Putting a Dent into the Memory Wall: Combined Power-Performance Modeling for Memory Systems

Laura Carrington, Martin Schulz, & Ananta Tiwari

University of California, San Diego

San Diego Supercomputer Center

Performance Modeling and Characterization Lab (PMaC)

July 16th 2015

The PMaC Lab

Research the complex interactions between HPC systems and applications and use that to understand the factors that affect performance and power on current and projected HPC platforms.

Motivation: Entering the era of Exascale – many core systems with strict power budgeting

- **Trend towards multi- and many-core systems has accelerated over the last decade –**
  - **Multi-core designs allow for greater energy efficiency**
    - **Increase the compute performance through many simple and more energy savings cores**
  - **More cores/processor → less memory BW per core**
    - **In particular the off-chip bandwidth which is limited by pin constraints and slowly rising memory speeds**
- **Exascale comes with strict power budgeting**
  - **Power capping on the memory sub-system**
  - **Reduced power → reduced performance**

Understand HPC applications sensitivities to these performance/power changes to the memory sub-system

Performance & Power Models provide this understanding

Memory sensitivity models -
Modeling an HPC application's sensitivity to power/performance changes in memory sub-system

Models that capture:

- Application's sensitivity to reduced per core memory BW (e.g. many core)
- Application's sensitivity to power capped memory sub-system

Model development:

- Identify software parameters that determine application's sensitivity to changes
  - How sensitive are different types of computations?
  - Are certain algorithms less sensitive and would result in improved energy efficiency/performance?

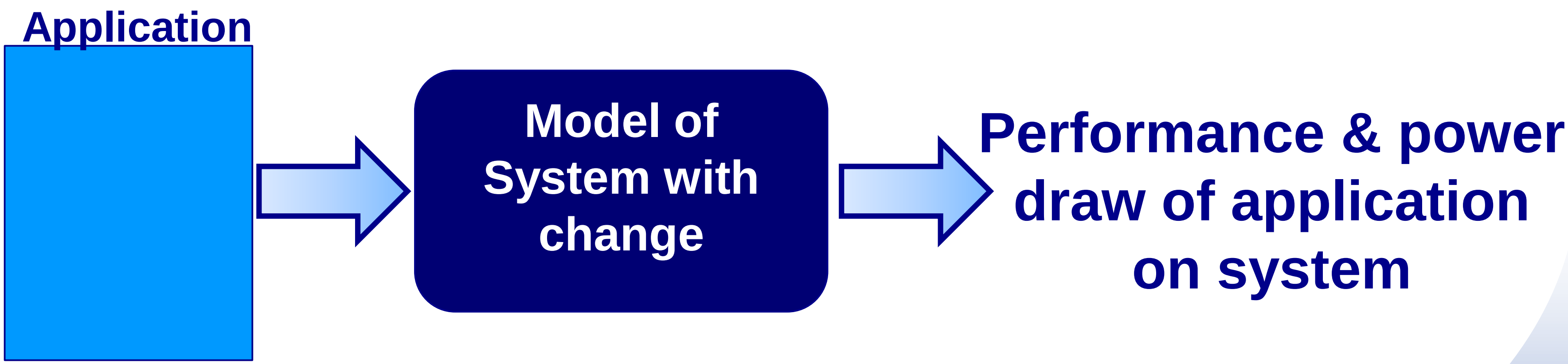
Overview

- Brief description of modeling technique:
  - Application characterization
  - Training the model
  - Validation of model
  - Results
  - Use cases

HPC Application Model Development – predicting power & performance

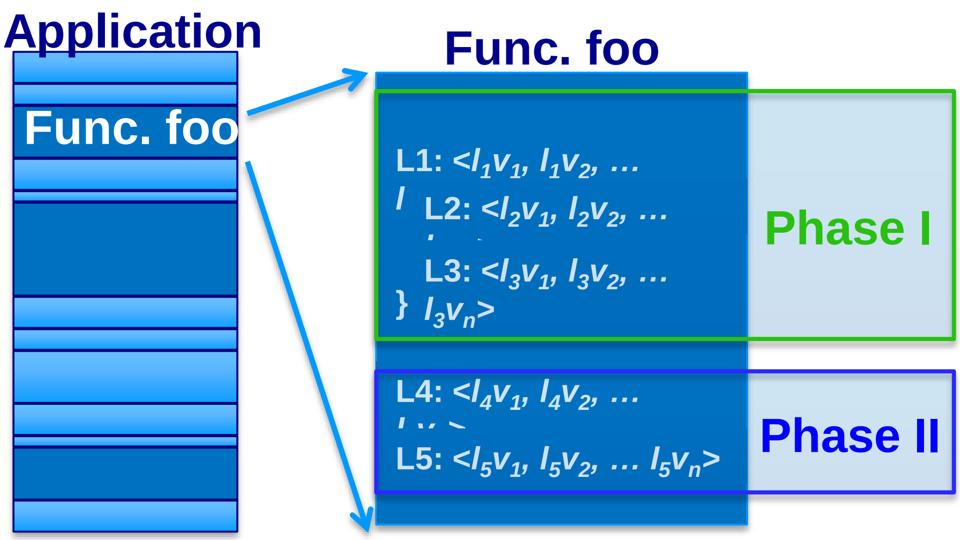
Main goal:

- **Develop model that predicts power and performance of application given a change in memory sub-system:**
  - **Reduced per core memory bandwidth**
  - **Power capped memory sub-system**



