SUPER
INSTITUTE FOR SUSTAINED PERFORMANCE, ENERGY, AND RESILIENCE

SUPER Power Awareness for HPC

Laura Carrington (Lead), Ananta Tiwari — UCSD/PMaC
Rob Fowler — RENCI
Heike McCraw — UTK
Shirley Moore, Rogelio Long — UTEP

U.S. DEPARTMENT OF ENERGY | Office of Science

Abstract: SUPER's Energy thrust is charged with understanding how computation and communication patterns affect the overall power and energy requirements of HPC applications. We then leverage this understanding to design software- and hardware-aware optimization techniques that enable computations within a power bounds and reduce the DOE's HPC energy footprint. Two focus areas have emerged within this thrust: software solutions that provide fine-grained access to the power measurements and energy efficiency research that utilizes these measurements to develop green optimization strategies. We highlight recent accomplishments in each area and present empirical results that illustrate SUPER's contributions in minimizing DOE's HPC energy requirements.

Power Monitoring with PAPI

POWER MONITORING ON INTEL XEON PHI CO-PROCESSORS

Power and energy measurement activities continue to be of importance for both PAPI and the larger HPC community. The latest PAPI release (5.3) offers two components that allow for monitoring of power usage and energy consumption on the Intel Xeon Phi (a.k.a. MIC) architecture.

Direct Power Reading: "micpower"	**Offloaded Power Reading: "host_micpower"**
The "**micpower**" component runs in native mode – the actual application and PAPI are running natively on the co-processor and its operating system – without being offloaded from a host system. • Provides access to an on-board power sensor on the Xeon Phi which allows measurement of current and voltage (and computed power) for various Phi subsystems at ~50 ms resolution. • Power values are periodically read from the contents of the file /sys/class/micras/power.	The "**host_micpower**" component appears more convenient for users as PAPI is offloaded from the host system, and only the application is running on Xeon Phi. • Easier to measure power consumption of applications running on the Phi at fairly high resolution without actually instrumenting the MIC code directly. • Power data exported through the MicAccessAPI (distributed with Intel MPSS).

The following Table provides a list of events that a user can choose from to obtain power readings for an application running on the Xeon Phil co-processor:

Events	Description
tot0,tot1	Total (average) power consumption over two different time windows (uW)
pcie	Power measured at the PCI-express input (connecting CPU with the Phi) (uW)
inst imax	Instantaneous power consumption reading (uW) Maximum instantaneous power consumption observed (uW)
c2x3,c2x4	Power measured at the input of the two power connectors located on the card (uW)
vccp vddg vddq	Power supply to the cores (core rail) Power supply to everything but the cores and memory (uncore rail) Power supply to memory subsystem (memory rail) (current (uA), Voltage (uV), Power reading (uW))

CASE STUDY

We measure power usage in Figure 1(a) and energy consumption in 1(b) of a Hessenberg reduction computed on the Xeon Phi coprocessor utilizing all 244 cores.

