Runtime Power Modeling to Enable Energy Optimizations in General-Purpose Graphics Processing Units

Vignesh Adhinarayanan
Ph.D. (CS) Student
Synergy Lab, Virginia Tech

MOTIVATION

- Supercomputing constrained by power consumption
  - DOE goal: Reach exascale levels, but do not exceed 20 MW

- Typical power requirement for Los Alamos = 66 MW
- Power budget for Trinity supercomputer alone = 15 MW
- Exceeding power budget ➔ Brownouts in Los Alamos
  - *Installing and starting ASCI White supercomputer in Livermore may have played a small part in the 2001 rolling California brownouts*

MOTIVATION

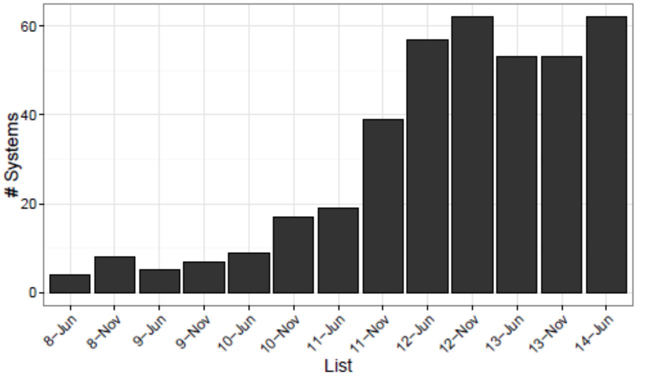
- Supercomputing constrained by power consumption
  - DOE goal: Reach exascale levels, but do not exceed 20 MW
- Power management necessary to reach exascale goal
  - Given an upper bound on power, maximize performance
    - You can't manage what you cannot measure

MOTIVATION

- Supercomputing constrained by power consumption
  - DOE goal: Reach exascale levels, but do not exceed 20 MW

- Power management necessary to reach exascale goal
  - Given an upper bound on power, maximize performance

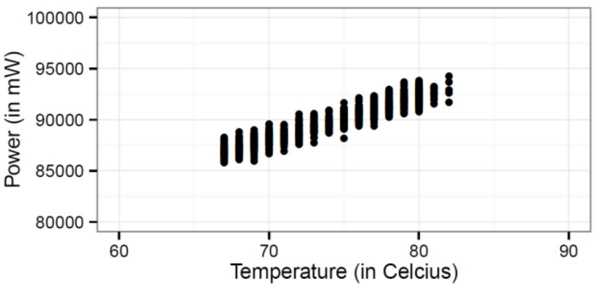
- Important to focus on GPGPUs
  - 60+ systems in Top500 lists
  - 35% performance share



