Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Enabling Accurate Power Profiling of HPC Applications on Exascale Systems

GOKCEN KESTOR, ROBERTO GIOIOSA, DARREN KERBYSON, ADOLFY HOISIE

Pacific Northwest National Laboratory

Richland, WA

Introduction

- Power is one of the main limiting factors on the road to Exascale
  - Other challenges may be reduced to power limitation (e.g., resilience)
- To increase power efficiency with limited power budget:
  - Shift power to threads on the application's critical path
  - Quickly move idle cores to low power mode
- New aggressive power-aware algorithms need to be developed:
  - precisely measure power consumed by each system component at a given time
  - Power information available to runtimes and applications during execution

Motivation

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

- Despites its importance, power is not yet considered a “first-class” resource
  - Difficult to measure power and to account power consumption to a given process/thread
  - Limits the development of power-aware software algorithms

- Current power measurement infrastructures:
  - Typically out-of-band
    - Coarse-grained sampling frequency (2-3s sampling interval)
    - Spatially coarse grained (measure the entire node)
    - Good for debugging/offline analysis
  - In-band
    - Timing fine-grained (register updated every 1-10ms)
    - Spatially finer grained but still at the level of an entire chip (e.g, Intel SandyBridge)

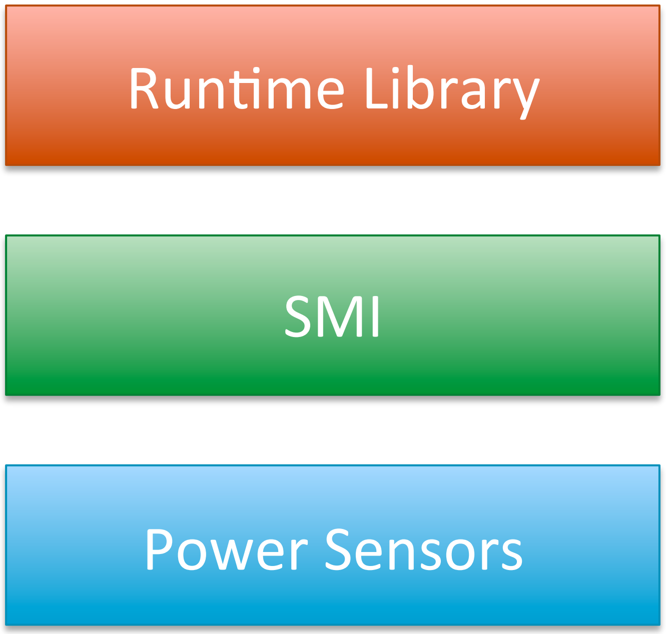
Outline

Proposal

- We want to start developing power-aware algorithms for exascale systems
  - Decoupled algorithms from power measurement infrastructures
- System Monitor Interface (SMI):
  - Portable interface between user and kernel mode
  - Hides low-level architecture details
- Proxy per-core power sensors:
  - Based on per-core activity
    - uses performance counters
  - Generated with regression model
- The proxy power sensor can be replaced with real power sensors without modifications to upper level software

System Monitor Interface (SMI)

- Generic interface between OS and user runtimes, tools, and apps
- Enables the development of power-aware software algorithms/ runtime
  - Abstract the low-level details of the architectural power sensors
  - Uses common **`sysfs`** interface
- Implemented as a Linux kernel module
  - Need to port only *once* this module to use future per-core power sensors rather than *all* runtime systems



System Monitor Interface (SMI) (2)

- Access per-core power information from:

```
/sys/smi/cpu/cpuX/power/core
```

  - Represents the per-core active power of **`cpuX`**
  - Can be accessed by a program or system administration tools (e.g., **`cat`**)

- Access to uncore power information from:

```
/sys/smi/system/power/uncore
```

- We also develop a dynamic profiling library that periodically requests per-core power information while an application is running.

