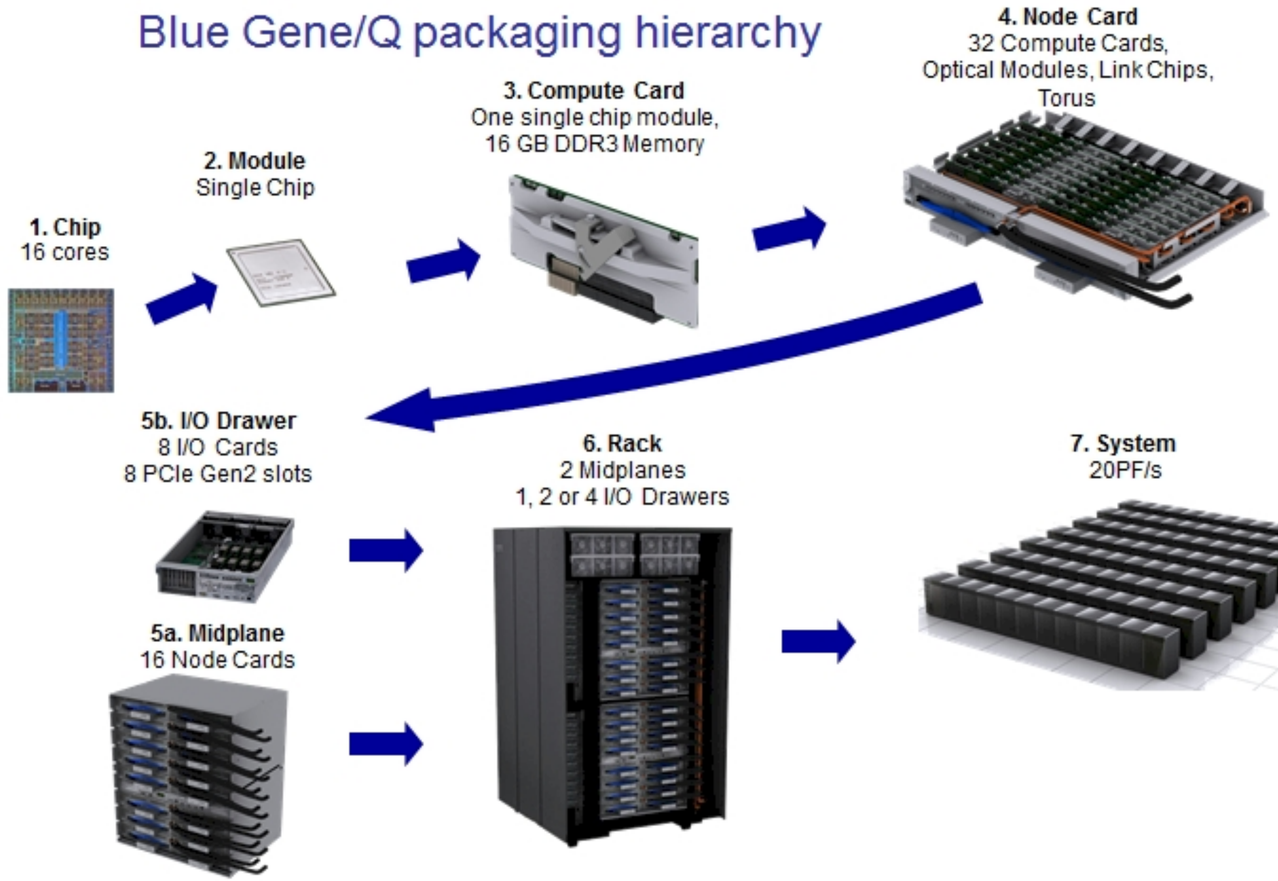
It’s just data, interpretation makes it meaningful.

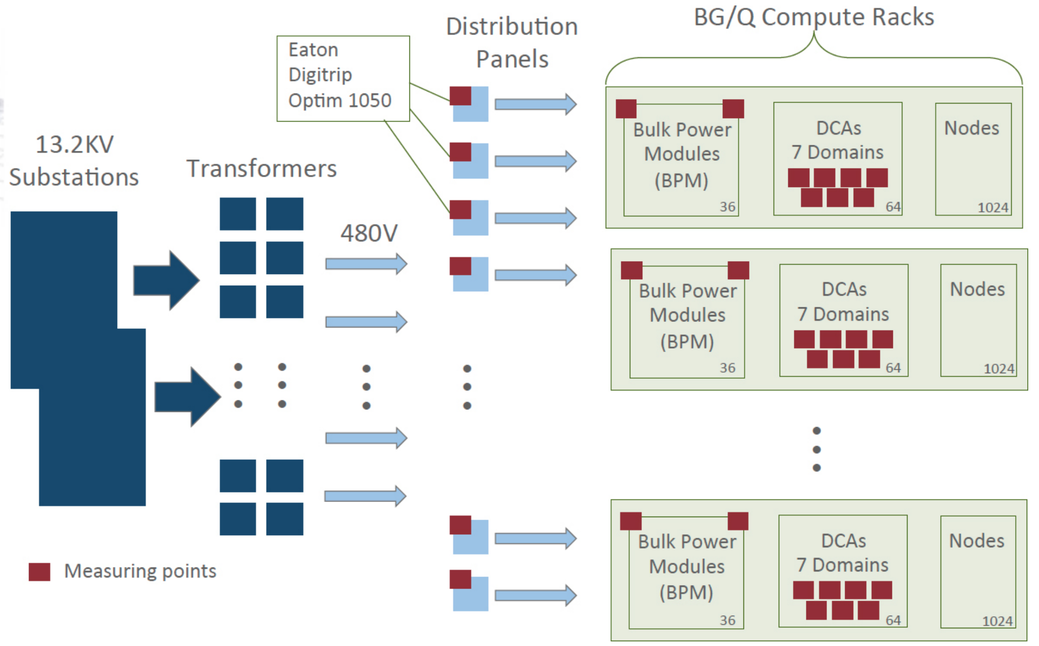
How is it obtained?