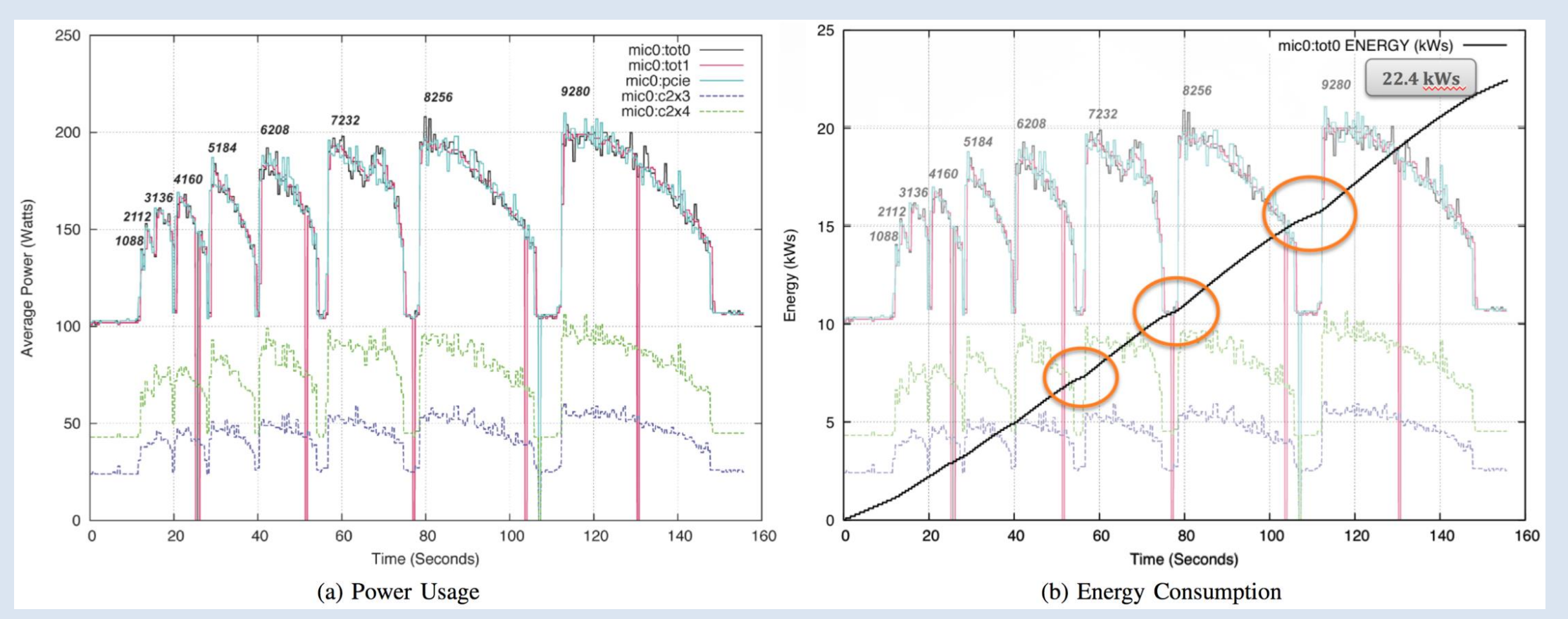


Figure 1: PAPI measurements of 9 Hessenberg reductions (each with different matrix size) performed on Xeon Phi

POWER MONITORING ON IBM BLUE GENE/Q

The PAPI "**bgq-emon**" component exposes power data through the IBM EMON API. Power is supplied to an entire node-board by two identical main power modules, each providing seven high voltage Direct Current (DC) power lines (a.k.a. "domains"). Each power domain from one main power module is paired with the equivalent domain from the other module, and a step-down transformer provides final voltage for the compute cards.

The domains – which are also the power statistics that can be monitored through the PAPI **bgq-emon** component -- are described in the following Table:

Domains	Description
1	Chip core voltage (0.9V)
2	Chip memory interface and DRAM (1.35V)
3	Optics (2.5V)
4	Optics and PCIexpress (3.5V)
6	Chip HSS network transceiver (1.5V)
8	Link chip core (1V)
7	Chip SRAM voltage (0.9+0.15V)

- Current is measured on each of the 14 domains leaving the main power modules.
- Voltage is measured after the final step-down transformation.
  A complete sample is taken in ~500µs (the IBM EMON interface accesses this data from the node-board, which includes a 1-10ms blocking delay for the caller)

CASE STUDIES

On one rank per node-board we register to receive a standard SIGPROF Unix profiling signal, which, when received, triggers a routine to sample PAPI "**bgq-emon**" data. An ITIMER PROF system timer is then requested to trigger at an interval of 300ms, calling the sampling routine.

(a) Matrix-Matrix Multiply: A benchmark, running many small GEMMs on all 512 cores per node-board. The observed power graphs 2(a) demonstrate a rapid climb to full core utilization with a reduction past five minutes as memory is freed and the program winds down.

(b) Parallel 3D-FFT: Figure 2(b) shows the power data for a 3D-FFT computation with 2D data decomposition – running on an entire node-board – for a problem size of 1024^3. The entire kernel is computed five times with a 10 second pause in-between.