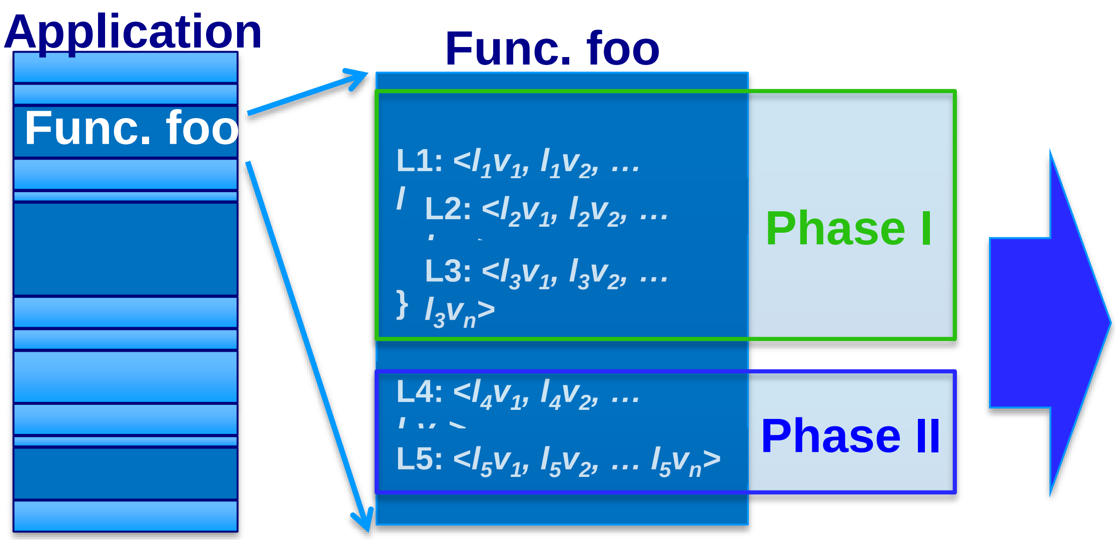
HPC Application Model Development – characterizing the application

- **Break large scale application into computational phases**



HPC Application Model Development – characterizing the application

- **Break large scale application into computational phases**
- **Identify software parameters that determine computational phase's sensitivity to change (e.g. characterization vector):**
  - **Data movement of computation (location, stride, type), computation metrics, etc.**

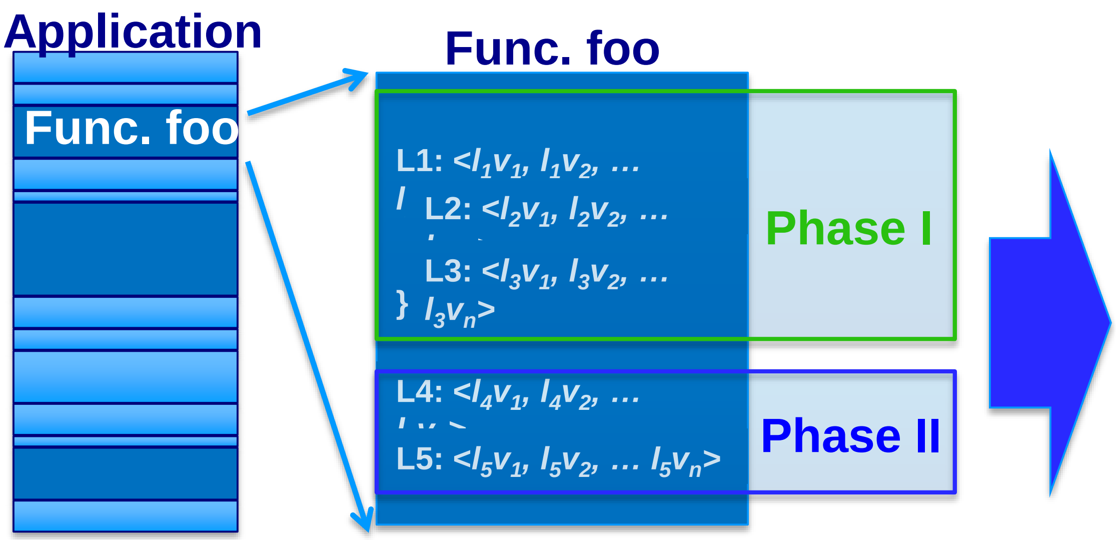


Characterization vector =

<data movement location, data movement stride, data movement type, computation metrics>

HPC Application Model Development – characterizing the application

- **Break large scale application into computational phases**
- **Identify software parameters that determine computational phase's sensitivity to change (e.g. characterization vector):**
  - Data movement of computation (location, stride, type), computation metrics, etc.

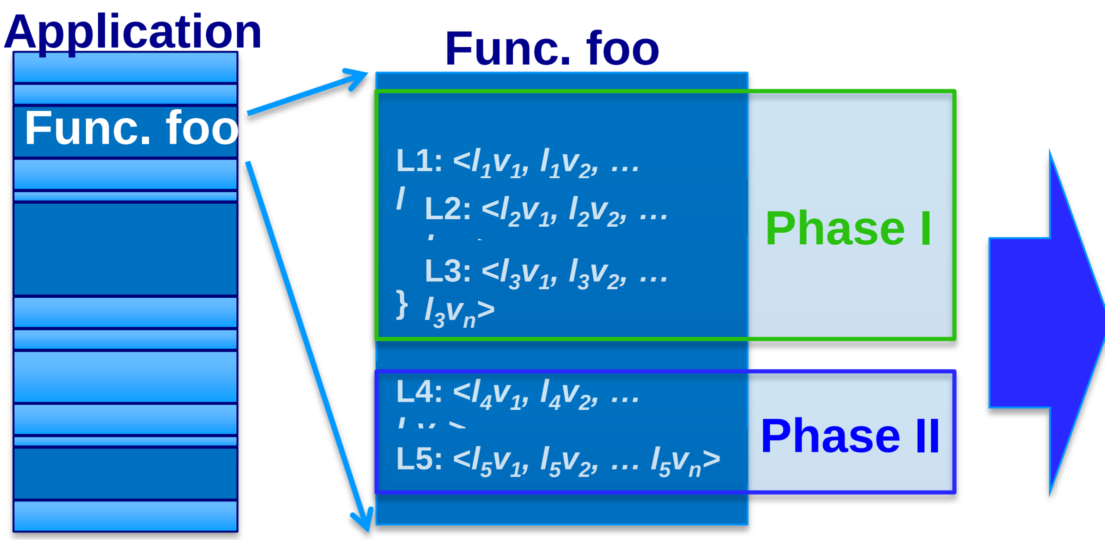


Characterization vector =

<cache hit rates, stride, load/store, FP/mem, # inst., etc. >

HPC Application Model Development – characterizing the application

- **Break large scale application into computational phases**
- **Identify software parameters that determine computational phase's sensitivity to change (e.g. characterization vector):**
  - **Data movement of computation (location, stride, type), computation metrics, etc.**