What is obtainable?

	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





BG/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).

BG/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”.

Intel RAPL

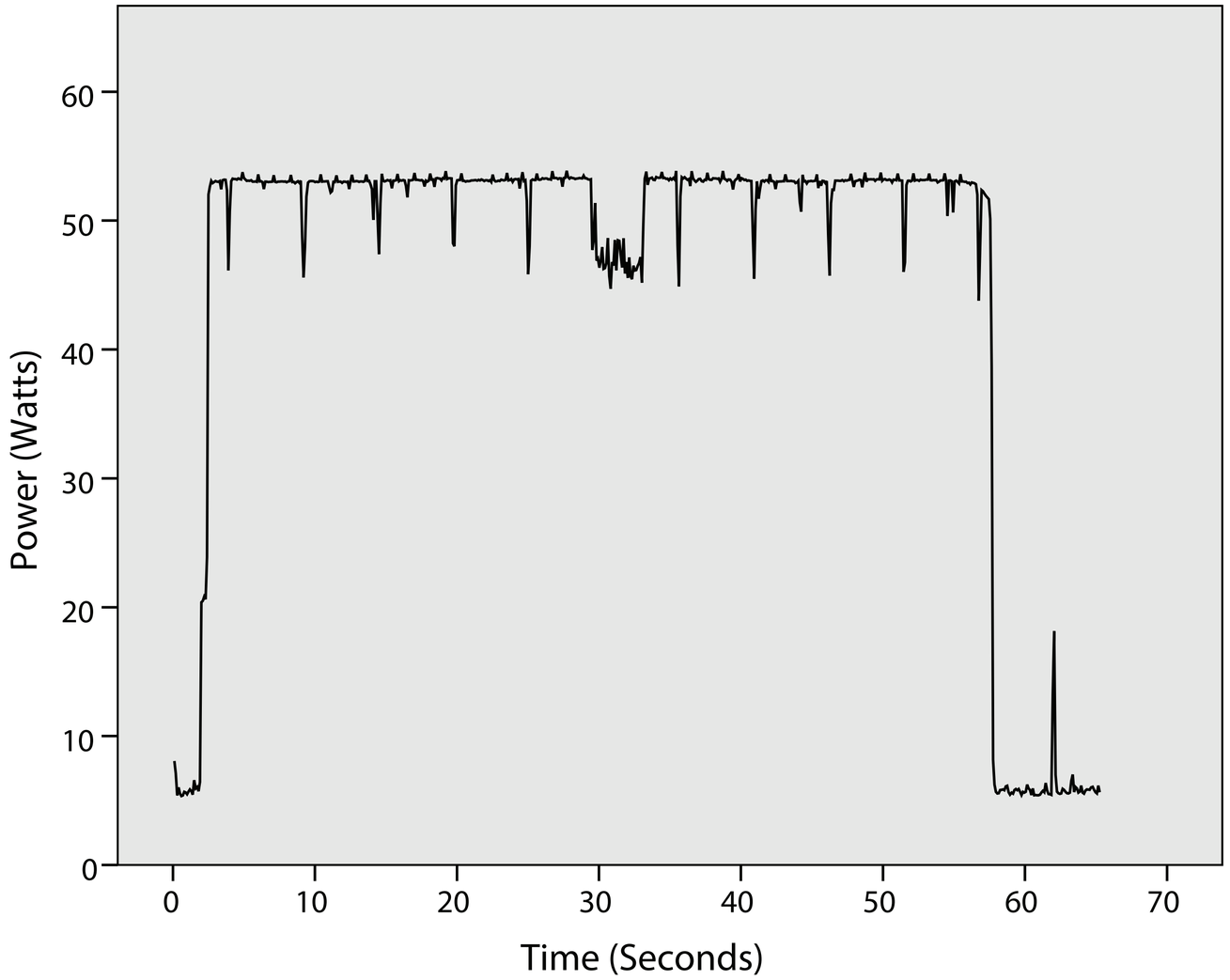
- As of the Sandy Bridge architecture, Intel has provided the “Running Average Power Limit” (RAPL) interface.
- Originally designed to provide a way to keep processors inside of a given power limit over a sliding window of time, but can also be used to calculate power consumption over time.
- Circuitry 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.