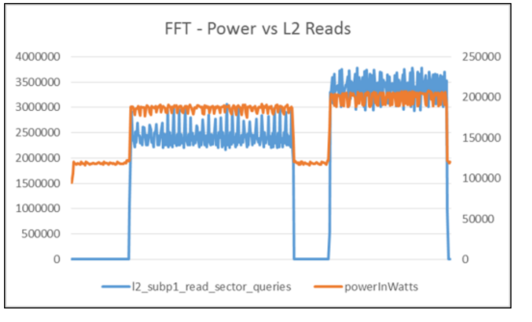
BACKGROUND

- Total power = Static power + Dynamic power

- Static power: Power consumed at idle state
  - ➢ Affected by temperature

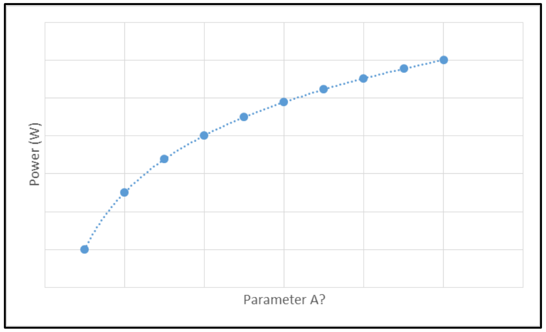


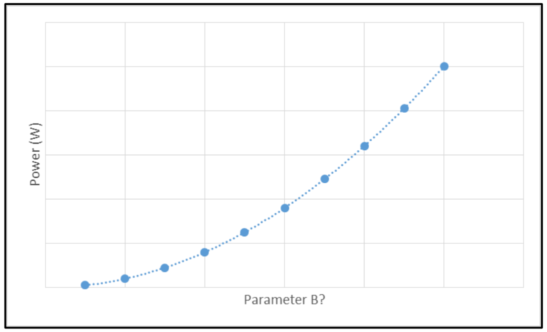
- Dynamic power:
  - ➢ Affected by GPU activity
  - ➢ Certain performance counters track dynamic power

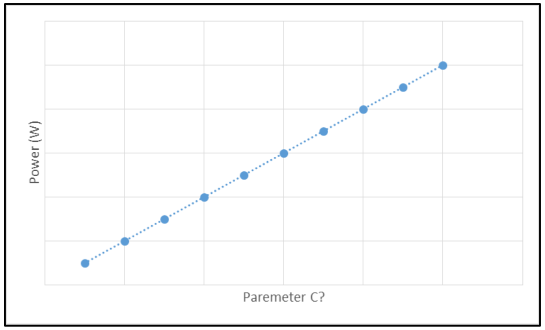


GOALS

- What parameters should we use to model power?
  - Example of input parameters: Instructions/s, Memory transactions, Cache hit rate etc.
- What mathematical functions express relationship between input parameters and power?







Approach

Systematically study various parameters and models with a variety of applications