Characterization vector for each computational phase =

P1: <p_1v_1, p_1v_2, … p_1v_n>
P2: <p_2v_1, p_2v_2, … p_2v_n>
⋮

Automated full-scale production application collection via

PEBIL (PMaC's Efficient Binary Instrumentor for Linux) static & dynamic analysis

Modeling Methodology

- **Training set: use HPC computational kernels & benchmarks (applications are not part of training set)**
  - **Capture computation vector per kernel**

    Kernel 1: <$k_1v_1, k_1v_2, \ldots k_1v_n$>
    Kernel 2: <$k_2v_1, k_2v_2, \ldots k_2v_n$>
    Kernel 3: <$k_3v_1, k_3v_2, \ldots k_3v_n$>
    …
  - **Measure performance of each kernel for target system under change (e.g. reduced per core BW, power cap)**

    Kernel 1: <$Perf_{1,90}$,Cap90%,$k_1v_1, k_1v_2, \ldots k_1v_n$>
    <$Perf_{1,80}$,Cap80%,$k_1v_1, k_1v_2, \ldots k_1v_n$>
    <$Perf_{1,70}$,Cap70%,$k_1v_1, k_1v_2, \ldots k_1v_n$>
    ….
    Kernel 2: <$Perf_{2,90}$,Cap90%,$k_2v_1, k_2v_2, \ldots k_2v_n$>
    <$Perf_{2,80}$,Cap80%,$k_2v_1, k_2v_2, \ldots k_2v_n$>
    <$Perf_{2,70}$,Cap70%,$k_2v_1, k_2v_2, \ldots k_2v_n$>
    …

Modeling Methodology

Training set:

$<Perf_{1,90}, Cap90\%, k_1v_1, k_1v_2, \ldots k_1v_n>$
$<Perf_{1,80}, Cap80\%, k_1v_1, k_1v_2, \ldots k_1v_n>$
$<Perf_{1,70}, Cap70\%, k_1v_1, k_1v_2, \ldots k_1v_n>$
….
$<Perf_{2,90}, Cap90\%, k_2v_1, k_2v_2, \ldots k_2v_n>$
$<Perf_{2,80}, Cap80\%, k_2v_1, k_2v_2, \ldots k_2v_n>$
$<Perf_{2,70}, Cap70\%, k_2v_1, k_2v_2, \ldots k_2v_n>$
…

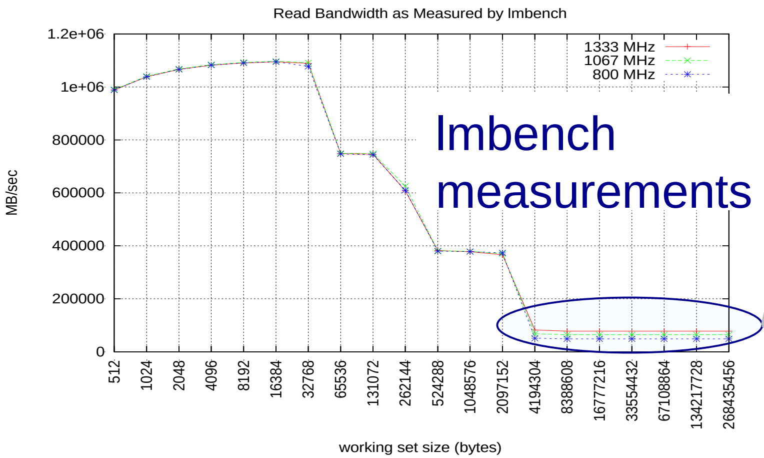
- **Modeling technique Cubist (e.g., tree of linear regression models) & Gradient Boosting**
- **Prevent over-fitting:**
  - **Split the empirical dataset into training and validation sets**
    - **60%-40% split: 60% used for training the model and 40% for validation**
  - **10-fold cross validation to avoid over-fitting during model training**
- **Variable importance analysis to determine which predictors have the most impact on performance degradation**

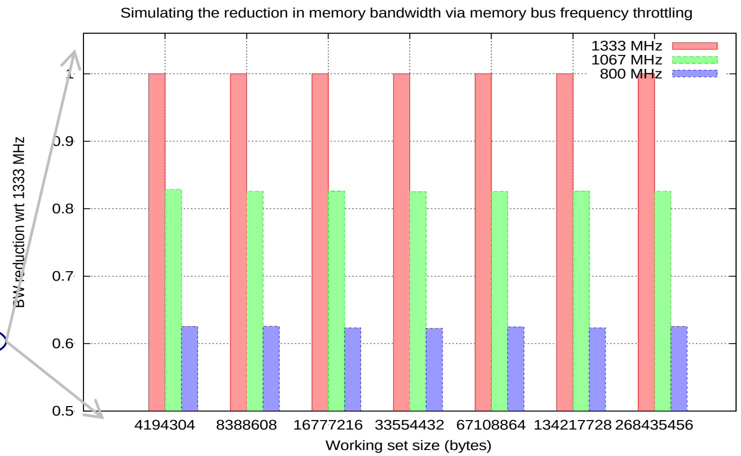
Modeling reduced per core memory BW

System Configuration for Validation

testing with the lmbench benchmark

- **How to approximate reduced per core memory BW?**
  - **Change the memory bus frequency (set at boot time)**
- **One node of the SDSC's Gordon Supercomputer**
  - **Sandy Bridge – 2 procs, 8 cores/proc, 64GB DDR3-1333MHz memory**
  - **Available bus freq. – 1333 MHz (max & default), 1067 MHz, 800 MHz**





1067MHz:17.5% reduction in BW when freq is reduced by 20%.