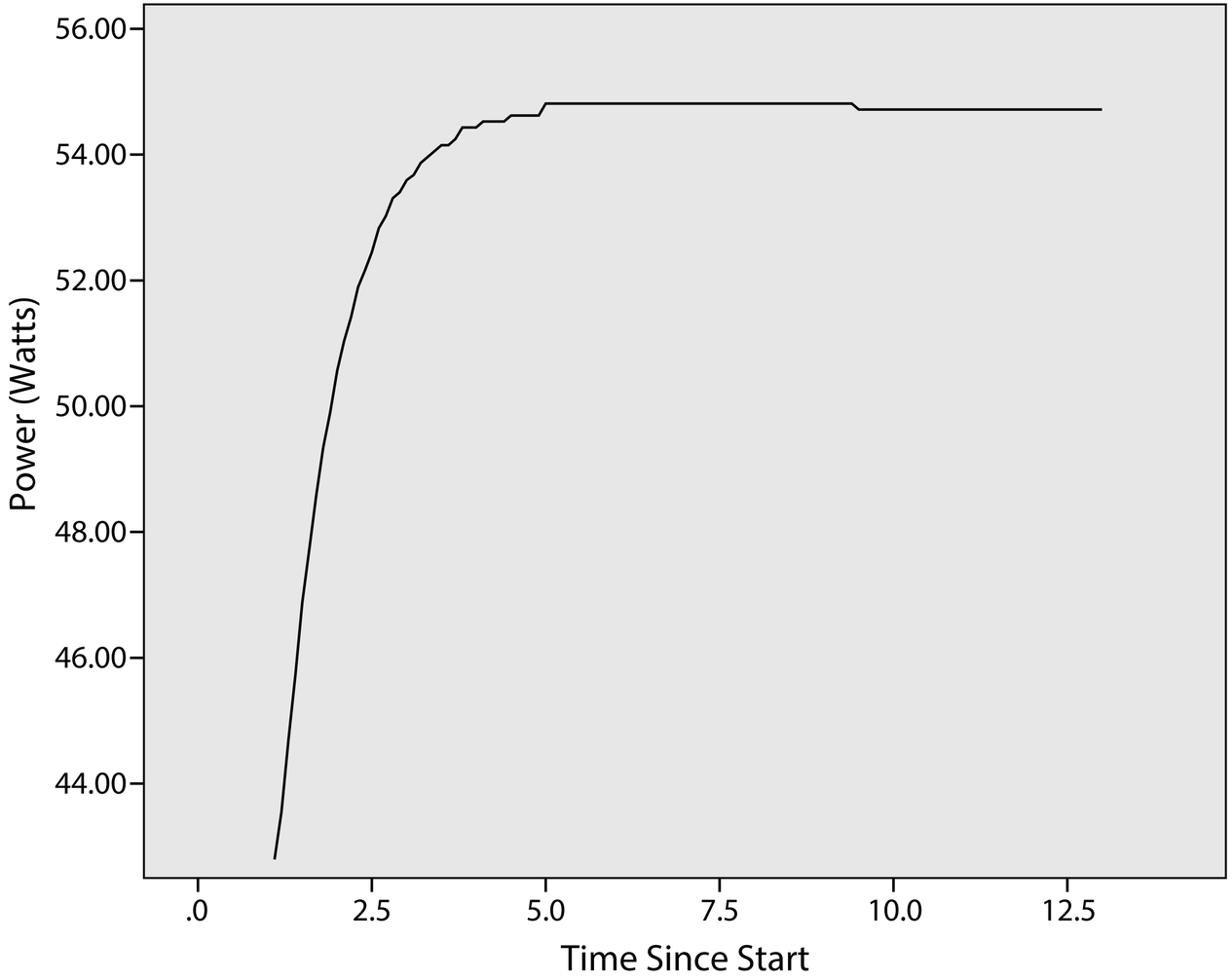
Intel RAPL



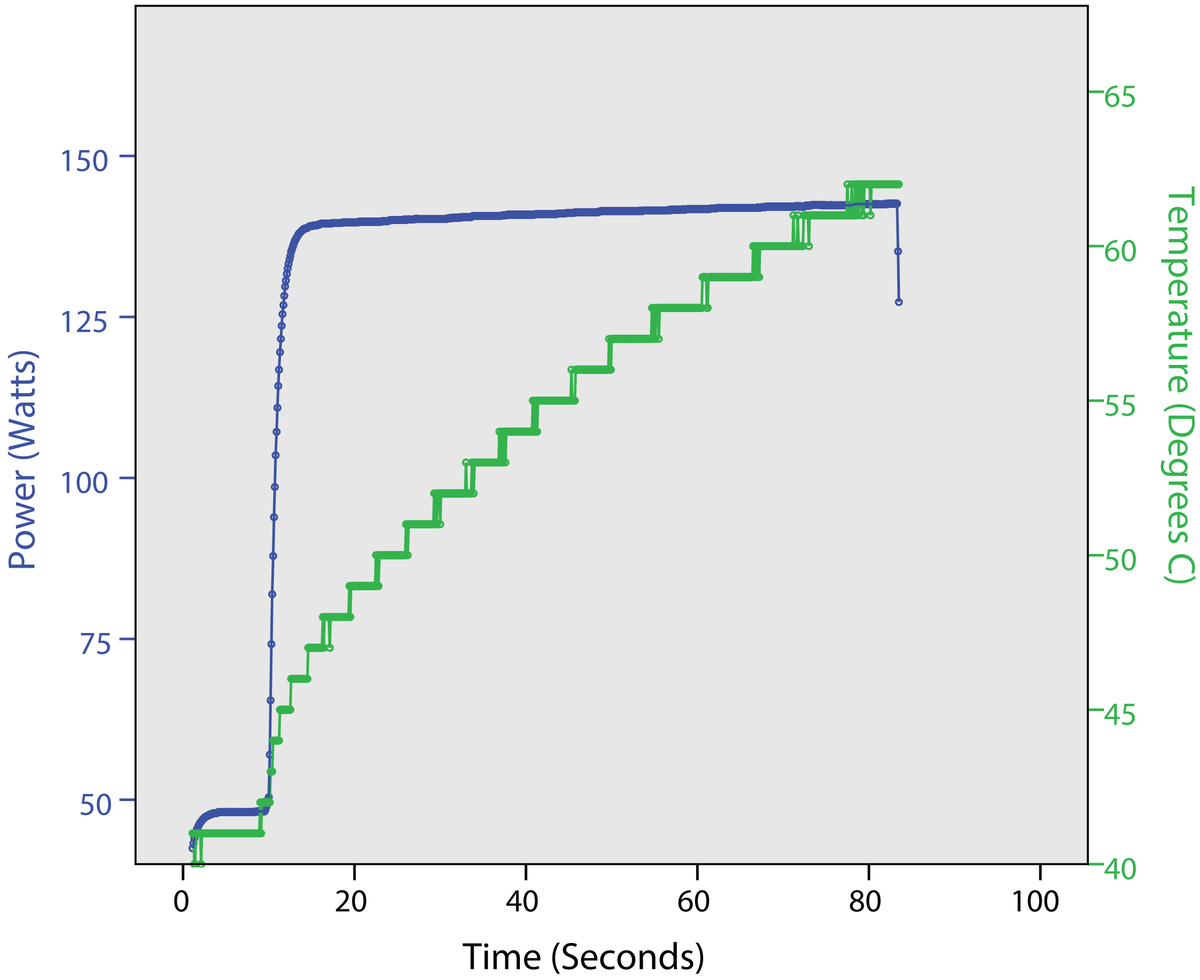
NVIDIA Management Library

- 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, K80, etc.).
- 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