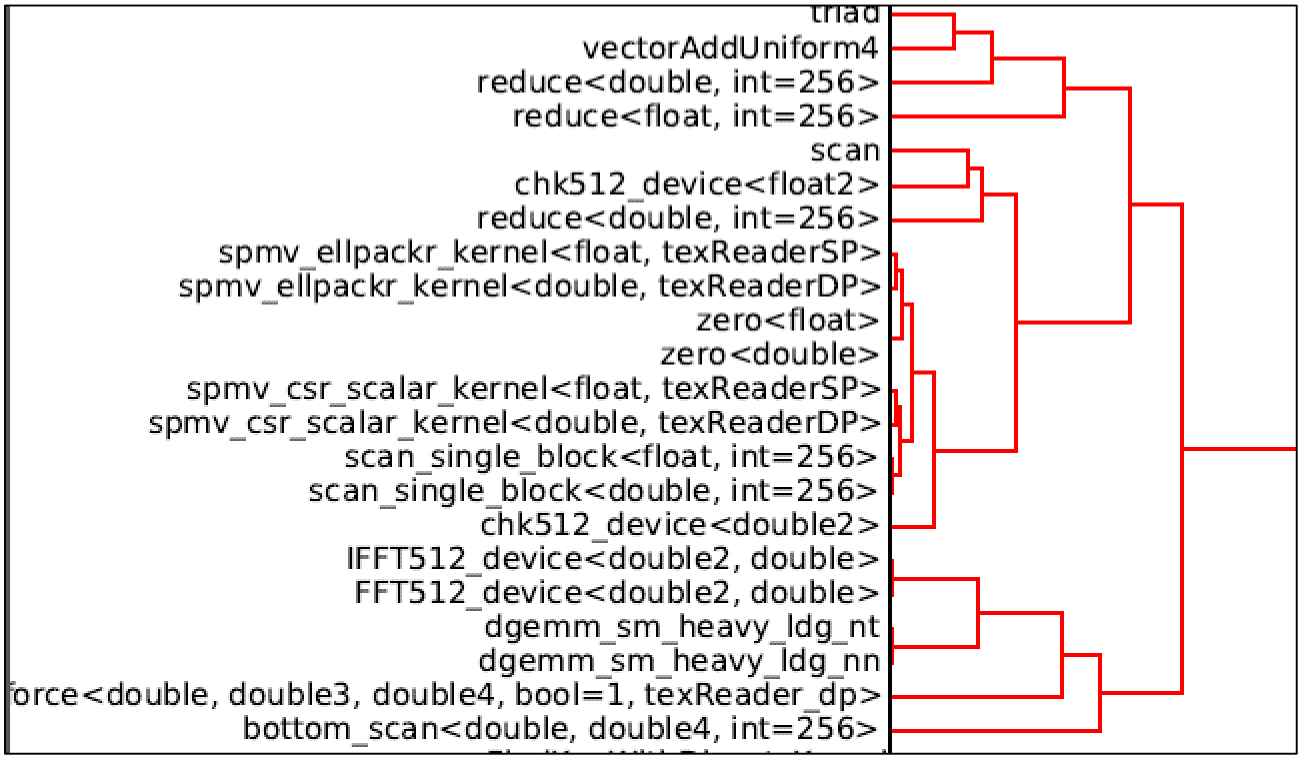
CHALLENGES

- Choosing the "right" applications to train the model
  - Models can be biased to the applications
- Choosing the "right" events to model
  - ~100 events within the GPU
  - Can track only 4-8 activities on a real hardware
- Choosing the "right" model
  - Linear mostly sufficient in the past

METHODOLOGY

- Selecting the right applications to train the model
  - Study several applications to see how they stress the various architectural components
    - Collect all relevant metrics
  - Remove redundancy in the dataset
    - Via principal component analysis
  - Hierarchical clustering to find similarity and difference
    - Choose one benchmark from each cluster

Methodology



Studied 100+ GPU kernels from 40+ applications to choose 6 dissimilar applications

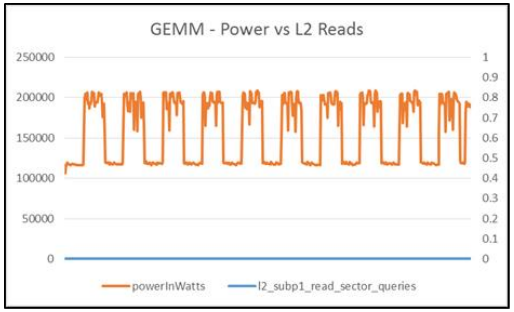
Methodology

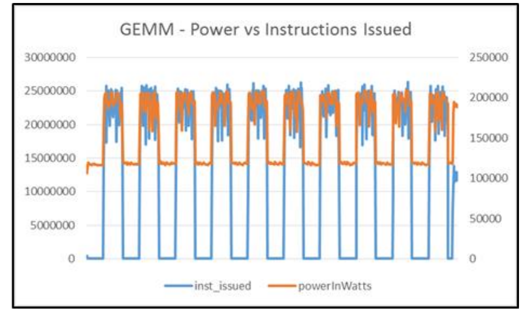
- Selecting the right performance counters (system activities) to construct the model