800MHz: 37.7% reduction in BW when frequency is reduced by 40%

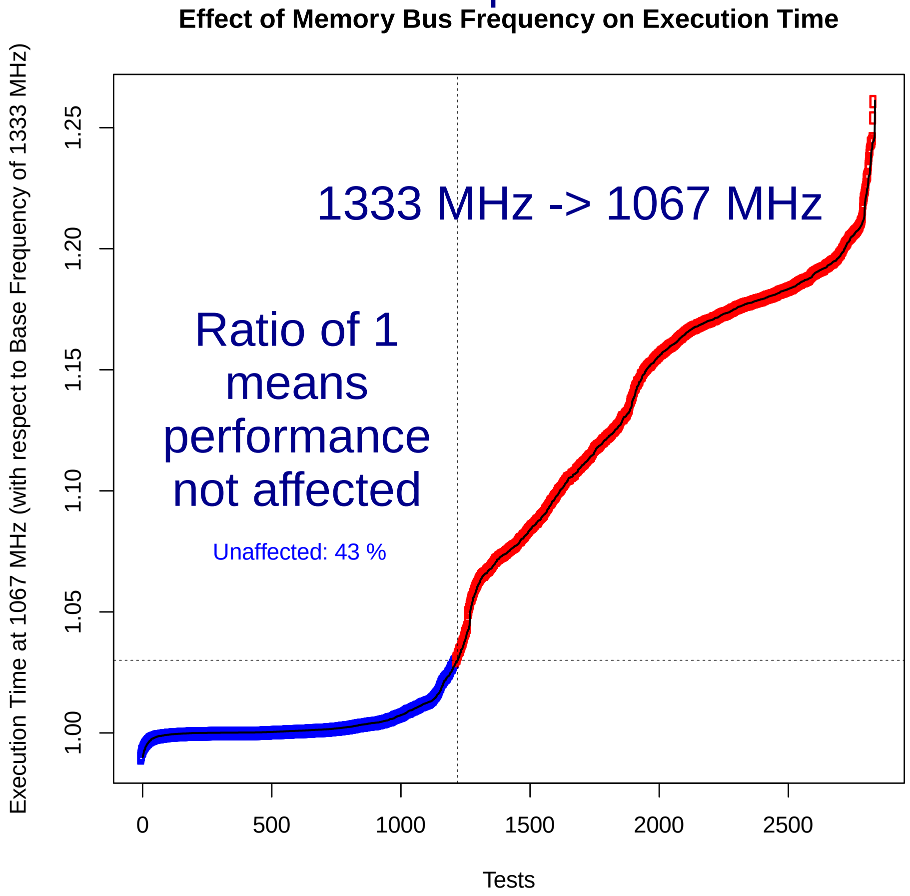
MEM BW (theoretical) = *F x L x W x I*
where,
***F*: DRAM clock frequency**
L: Number of lines per clock (2 for *DDR*N)
W: Bus Width (64 bits)
I: Number of Interfaces (2: dual channel)