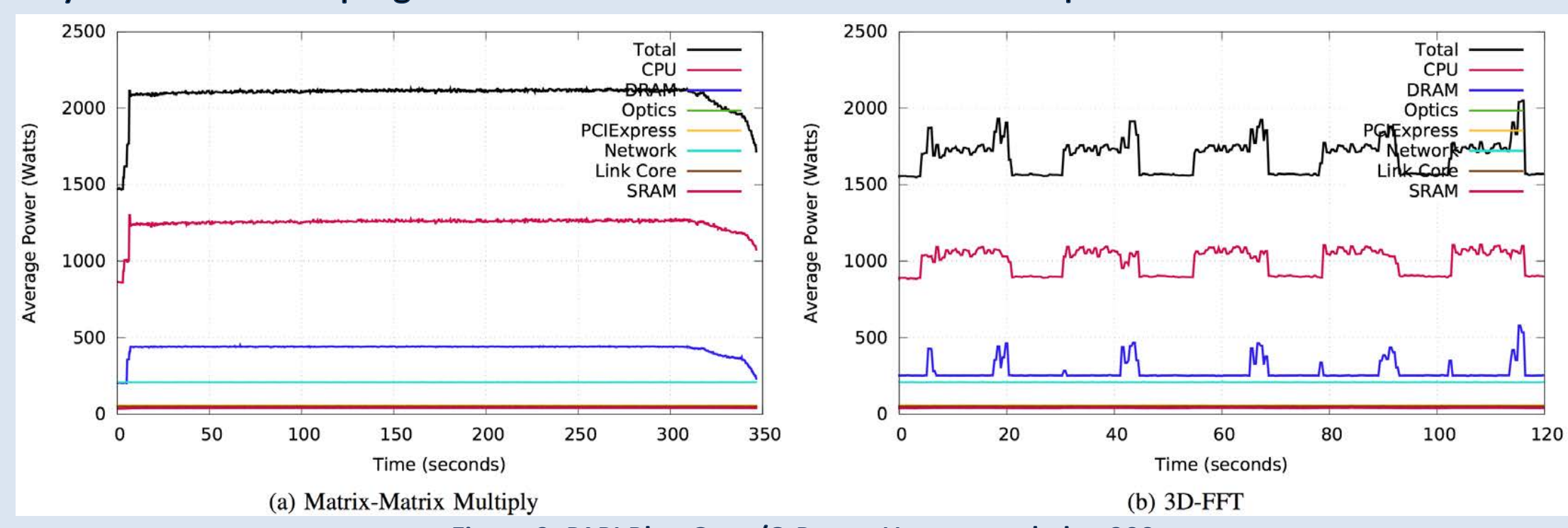


Figure 2: PAPI Blue Gene/Q Power Usage sampled at 300ms

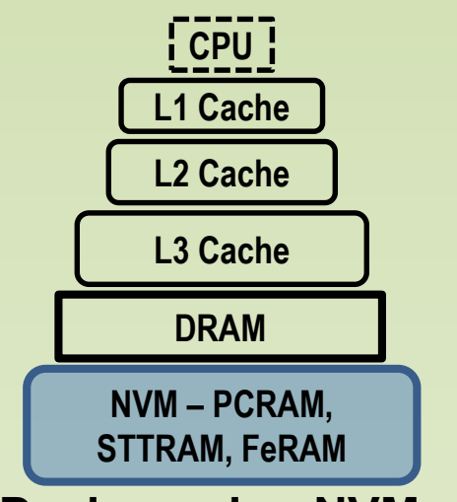
Application aware component level Power & Performance Analysis

For current and future systems power/energy are an important factor. On some current systems we can change the power draw of the different components in order to compute within a power budget or save energy. The next steps in this research is to extend the power/energy investigation from the cores towards the memory sub-system.

Investigating NVM as a power/energy-efficiency option for HPC

DRAM technology has several shortcomings in terms of performance, energy efficiency and scaling. Several emerging memory technologies have the potential to compensate for the limitations of DRAM when replacing or complementing DRAM in the memory sub-system. This experiment looks as NVM as main-memory with DRAM as cache. The design offers:

- Decreases DRAM size and therefore reduces refresh energy.
- DRAM as a cache limits the negative impact on performance and dynamic energy consumption of typical NVM technologies.