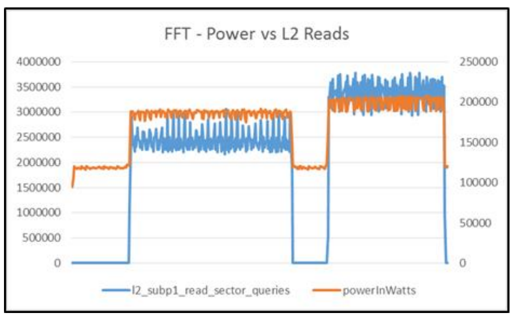
Courtesy: NVIDIA

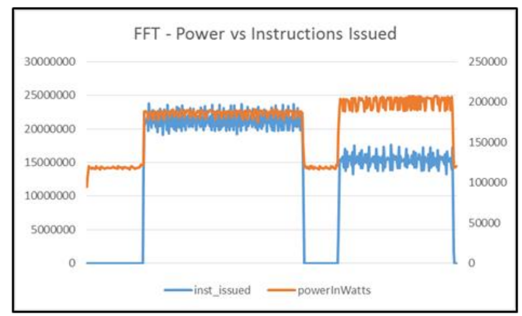
I-cache	**inst_issued**
Fetch	
Decode	
Schedule	
Dispatch	
Core	Inst_integer, inst_fp_32
DP Unit	Inst_fp_64
LD/ST Unit	Inst_compute_ld_st
SFU	Flop_count_sp_special
Register files	No direct proxy
Shared Memory	Shared_load_transactions, shared_store_transactions
L1 cache	Local_load_transactions, local_store_transactions
Read-only data cache	rocache_subp0_gld_thread_count_32b,
Texture cache	Tex_cache_transactions
L2 cache	L2_read_transactions, l2_write_transactions
DRAM	Dram_read_Transactions, dram_write_transactions

METHODOLOGY







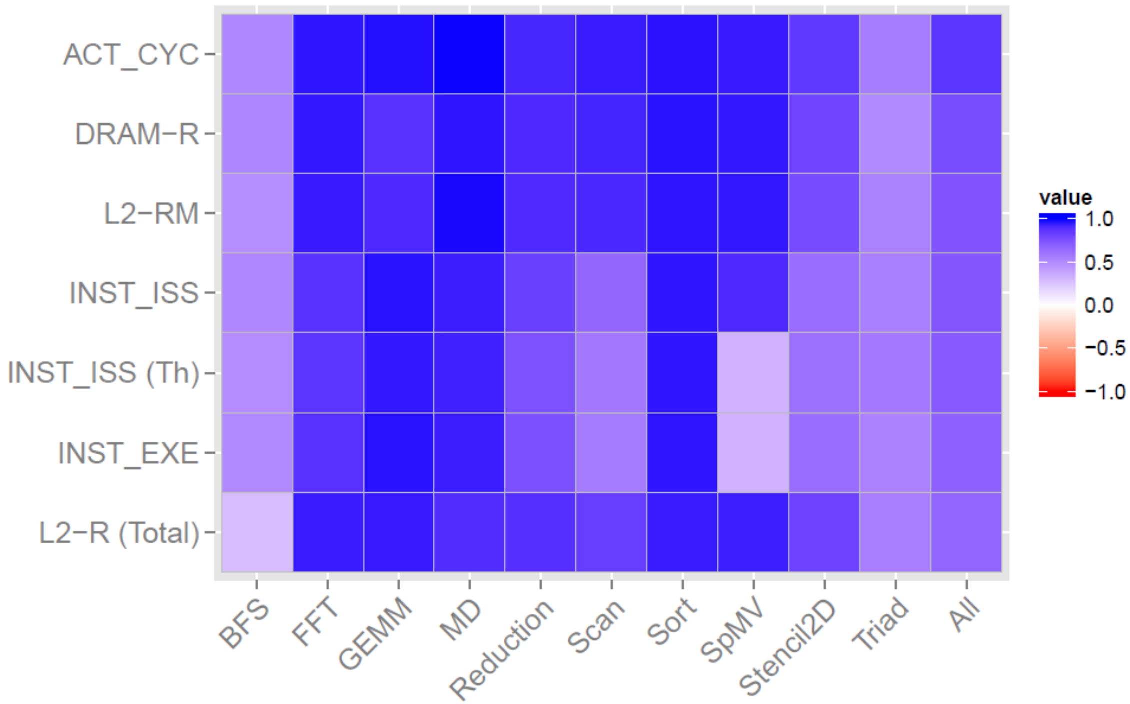


Application

System Activity

- Obtain performance counter and power values for several applications and various system activities

METHODOLOGY



- Calculate Person’s correlation coefficient between activities and power consumed
- Choose only events showing correlation greater than α (determined empirically)

METHODOLOGY

- Only limited events can be simultaneously profiled
  - Further limit events chosen

Input: E (Set of events showing high correlation)