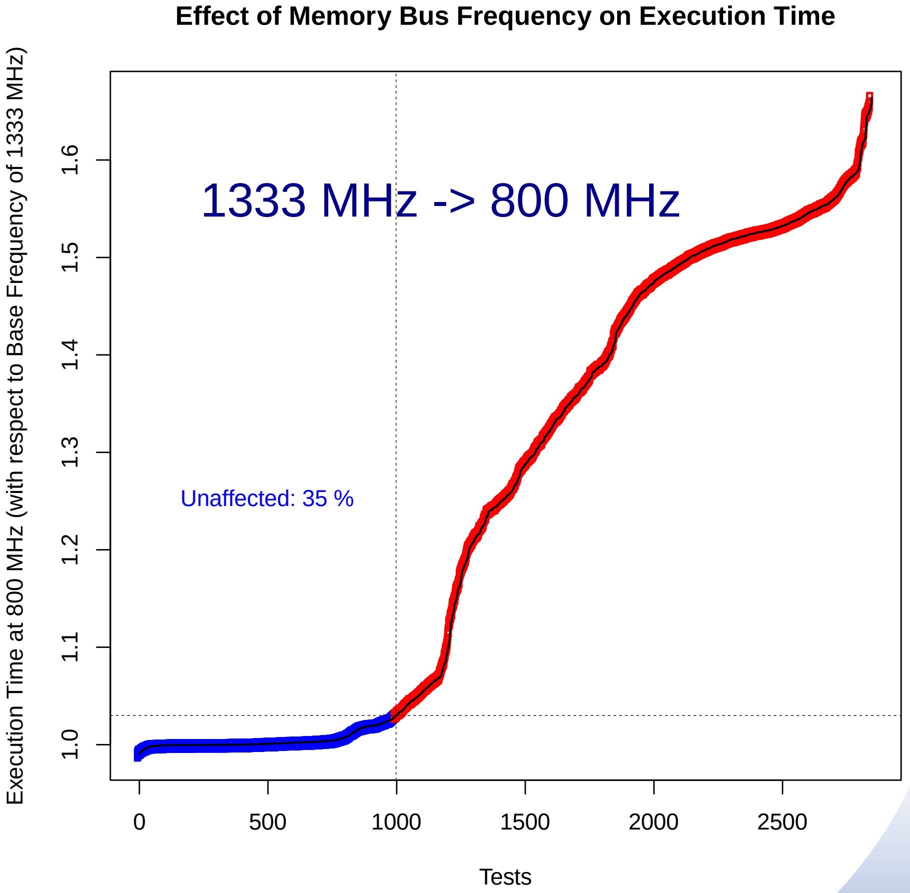
SDSC SAN DIEGO SUPERCOMPUTER CENTER

PMaC Performance Modeling and Characterization

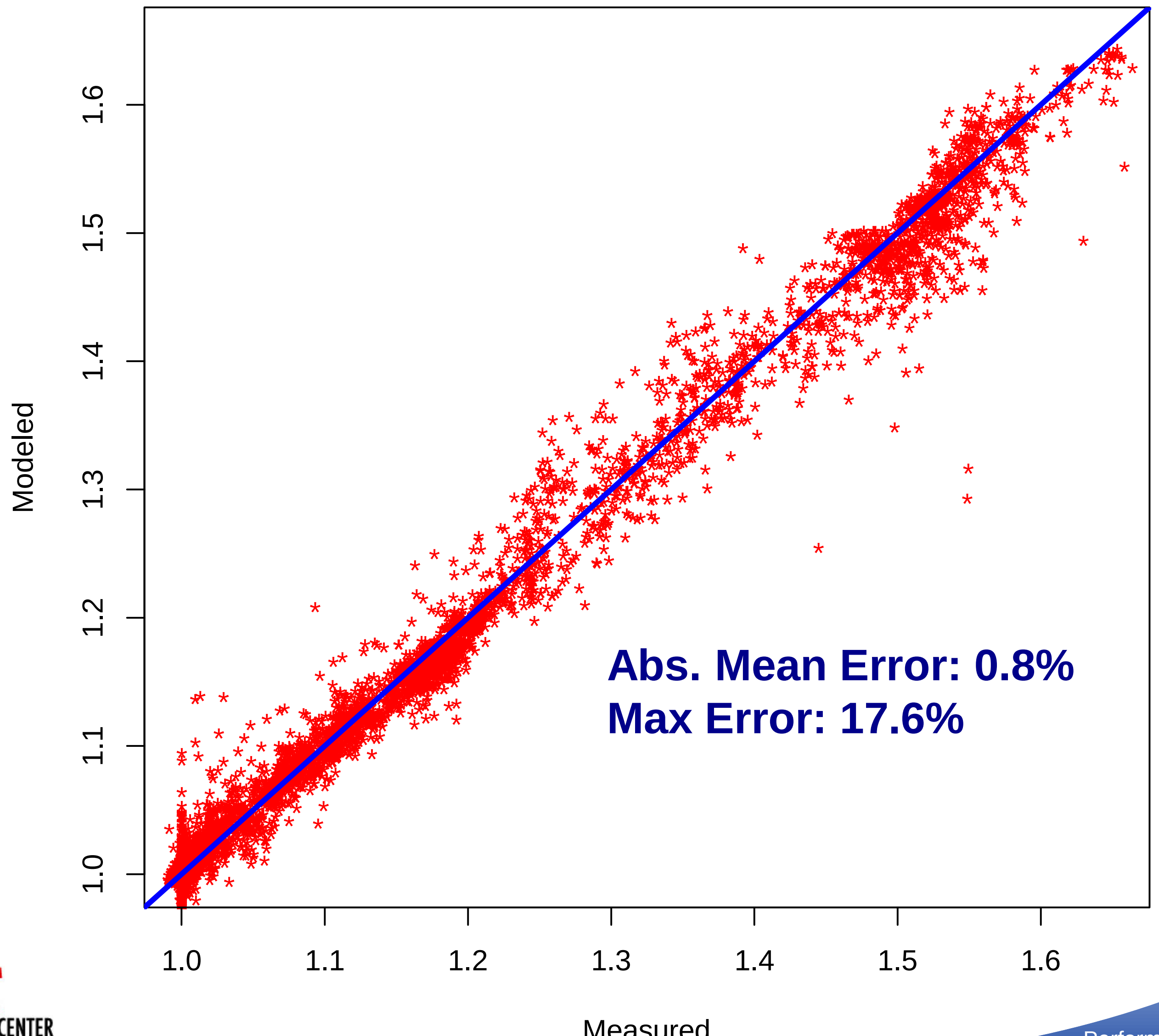
Are all computations sensitive to per core bandwidth?

Ratio of training set kernels and benchmark's performance at max relative to reduced BW





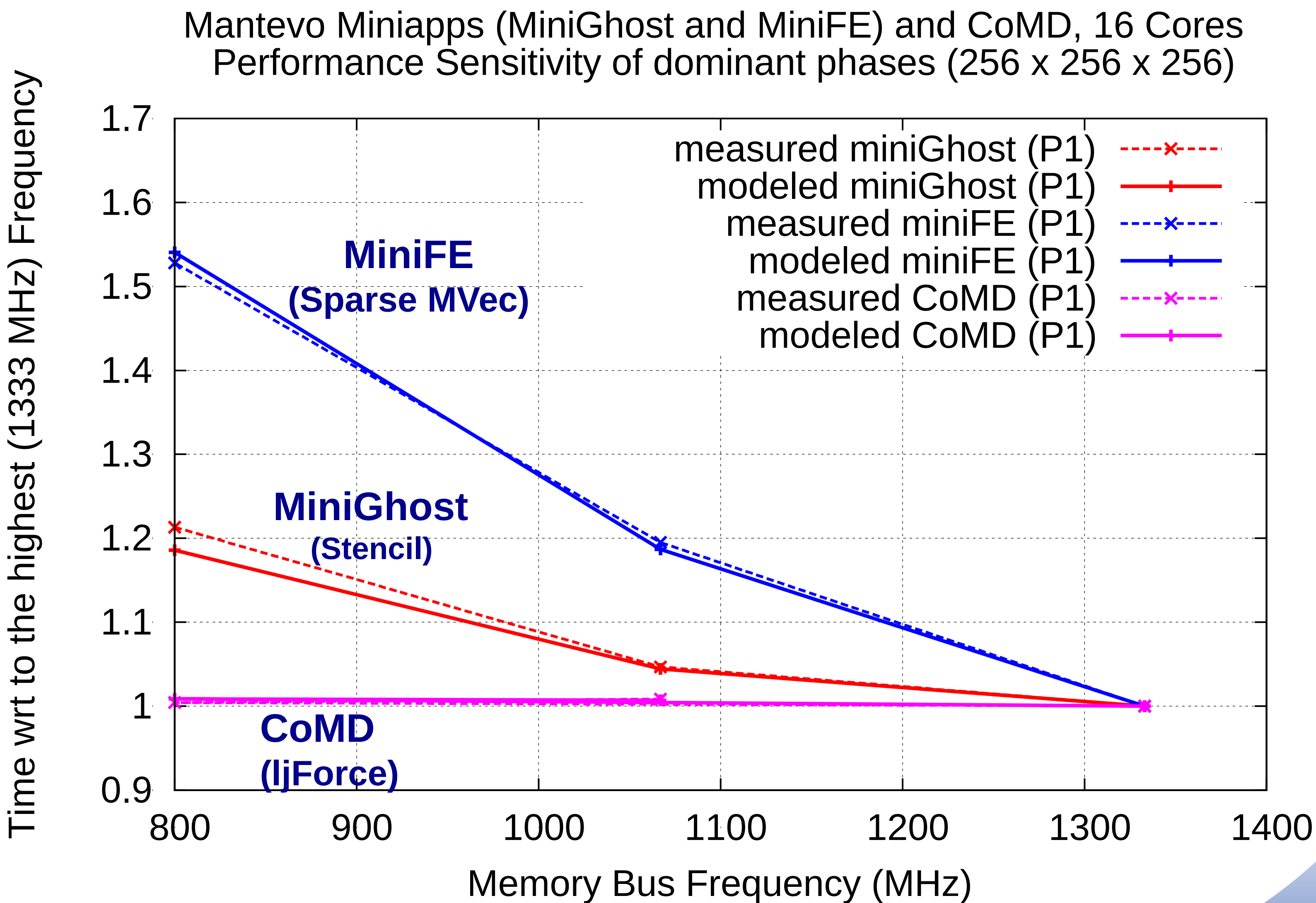
Model accuracy on training Set for reduced per core memory BW model
prediction of ratio of degradation