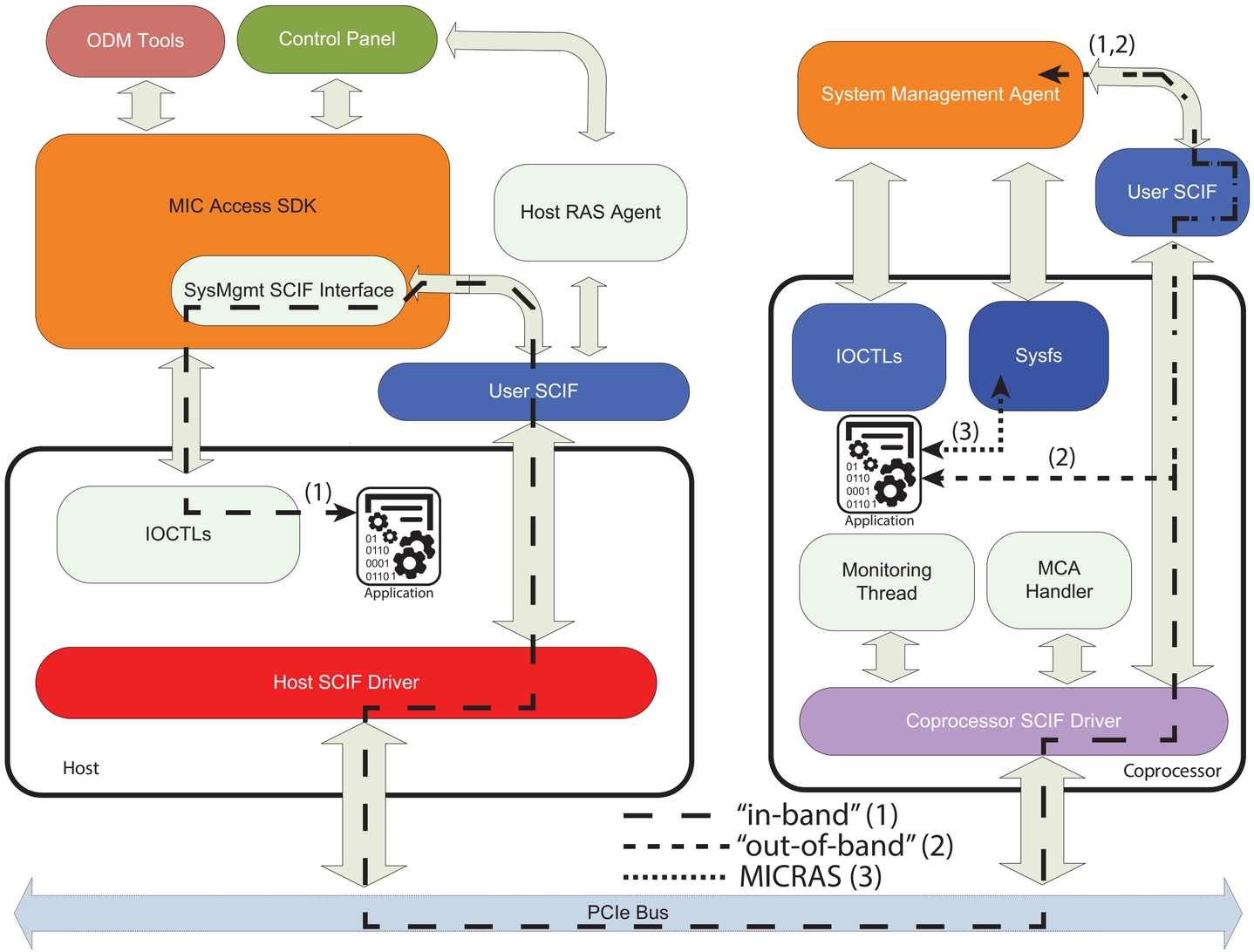


Vector Add Workload

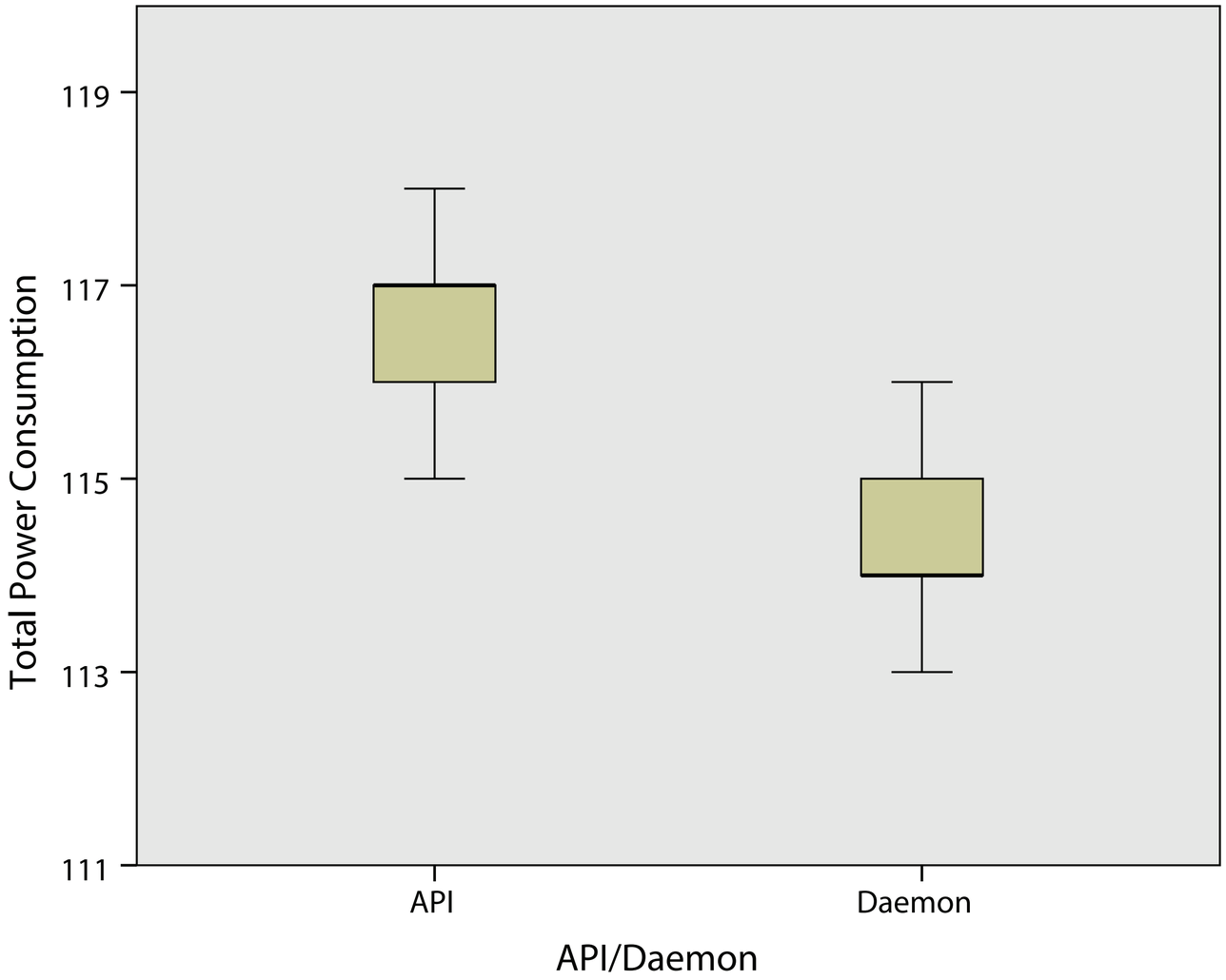
Intel Xeon Phi

- 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



There’s got to be a better way!

The case for a universal API

- Each platform/system has its own method of access, data obtainable, accuracy, latency, etc.
- Developers (unless they care about power data) aren't going to spend the time to implement the calls to the necessary APIs to gather this data.
- What if there was one tool that could capture any/all data with minimal impact to application runtime and with minimal code impact?

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.
- Supports more complex features like tagging specific areas of code.