Output: S (Set of events to be included in the model)

<u>Algorithm</u>

S ← Ø

for each event E_i (in decreasing order of correlation) in set E

    if E_i can be simultaneously profiled with events in Set S, **then**

        Calculate Pearson's correlation coefficient ρ_{ij} between E_i and all events S_j in Set S

        if $\rho_{ij} < \rho_{min}$ for all j, **then**

            $S \leftarrow S \cup E_i$

        end if

    end if

end for

METHODOLOGY

- Only limited events can be simultaneously profiled
  - Further limit events chosen

Input: E (Set of events showing high correlation)

Output: S (Set of events to be included in the model)

Al

Highest correlating events first

$S \leftarrow \varnothing$

for each event E_i (in decreasing order of correlation) in set E

  if E_i can be simultaneously profiled with events in Set S, **then**

    Calculate Pearson's correlation coefficient ρ_{ij} between E_i and all events S_j in Set S

    if $\rho_{ij} < \rho_{min}$ for all j, **then**

      $S \leftarrow S \cup E_i$

    end if

  end if

end for

METHODOLOGY

- Only limited events can be simultaneously profiled
  - Further limit events chosen

Input: E (Set of events showing high correlation)

Output: S (Set of events to be included in the model)

<u>Algorithm</u>

Simultaneously profilable with already chosen events

S ←

for each event E_i decreasing order of correlation) in set E

    if E_i can be simultaneously profiled with events in Set S, **then**

        Calculate Pearson's correlation coefficient ρ_{ij} between E_i and all events S_j in Set S

        if $\rho_{ij} < \rho_{min}$ for all j, **then**

            $S \leftarrow S \cup E_i$

        end if

    end if

end for

METHODOLOGY

- Only limited events can be simultaneously profiled
  - Further limit events chosen

Input: E (Set of events showing high correlation)

Output: S (Set of events to be included in the model)

<u>Algorithm</u>

$S \leftarrow \emptyset$

for ea

Should provide unique information not already available

if E_i can be simultaneously profiled with events in Set S, **then**

Calculate Pearson's correlation coefficient ρ_{ij} between E_i and all events S_j in Set S

if $\rho_{ij} < \rho_{min}$ for all j, **then**

$S \leftarrow S \cup E_i$

end if

end if

end for

METHODOLOGY

- Selecting the right models
  - After data collection treat as statistical modeling problem
  - Evaluate several mathematical functions

METHODOLOGY

MLR $m_1x_1 + m_2x_2 + c$	QMLR $m_1x_1 + m_{11}x_1^2 + m_2x_2 + m_{22}x_2^2 + c$
MLR+I $m_1x_1 + m_2x_2 + m_{12}x_1x_2 + c$	QMLR+I $m_1x_1 + m_{11}x_1^2 + m_2x_2 + m_{22}x_2^2 + m_{12}x_1x_2 + c$