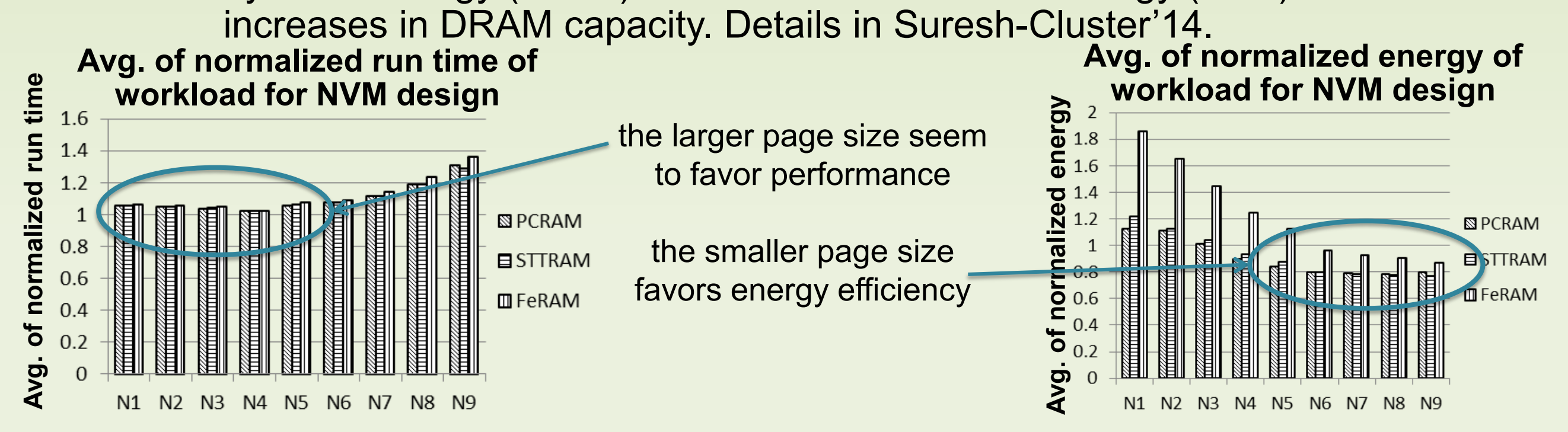
Design using NVM as Main-memory

Experiment: Using performance and energy models we investigate the power/energy and performance affects of a series of configuration designs using NVM technology to augment DRAM for an HPC workload.
(Workload: NPB-BT, NPB-LU, Graph500, Hashing-2, AMG2013, CORAL-CG, Velvet)

Results: Increases in DRAM capacity results in decrease in total access time (~2%) and dynamic energy (~10%) but an increase in static energy (~5%) because of increases in DRAM capacity. Details in Suresh-Cluster'14.

Design Configurations Investigated

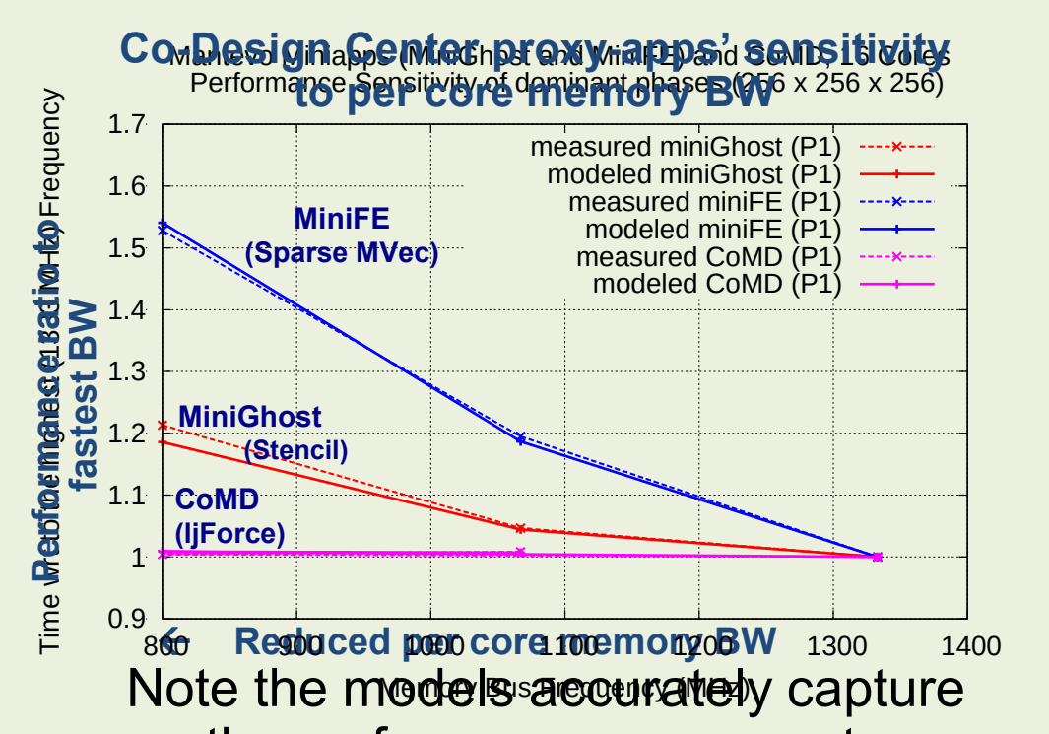
Design Name	DRAM-capacity (MB)	Page-size (KB)
N1	128	4
N2	256	4
N3	512	4
N4	512	2
N5	512	1
N6	512	0.512
N7	512	0.256
N8	512	0.128
N9	512	0.064