Outline

Proxy per-core power sensor

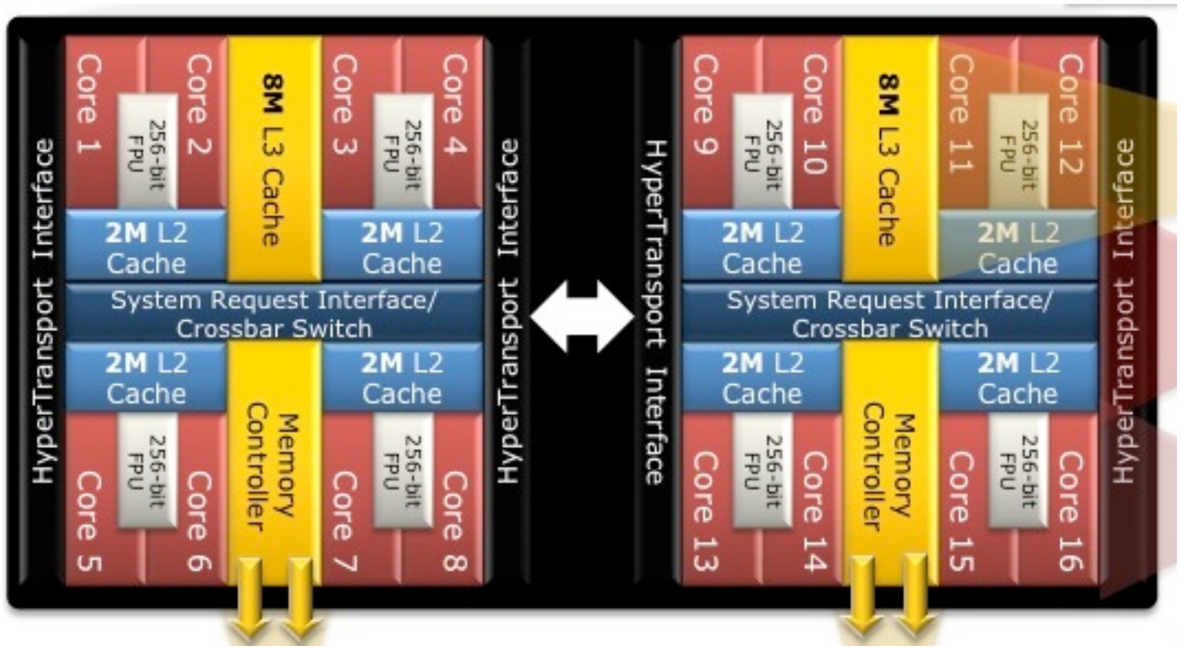
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

- Per-core power sensors are not commonly available
- We develop a per-core power model based on Ordinary Least Square (OLS) regression analysis
- Per-core power consumption is derived from core activity
  - Use performance counters as **`predictors`**
  - Use power value measured by a power meter as **`output variable`**
- We develop a set of micro-benchmarks (training set) that stress individual functional unites
  - Integer
  - Floating point
  - L1/L2/L3/Memory

Hierarchical distribution of resources

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*



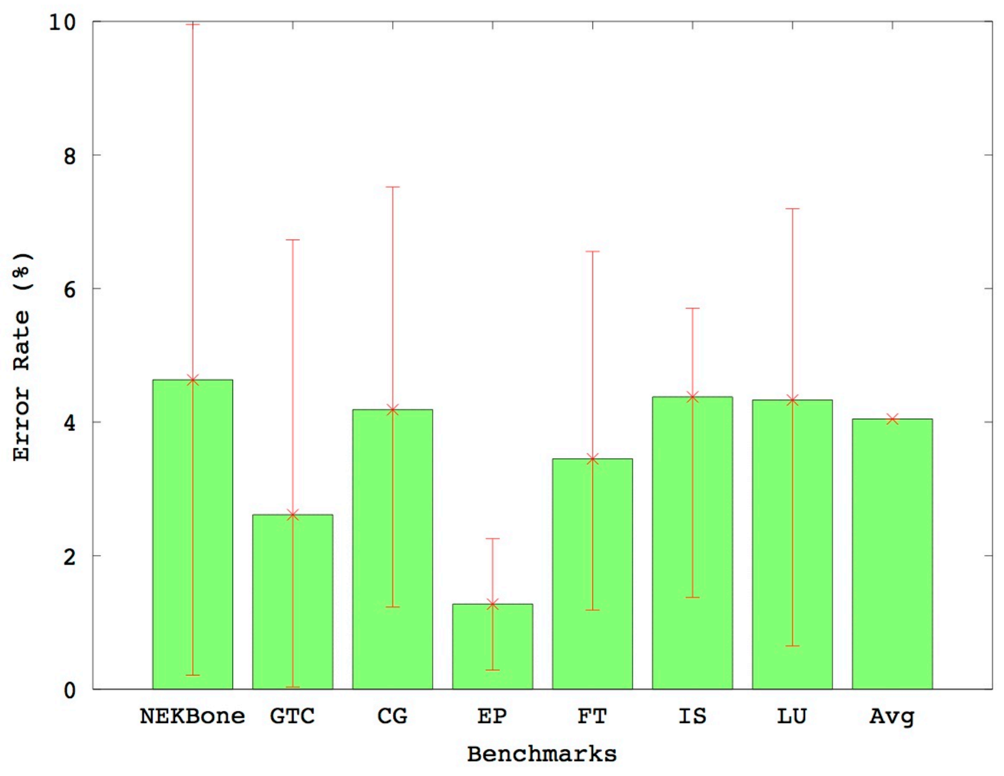
- Current processor architectures are hierarchical (e.g., AMD Interlagos)
  - Hardware resources are clustered
- When training the regression model, we take into consideration how resources are clustered together and shared among cores.
  - Micro-benchmarks are opportunely combined to account for shared resources