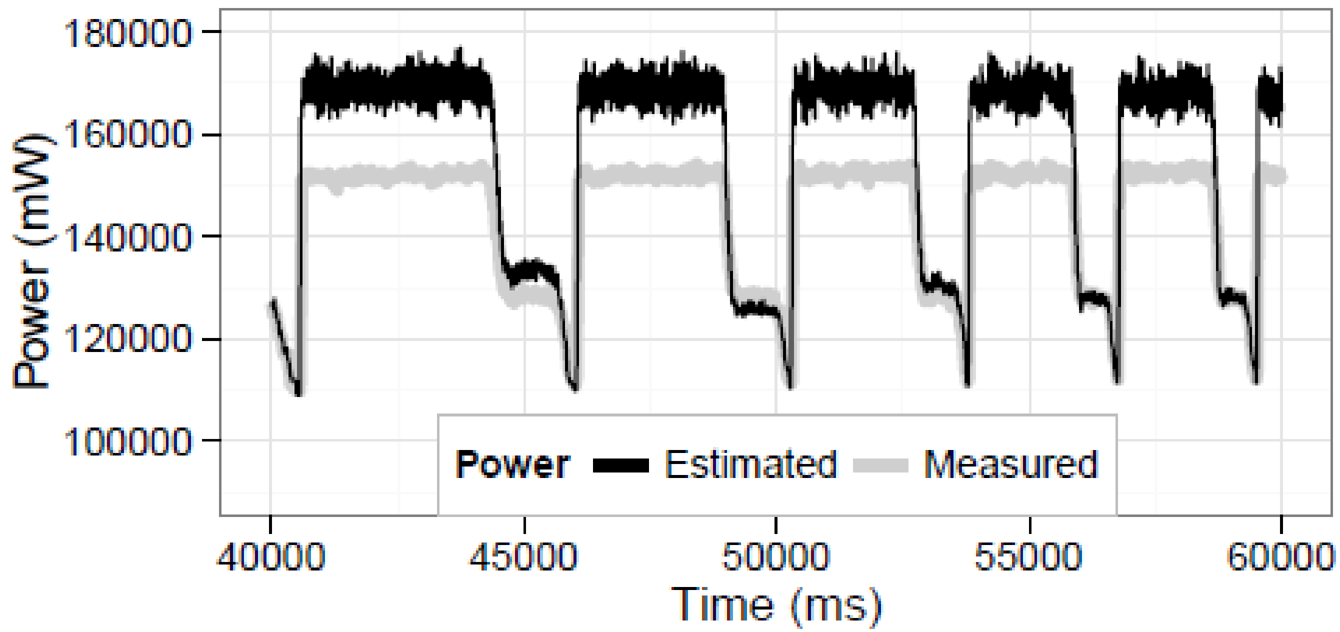
- Regression techniques to model power
- Evaluate different mathematical functions
  - Chosen based on CPU studies

RESULTS

Models	C2075	
	Basic	Temp-aware
SLR	17.96	8.59
MLR	11.59	**4.49**
MLR+I	14.02	6.83
QMLR	14.83	6.42
QMLR+I	19.05	10.31

- Multiple Linear Regression (MLR) model performs significantly better
- Effect of temperature on power is significant

RESULTS

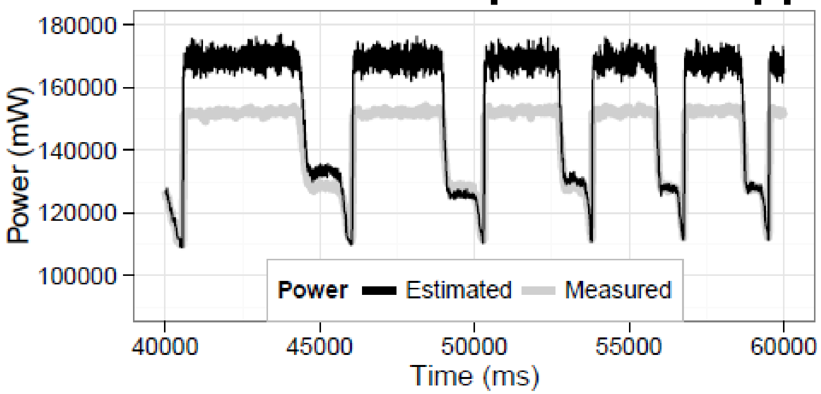


(a) QTC on C2075.

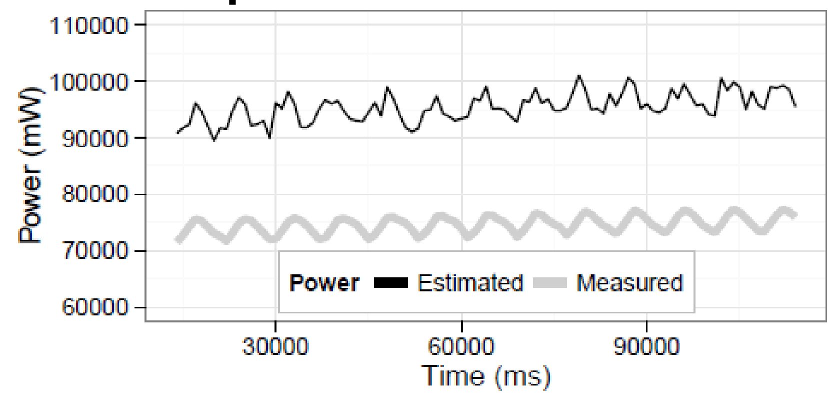
- Phase changes detected correctly
- Scope for improvement in exact power values