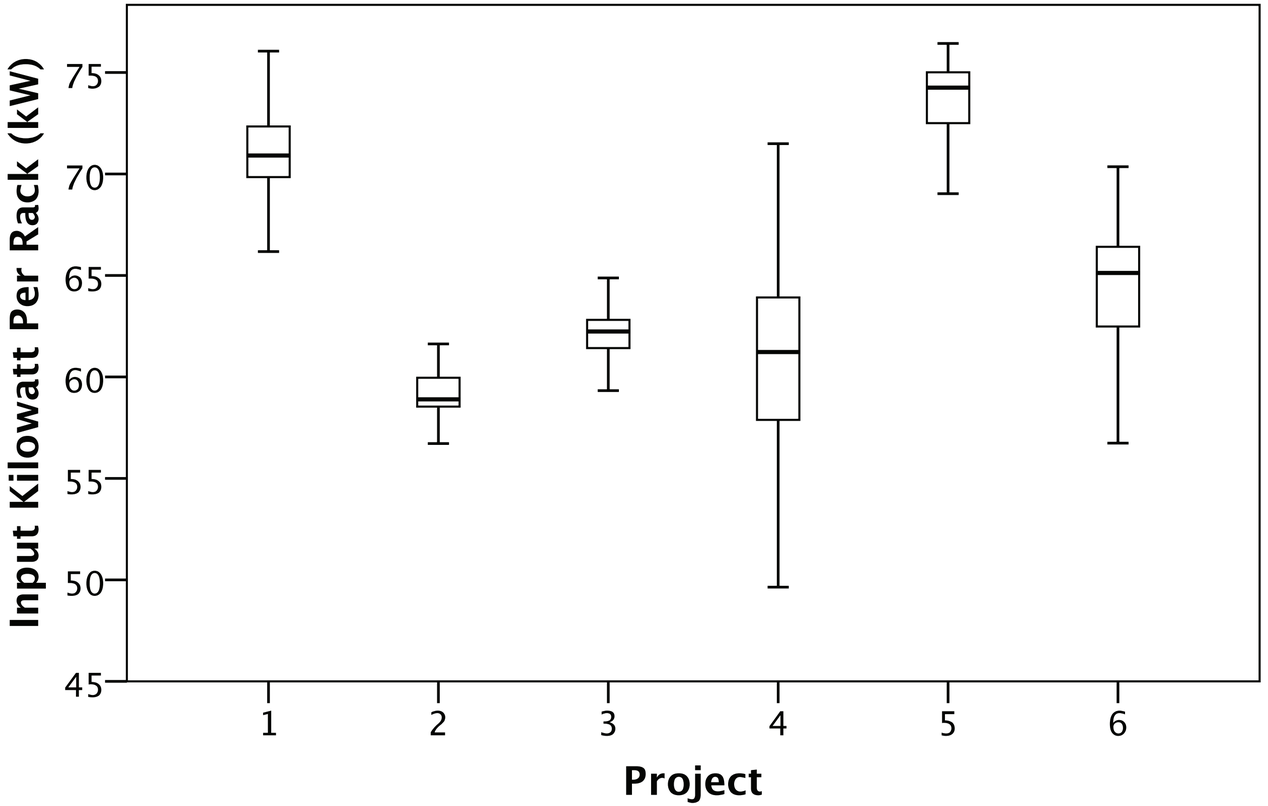
MonEQ

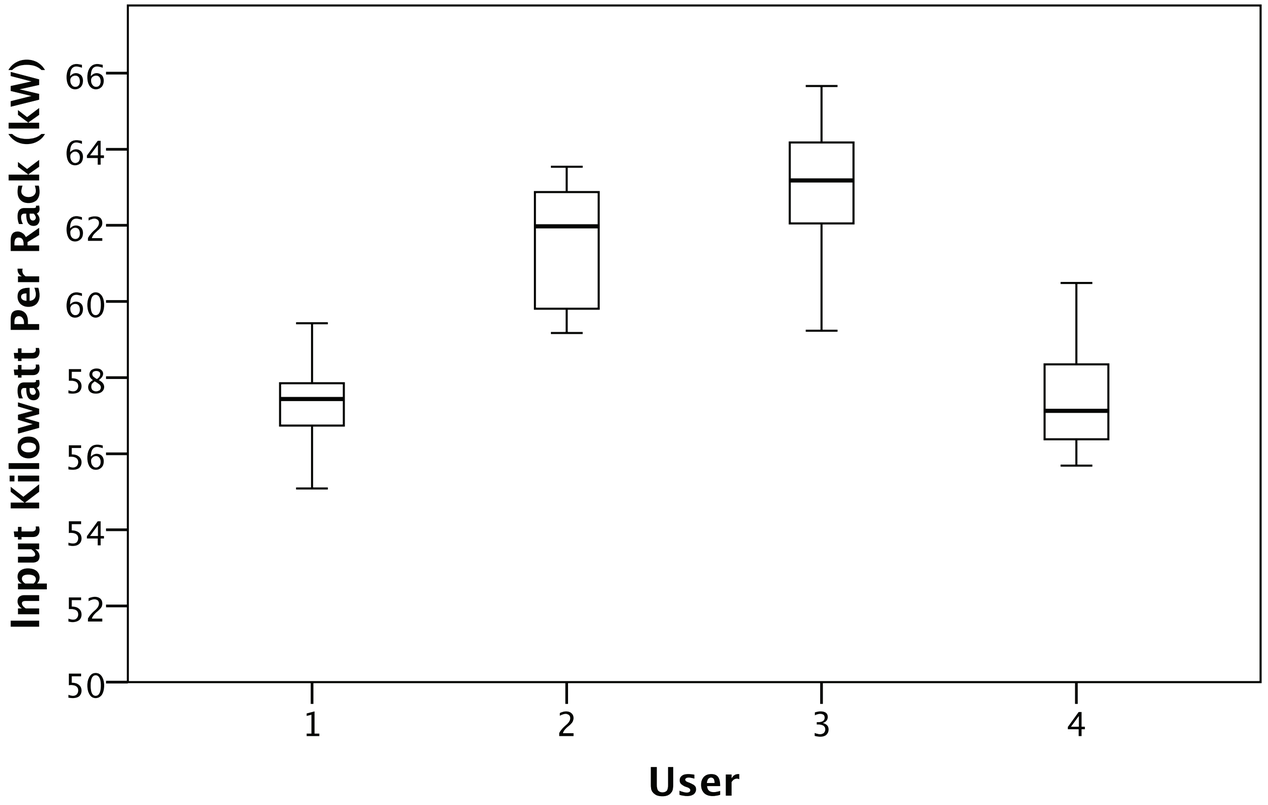
```
int status, myrank, numtasks;

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();
```