Performance counters selection

- Using a large set of performance counters increases the accuracy of the regression model
- Only a limited number of performance counter registers can be sampled at any given time
  - Above this number, the performance counter registers are multiplexed and the counter values extrapolated
  - Multiplexing performance counter registers reduces the accuracy of the regression model
- Remove performance counters that show high correlation with others
- We generate our final regression model based on four predictors
  - Retired instructions, stalled cycles, last level cache misses, FP operations

$$P = P_{uncore} + \sum_{i=1}^{N} P_i \qquad P_i = \sum_{j=1}^{4} \alpha_j r_j$$

Regression model validation



- Validate the accuracy of our power model with the NAS applications, Nekbone and GTC
- Error rate is usually below 5% with maximum error below 10%
- Error rate computed by comparing the estimated power to average power meter output

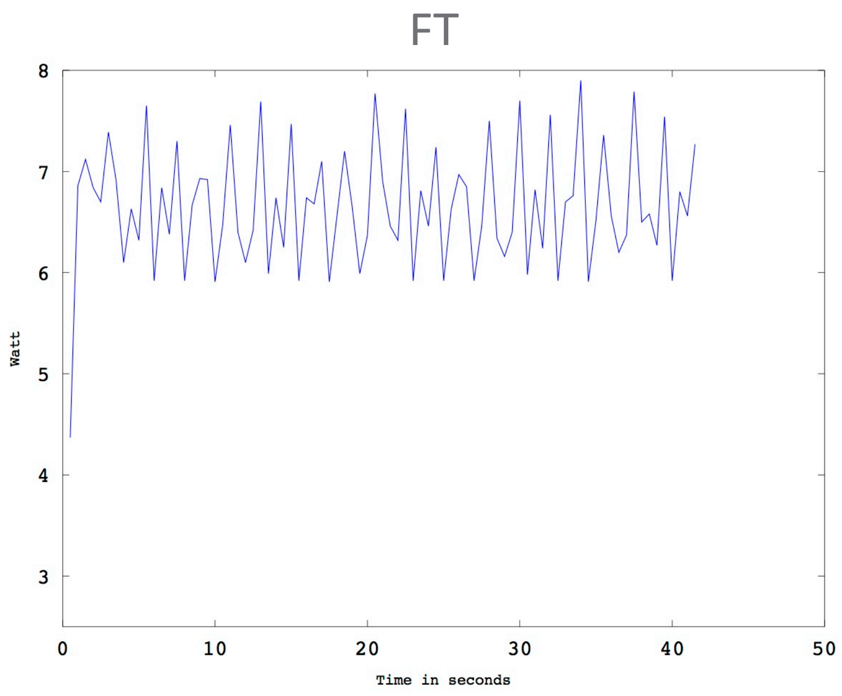
Outline

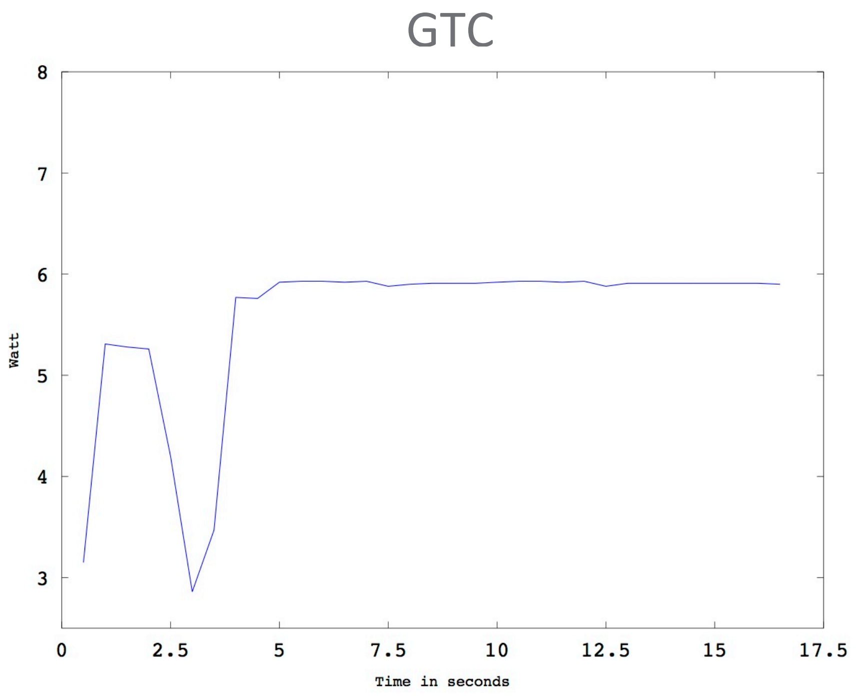
- Introduction
- Proposal
  - System Monitor Interface
  - Proxy per-core power sensor
- Experimental results
- Conclusions

Per-Process Power Profile

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*



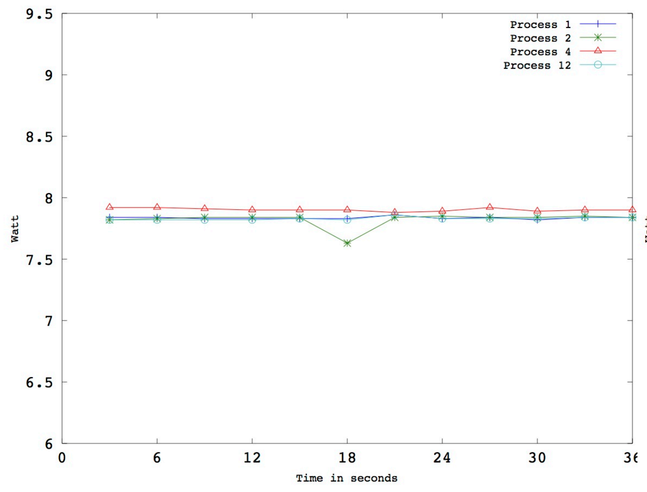


- Sampling rate 2Hz
- **FT**, irregular power profile due to the all-to-all communication pattern
- **GTC**, an initialization phase followed by a regular computing phase

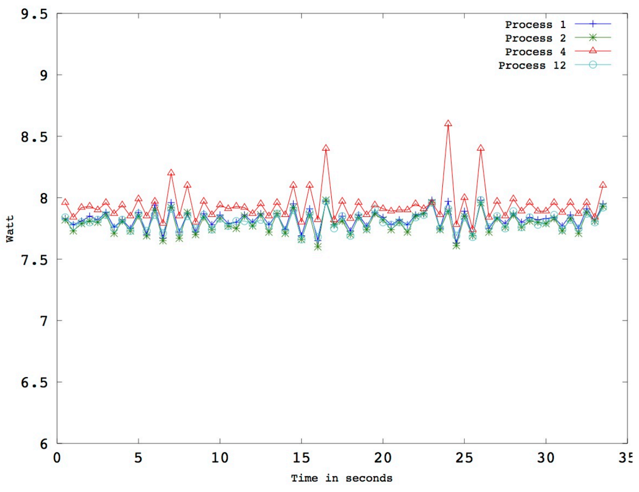
Power profile reflects applications characteristics