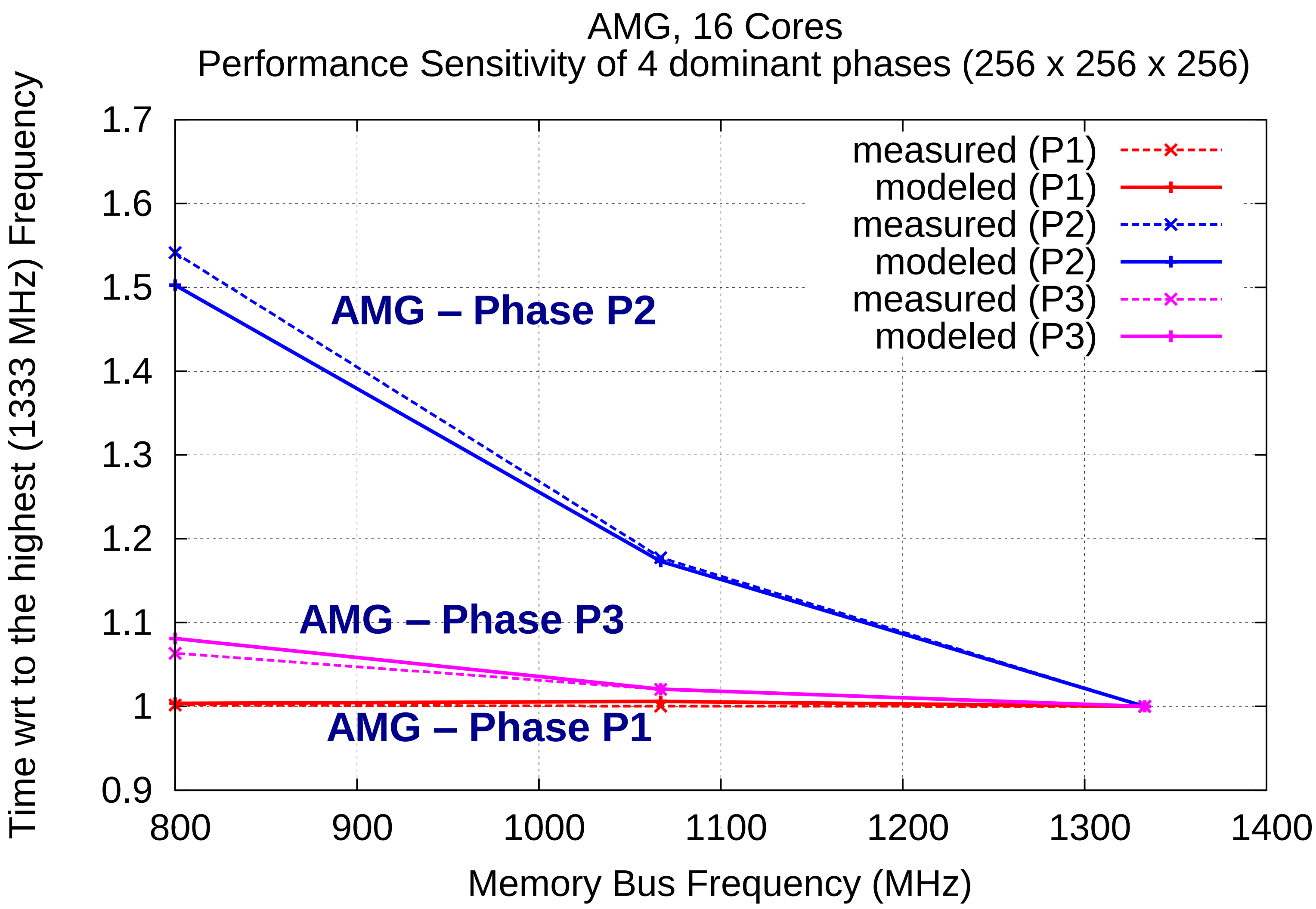
Evaluation of modeling methodology

- Evaluation uses several applications –
  - NPBs (CG, LU, FT and MG)
  - SMG2000 (Semi-coarsening multigrid)
  - AMG (Algebraic multigrid)
  - Mantevo Miniapps
    - MiniFE
    - MiniGhost
  - CoMD
- Hotspot selection based on dynamic instruction count attributed to loops
- Verify model on dominant loop(s)/phase(s) of application→ collect characterization vector