Using Memory Models to Explore Algorithmic Choice for Power Provisioning

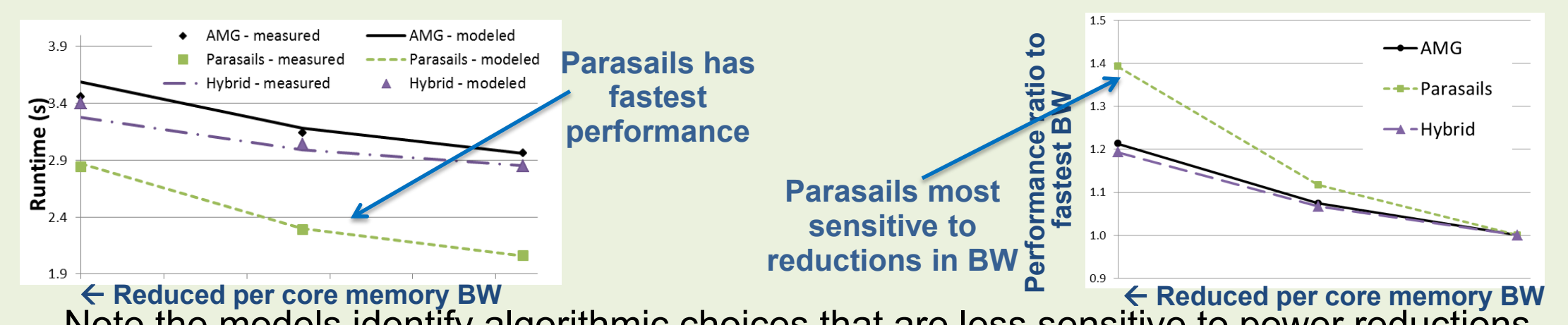
Power can be distributed to core or memory depending on the application's sensitivity to power changes on the given component. In this work we develop model's to capture an application's sensitivity to power changes in the memory by reducing the memory bus speed, which also reduces the per core memory bandwidth.

- Reducing power to memory system by reducing memory bus frequency.
- Memory bus frequency also reduces per core memory bandwidth.
- Growth of multi-core means also means less per core memory bandwidth
- Future systems will most likely continue to have reduced per core memory bandwidth
- Models identify sensitivities of algorithms to reduced per core memory bandwidth (e.g. reduced memory power)
- Models aid in algorithmic choice for future systems



Note the models accurately capture the performance response to reductions in the memory frequency or power. Models are fine-grained and capture the different computational phases within an application and their different responses.