RESULTS

Power profile for application-independent models

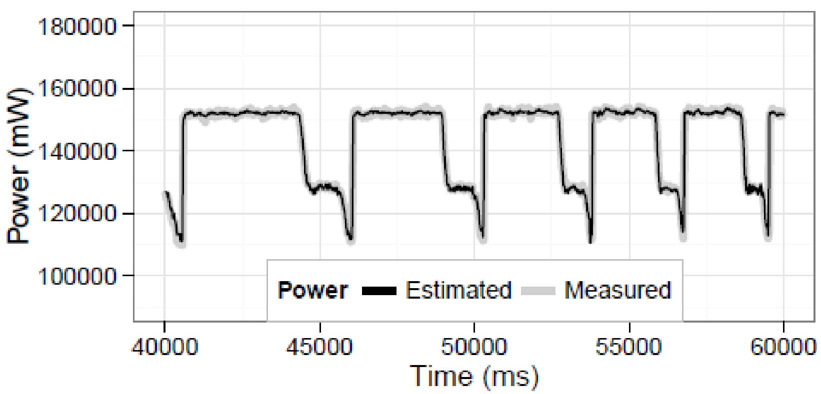


(a) QTC on C2075.

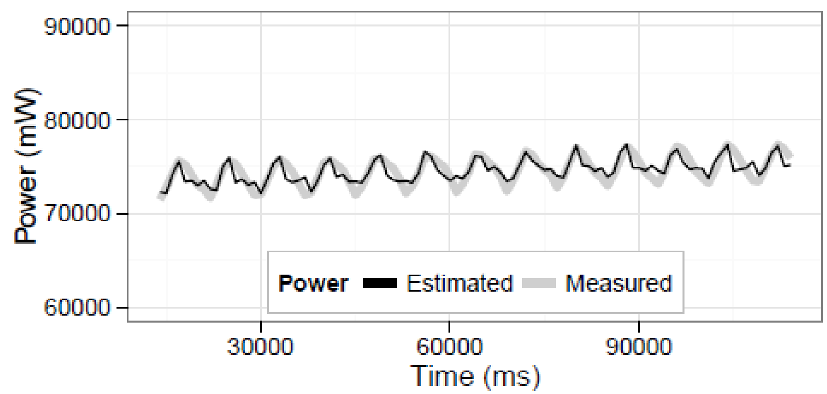


(b) Eigen Values on K20c.