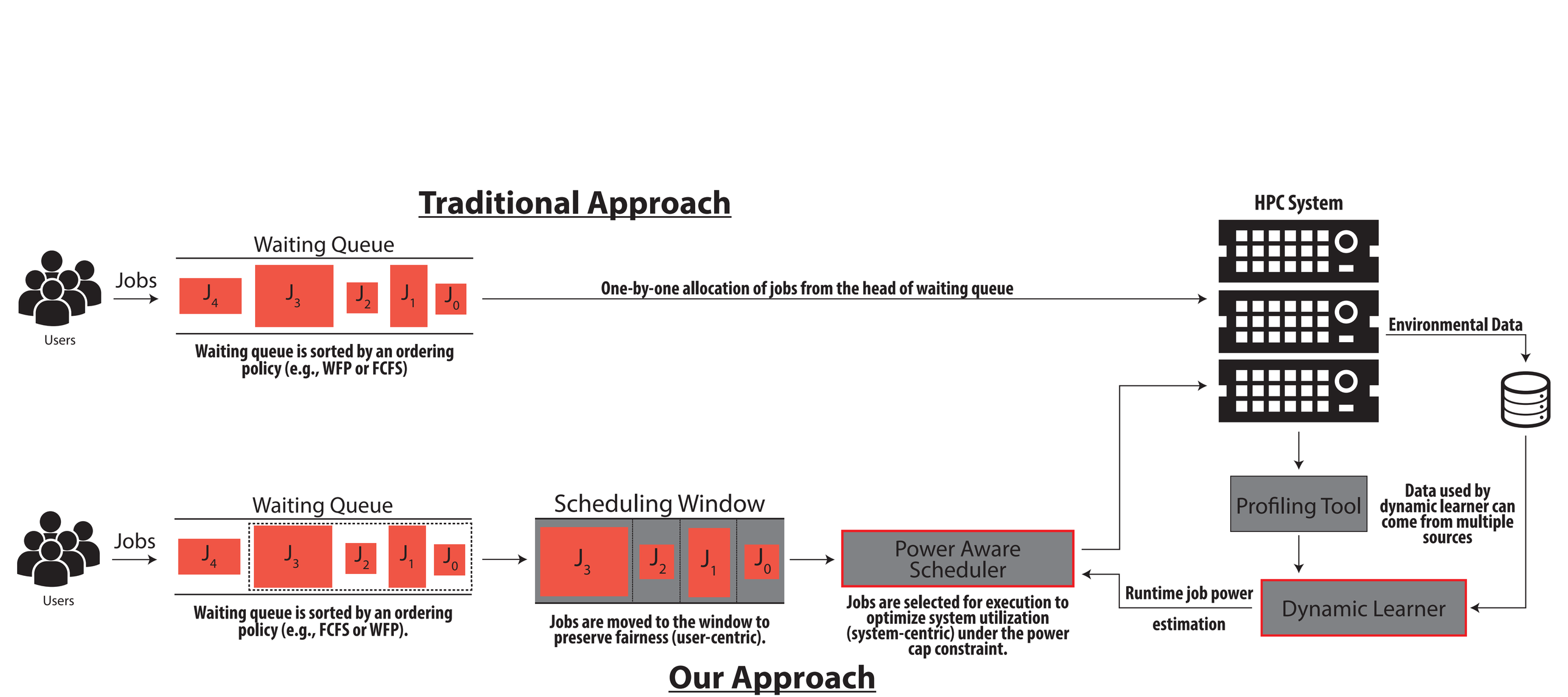
	32 Nodes	512 Nodes	1024 Nodes
Application Runtime	202.78	202.73	202.74
Initialization	0.0027	0.0032	0.0033
Finalize	0.1510	0.1550	0.3347
Collection	0.3871	0.3871	0.3871
Total	0.5409	0.5455	0.7251

What use it it?