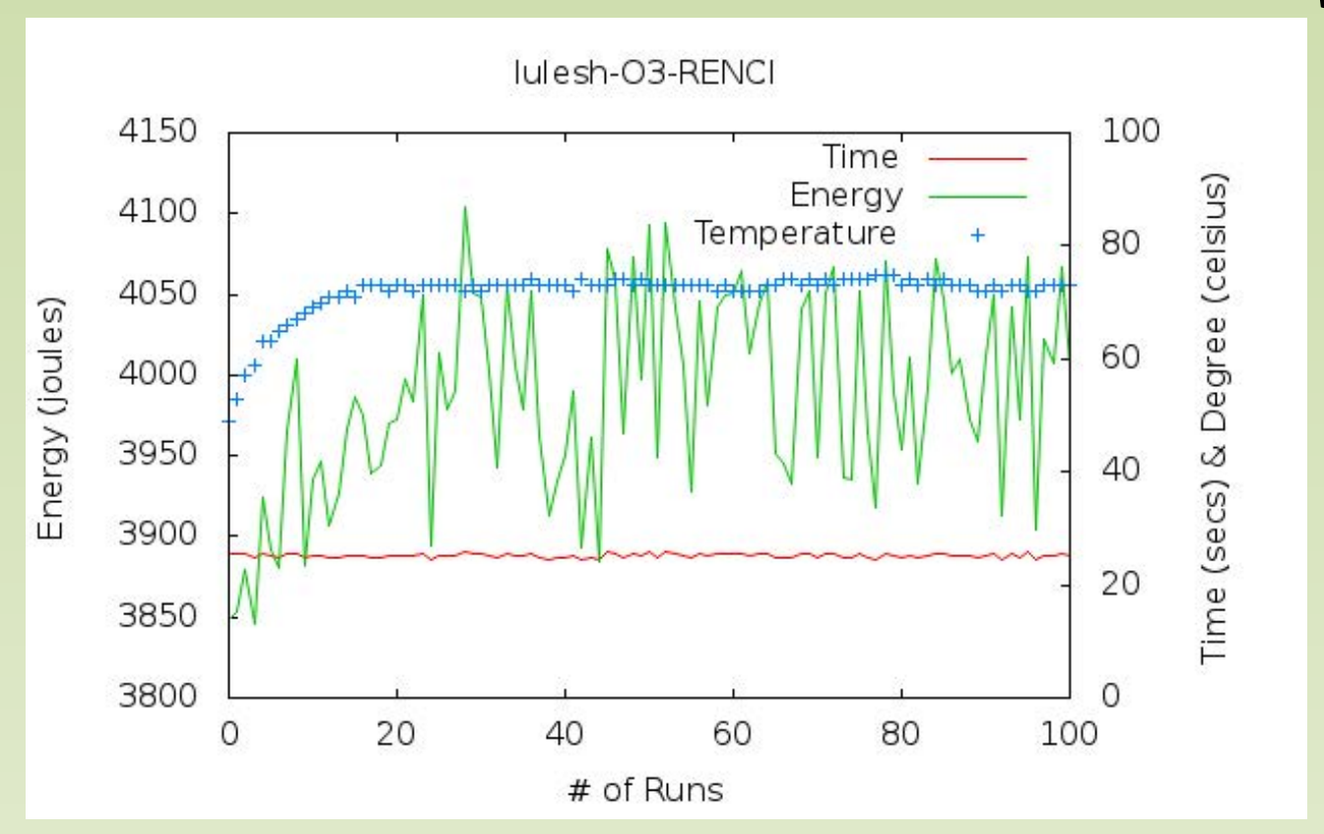
Determining the algorithm sensitivity to power changes using models
Models of 3 different algorithms for AMG application



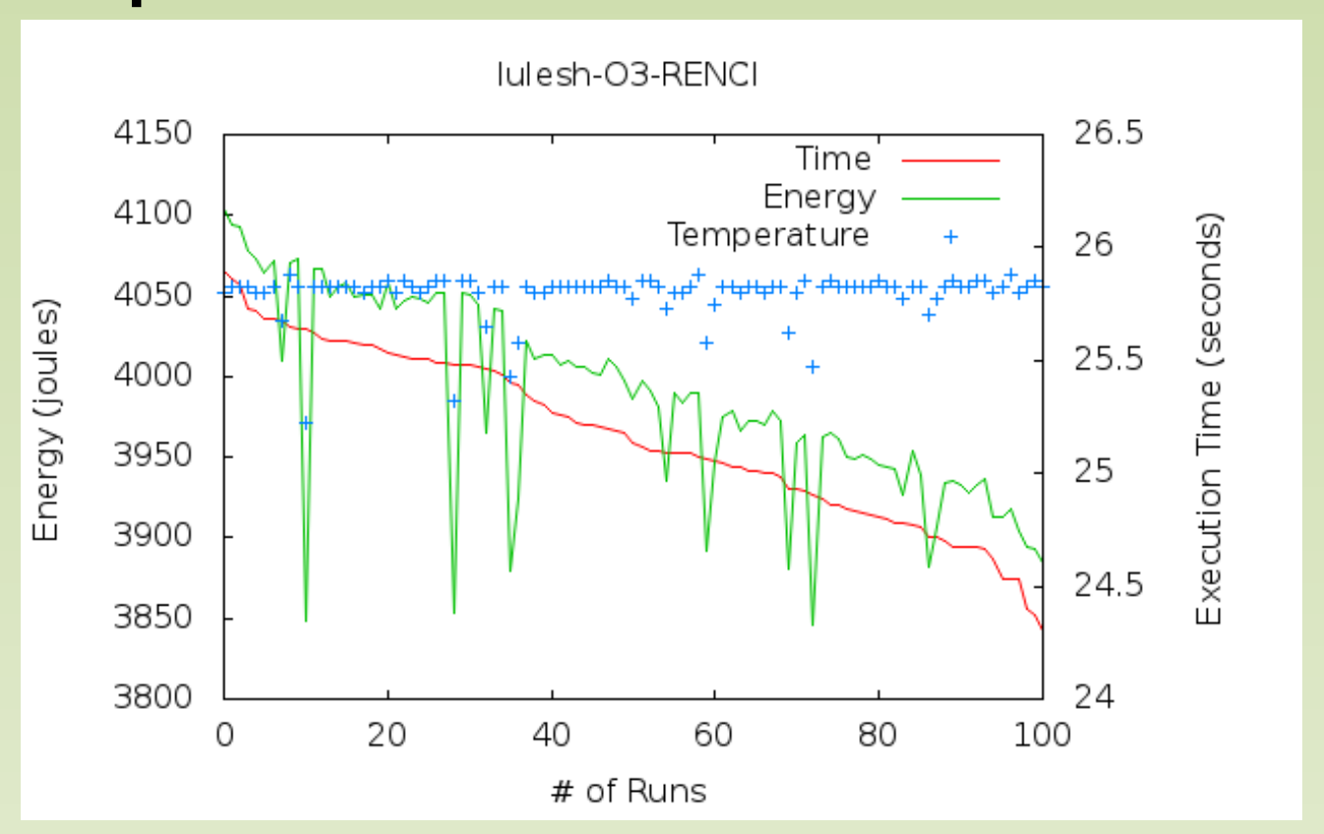
Note the models identify algorithmic choices that are less sensitive to power reductions in the main memory enabling optimal decisions in power budgeting for applications. Details in Tiwari-Cluster'14.