Varying sampling frequency

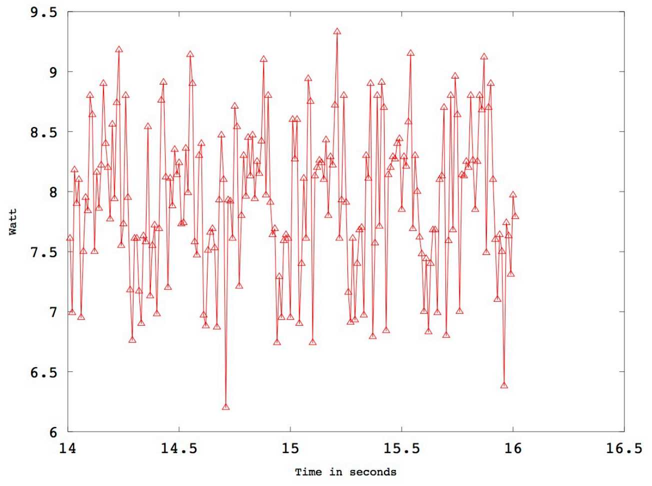
0.3Hz



2Hz



100Hz

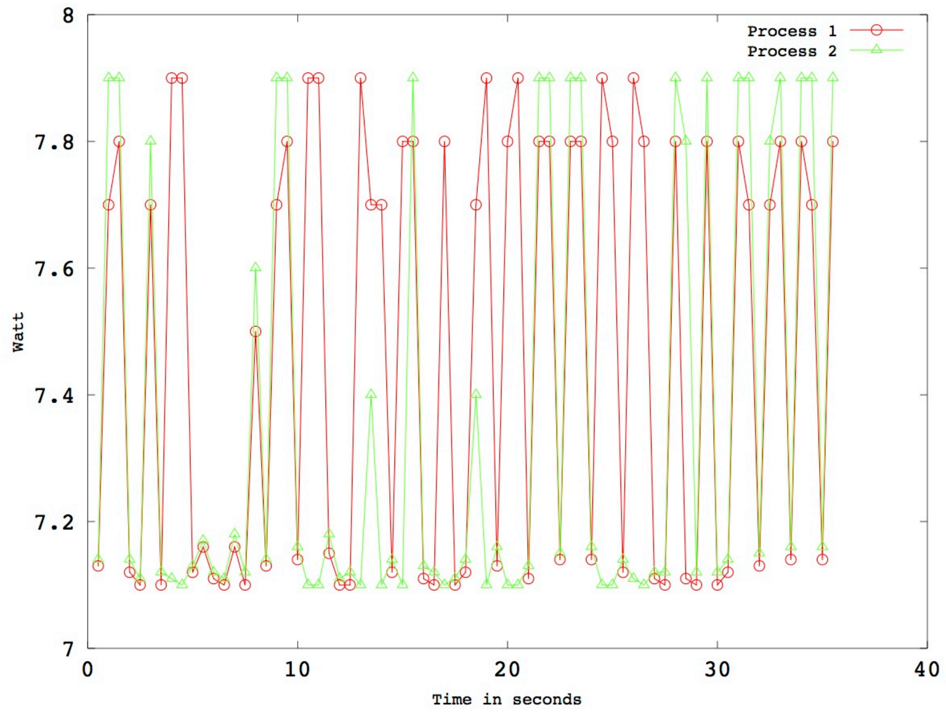


- Process 4 power profile is consistently higher than others
- But… increasing the sampling frequency (2Hz) highlights more details such as large peaks up to 8.6 Watt
- Close-up Process 4's power profile, the power variation is much higher with peaks up to 9.4 Watt