Workload Power Analysis






Traditional Approach
HPC System
Waiting Queue
Jobs
Users
J4
J3
J2
J1
J0
One-by-one allocation of jobs from the head of waiting queue
Waiting queue is sorted by an ordering policy (e.g., WFP or FCFS)
Environmental Data
Profiling Tool
Data used by dynamic learner can come from multiple sources
Waiting Queue
Scheduling Window
Power Aware Scheduler
Runtime job power estimation
Dynamic Learner
Waiting queue is sorted by an ordering policy (e.g., FCFS or WFP).
Jobs are moved to the window to preserve fairness (user-centric).
Jobs are selected for execution to optimize system utilization (system-centric) under the power cap constraint.
Our Approach

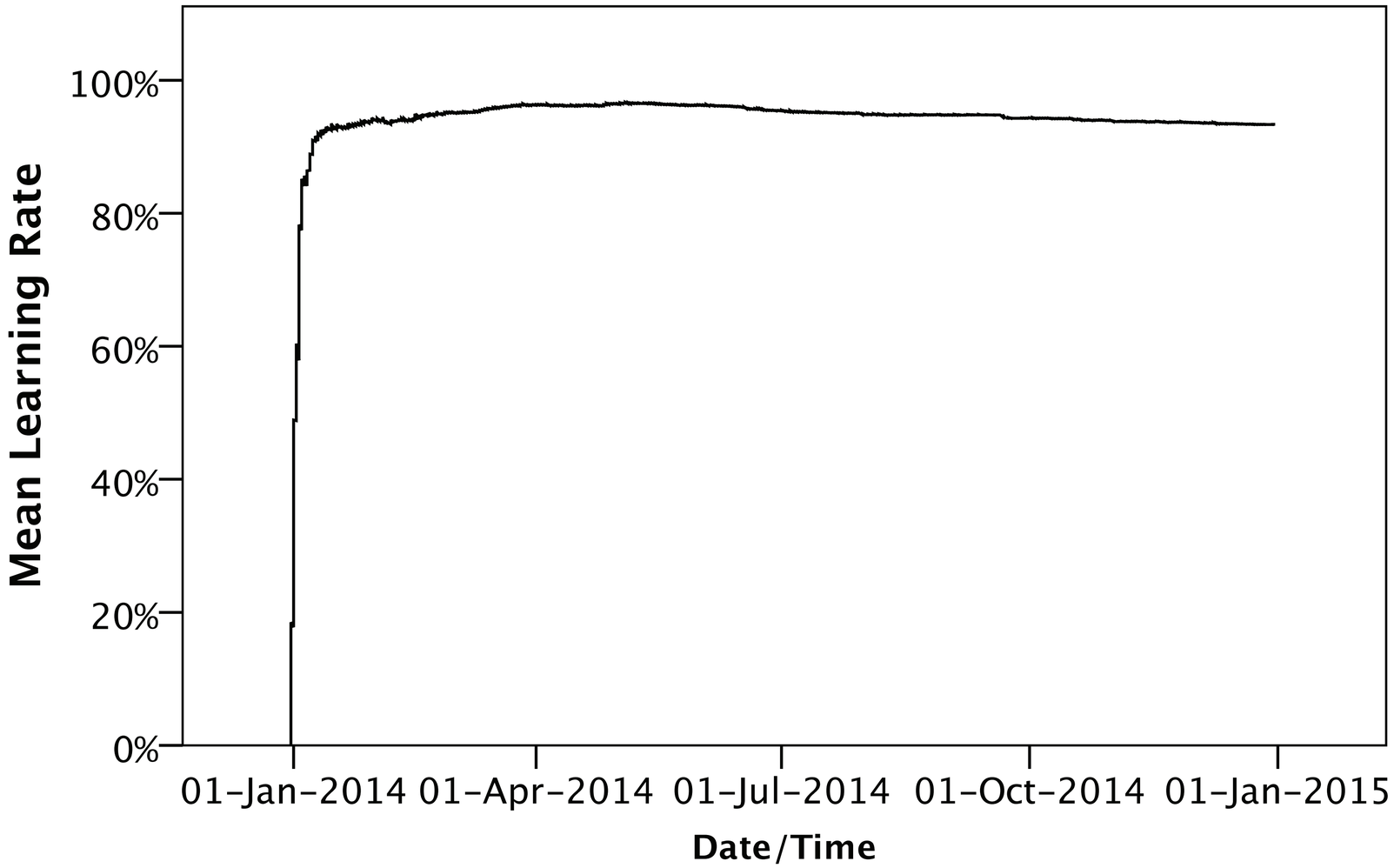
Dynamic Learner

- Takes power data from power monitoring facility (e.g., MonEQ) to estimate job power profiles.
- At each scheduling instance, learner has two tasks:
  - Estimate power profiles of the jobs in the queue.
  - Calculate the available power budget for incoming jobs by estimating power requirements of running jobs.

Power Aware Scheduler

- Selects jobs in the waiting queue for execution to meet scheduling goal under power constraint.
- Uses proposed window-based optimization method.
  - In contrast to conventional approach, our design examines a window of jobs in the queue which helps maintain fairness.
- 0-1 knapsack problem is formulated to describe power monitoring problem.

Learning Accuracy



94% accuracy after just 26 days of execution.

Conclusions

- Power is a funny thing:
  - It means different things to different people, it's not always directly comparable, getting it might mean jumping through hoops, etc., etc.
- But, when used carefully enough, it can be more than just insightful; it can be actionable!
- In the very near future power won't just be an interesting research subject, it will define the very limits of HPC.

Wish List

- Better documentation!
- Continued development of first-class tools.
- Feedback from end-users…what do you want?