Power profile for application-dependent models

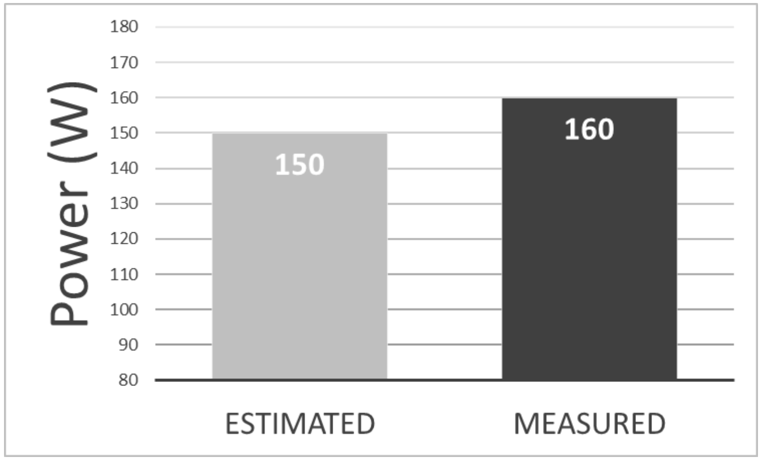


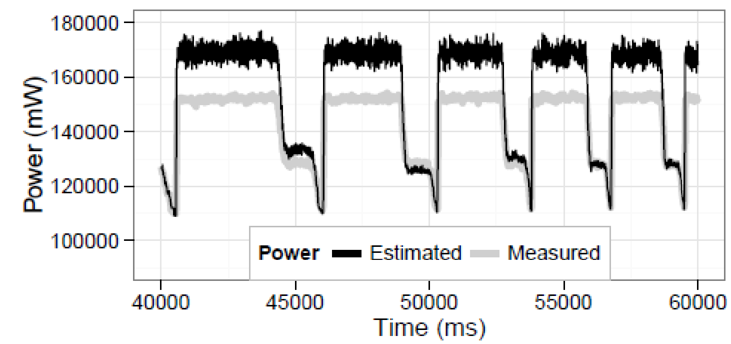
(a) QTC on C2075



(b) Eigen Values on K20c

CONTRIBUTION





(a) QTC on C2075.

- First accurate instantaneous power model on real GPU systems
  - 6% mean absolute error on real systems
  - 1% error from application-specific models

APPENDIX

Current and future work

- Develop a DVFS-agnostic model
  - Alternative: Model for each DVFS setting separately, but can be time consuming (ex. 55 settings in NVIDIA Titan)

- Use of DVFS-agnostic model for energy management at runtime
  - Achieve maximum performance under a power budget

SUMMARY OF RESULTS

Mean error % - Application-independent models

Models	C2075		K20c	
	Basic	Temp-aware	Basic	Temp-aware
SLR	17.96	8.59	21.67	9.44
MLR	11.59	**4.49**	18.66	8.29
MLR+I	14.02	6.83	14.74	**6.14**
QMLR	14.83	6.42	15.46	7.82
QMLR+I	19.05	10.31	19.56	8.86

Mean error % - Application-dependent models

Models	C2075		K20c	
	Basic	Temp-aware	Basic	Temp-aware
SLR	7.32	2.26	3.39	1.49
MLR	4.73	1.62	2.64	1.22
MLR+I	2.94	1.07	2.22	0.92
QMLR	3.04	1.08	2.24	0.96
QMLR+I	2.79	**1.02**	2.17	**0.88**

Objectives

- Which system activity to use?
- What type of mathematical function?
- Are the models portable across architectures?
- How much overhead?
- Are application-dependent models necessary?
  - Yes, application-dependent models significantly better
- How do we overcome associated overheads?

Conclusion

- Questions we answer
  - Which system activity to use?
    - Decided by our algorithm
    - Temperature as a factor
  - What type of mathematical function?
    - Linear expressions are better than quadratic expressions
  - Are the models portable across architectures?
    - No; micro-architecture dependent
  - How much overhead?
    - Negligible overhead for GPU-only application
  - Are application-dependent models necessary?
    - Generally useful
  - How do we overcome associated overheads?
    - Fewer samples sufficient for modeling at runtime

Related Work

Paper	Modeling Approach	Model Input	Run-time	Real system	Result
Nagasaka et al.	Multilinear Regression	14 Perf. counters	No	Real	4.7% avg. on 47 SDK + Rodinia
Song et al.	Neural Networks	13 Perf. counters	No	Real	2.1% avg. in select CUDA SDK
Abe et al.	Multilinear Regression	10 Perf. counters	No	Real	20% to 30%
McPAT (Lim et al.)	Analytical	10s of parameters	No	Real	7.7% and 12.8% for micro + merge
GPUWattch (Leng et al.)	Analytical + empirical	30 Perf. counters	Yes	Sim.	9.9% and 13.4% on micro + real