Model validation – Mantevo & CoMD

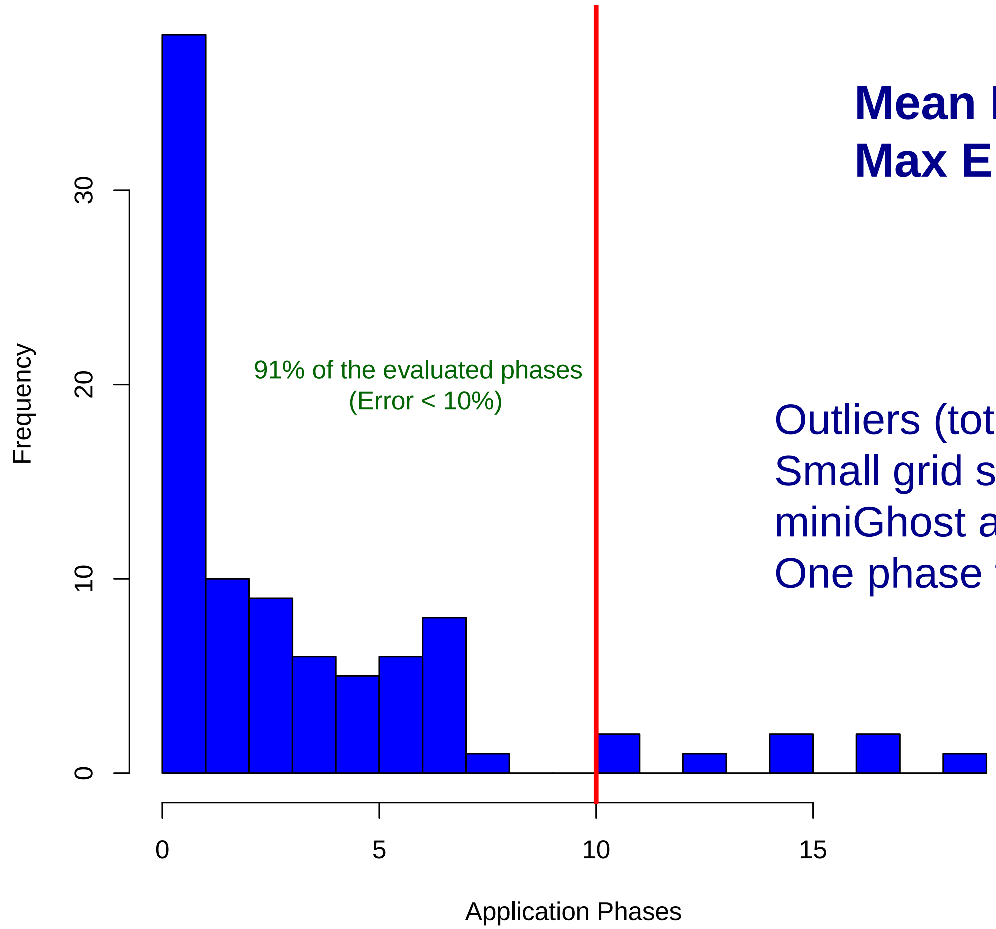


Model validation: AMG



Model validation : Mean Error of all phases

Histogram of Prediction Accuracy for Test Application's Phases



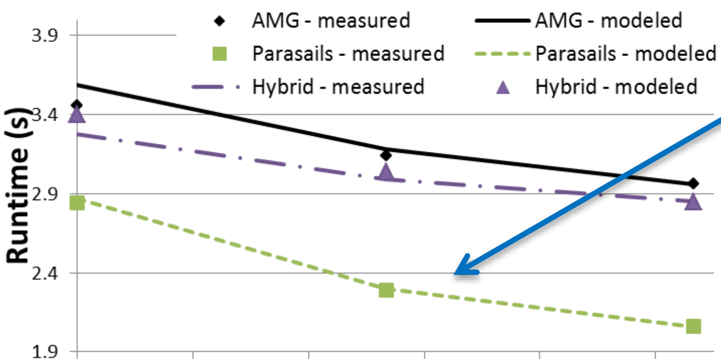
Mean Error: 3.8%
Max Error: 18%

Outliers (total 8 phase-freq pairs):
Small grid size runs for
miniGhost and SMG2000
One phase from NPB FT.

Use-Case: Algorithm selection

- **Exascale systems will most likely have reduced per core memory bandwidth-models help identify optimal algorithms for these systems**

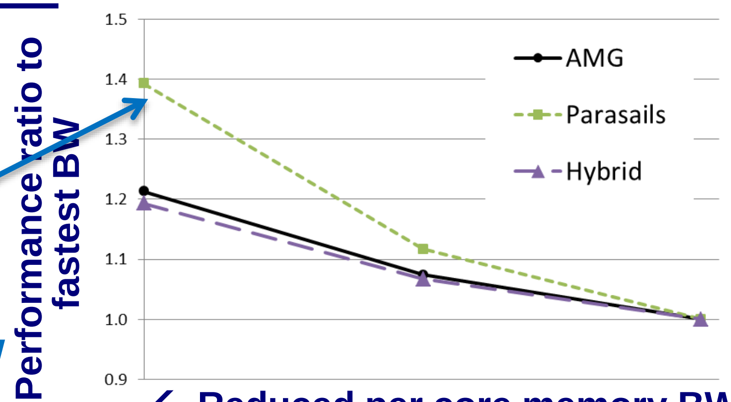
Determining the correct algorithm for future Exascale systems using performance models



← Reduced per core memory BW

Parasails has fastest performance

Parasails most sensitive to reductions in BW



← Reduced per core memory BW

Performance models can identify algorithmic choices that are less optimal as hardware changes in future systems.

Test bed for power capping

- Dual Intel SandyBridge processor, 8 cores per processor, 64 GB RAM, Turbo-Boost off, SMT off