Adapting to Power and Performance Variability

Problem: Computation speed and energy efficiency vary with and are constrained by temperature.



100 Lulesh runs sorted by execution order.



100 Lulesh runs sorted by execution time.

Processors are not identical:

- Two "identical" chips at the same clock speed will have different voltages and temperatures, or need to have different speeds for identical temperature bounds.
- Actual cooling capacity varies between sockets, blade slots, chassis, …
- Performance heterogeneity induces effective load imbalances where none should exist.
  - Imbalances look too "small" to fix by moving work around.
  - Faster processors waste energy by racing ahead of synchronization partners.

Recent processors use adaptive control to adjust performance within a temperature/power envelope.

- Hardware dynamically adjusts clock and voltage.
- Programmer accessible controls:
  - Complete manual override of HW control.
  - Per core clock cycle skipping (a.k.a. clock modulation)
  - Set power capping policy used by HW control.

Approach: Apply Power Capping and Clock Modulation to smooth synchronization arrival times.

- Launch privileged daemon that accesses power MSRs using the batch system (Slurm).
- Monitor MPI_wait () times to identify problems.
- Use either power capping or modulation to adjust power and performance.

Results:

- Wasted energy is effectively scavenged.
- On a single chip, these methods have the potential of shifting the thermal budget to the cores that can benefit.

Iso-Power-Efficiency

- True cost of running an application is the energy used.
- Future machines will have a power bound and users will request a power budget when they submit a job.
- Iso-efficiency behaves differently when scaling the problem size up with increasing power budget, rather than just the number of cores.

$$T_o = \frac{p}{p_1} T_p - T_{p_1}$$ where p_1 is the minimum number of processors that can be used