Accounting power consumption to each process is essential

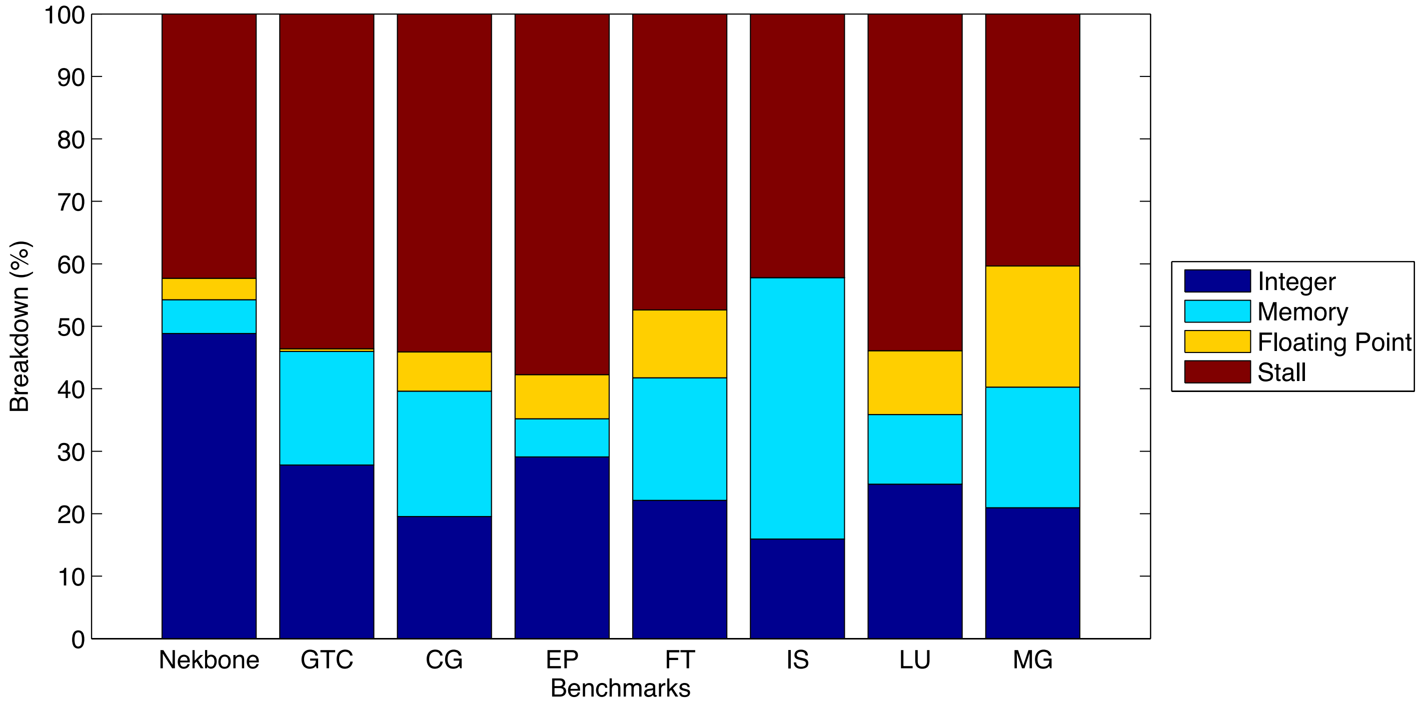
Power saving opportunities

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*



- NEKBone performs several iterations of a conjugate gradient solver
- Higher power consumption during the execution CG and lower power consumption when waiting at the barrier
- Process 1 and Process 2 present different power profile
- Might be effective to shift power between Process 1 and Process 2

Power Breakdown



- Some applications spend a considerable percentage of power in moving data from memory to the processor
- FT, MG and LU, the FP component of power breakdown is also high
- 40-50% of power for all application is wasted due to
  - Data dependencies, functional unit contention, etc.

Outline

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

- Introduction
- Proposal
  - System Monitor Interface
  - Proxy per-core power sensor
- Experimental results
- Conclusions

Conclusions

- Power is one of the major challenges to achieve exaflops performance
- Accurate measurements of power consumption of individual cores is complicated
  - Makes it difficult to develop power-aware algorithms
- We decoupled power-aware algorithms from power sensors
  - System monitor interface
  - Per-core proxy power sensors
- We analyzed scientific applications from NAS benchmarks suite and the exascale co-design centers
  - Power profiles of each application's thread
  - Effects of sampling frequency
  - Power consumption breakdown
- Found power saving opportunities that can be explored in future work

Questions?

gokcen.kestor@pnnl.gov