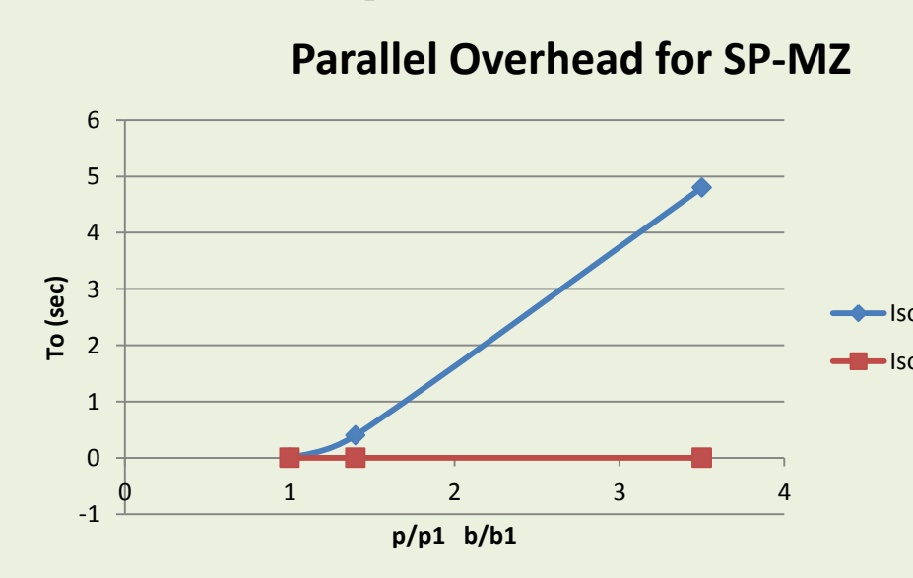
- Iso-power-efficiency overhead function:

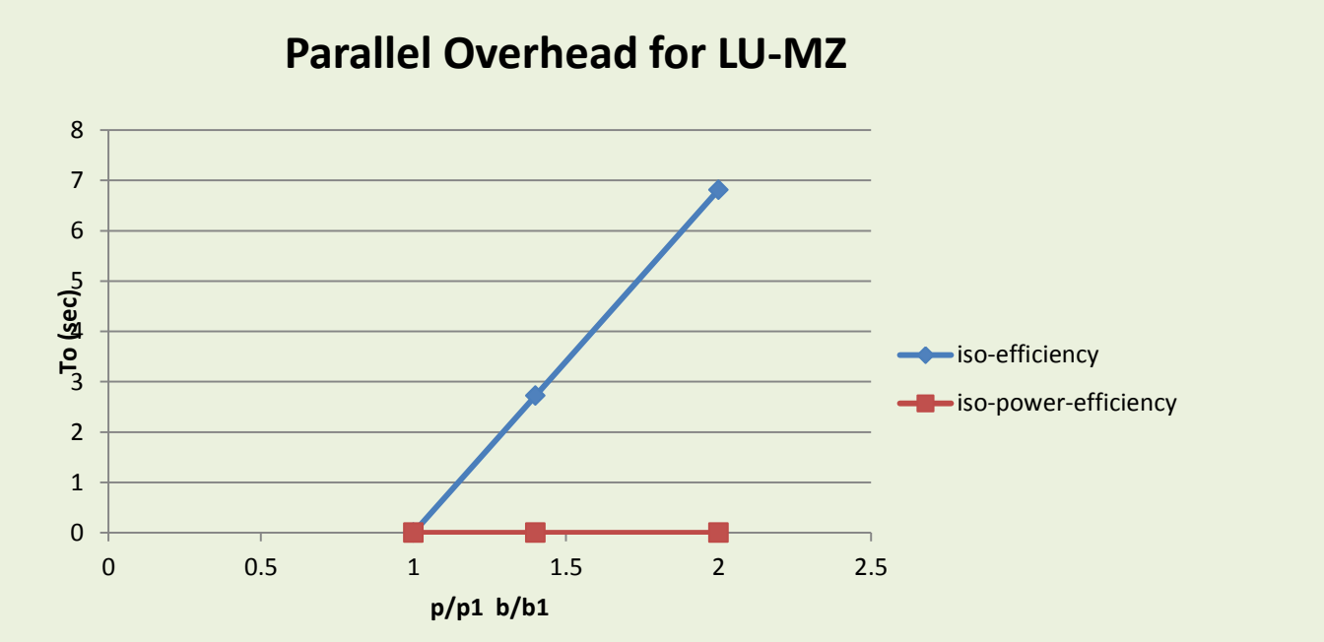
$$T_o = \frac{b}{b_1} T_b - T_{b_1}$$ where b is the power budget and b1 is the smallest power that can be used

- The the Iso-(power)-efficiency function gives the rate at which we must scale up the problem size per core to maintain the same parallel efficiency E:

$$W = \frac{E}{1-E} T_o$$

- Graphs below are using data from Patki et.al. ICS'13.
- $T_0 = 0$ indicates we can achieve perfect linear strong scaling with fixed problem size or maintain constant efficiency with no increase in problem size per core:





Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (and Basic Energy Sciences/Biological and Environmental Research/High Energy Physics/Fusion Energy Sciences/Nuclear Physics).