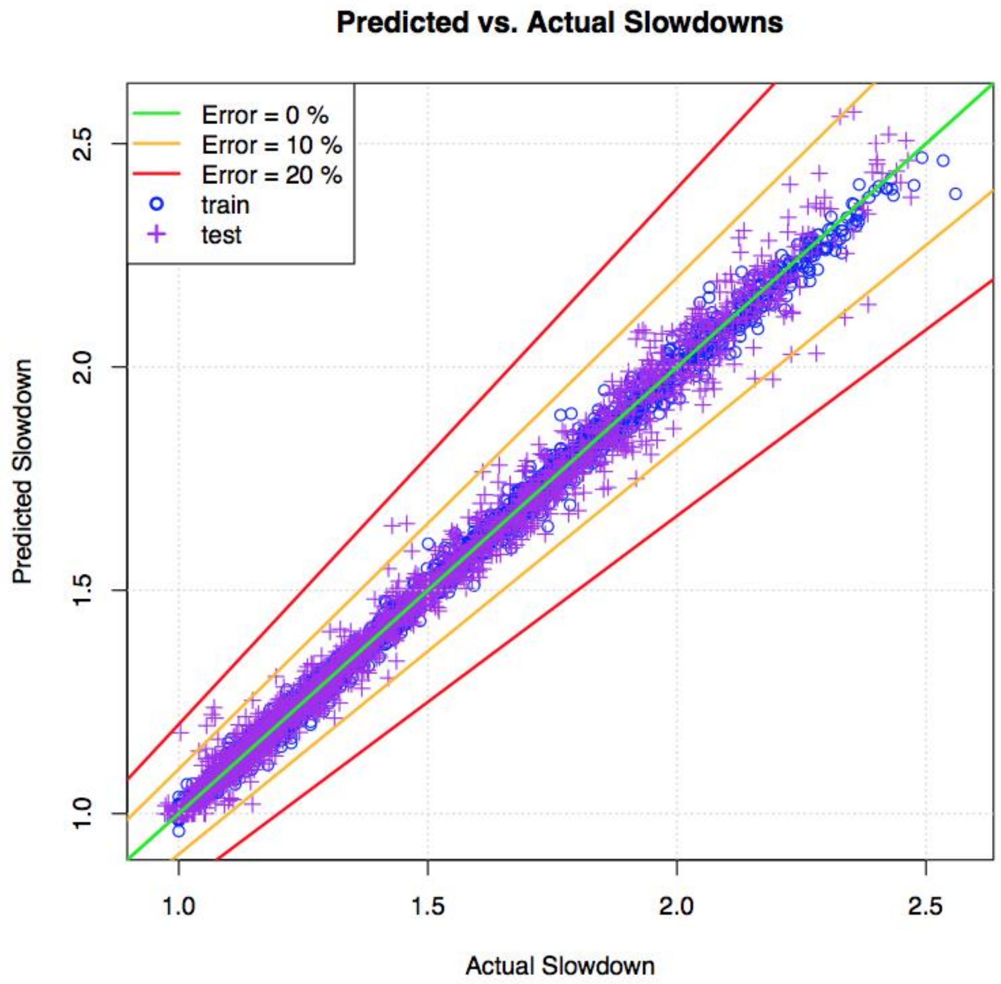
- Power capping using Running Average Power Limit (RAPL) interface
  - Enables the collection of (modeled) power measurements for CPU and DRAM subsystems
  - Allows users to set power limits on these domains and the underlying hardware infrastructure enforces these power limits

Results: Model Accuracy for power capping

60%-40% split of the empirical data.

60% used for training the model

40% makes up the test/validation set.



SDSC
SAN DIEGO SUPERCOMPUTER CENTER

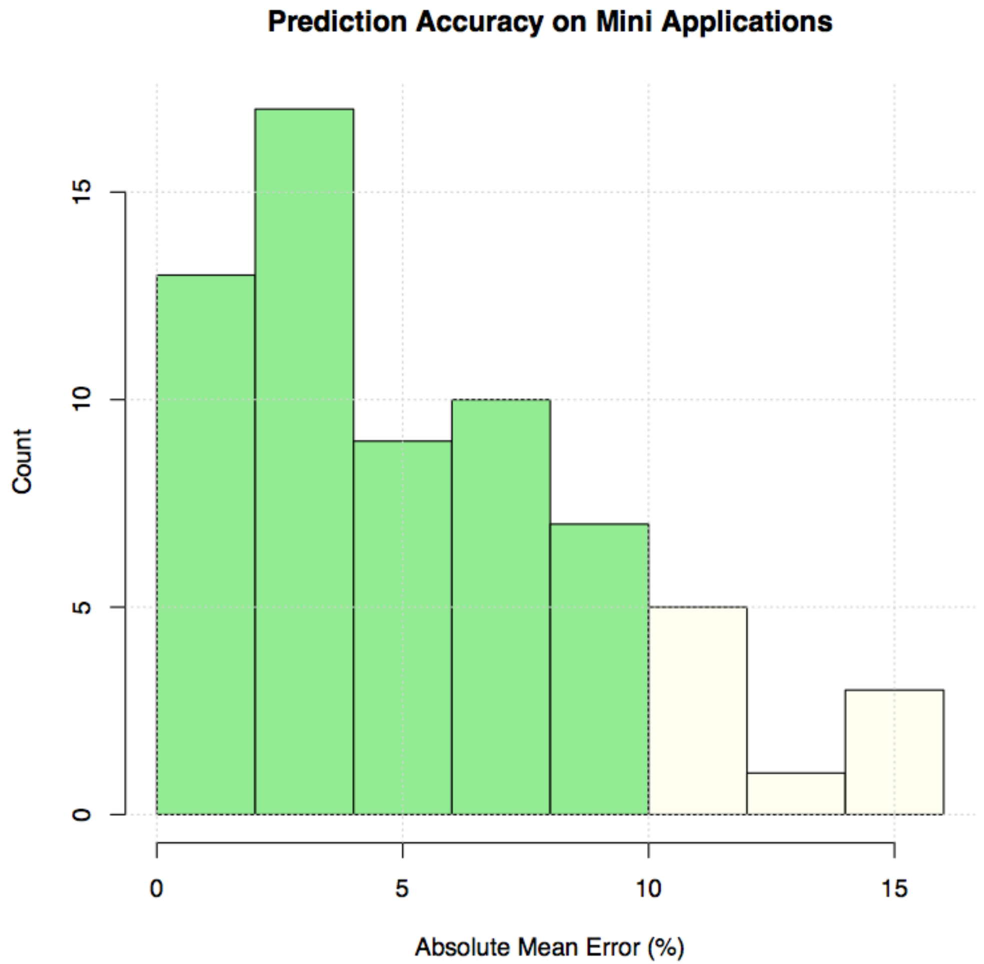
PMaC
Performance Modeling and Characterization

Results: Evaluation on Mini-apps

- Evaluation done using two mini-apps –
  - MiniGhost (Finite Difference)
  - CoMD (Molecular Dynamics)
- Multiple input sizes
- Loop selection based on dynamic instruction count attributed to loops
  - Compare actual performance loss due to different power caps to modeled performance loss

Results: Evaluation on Mini-apps

Average absolute error: 6%



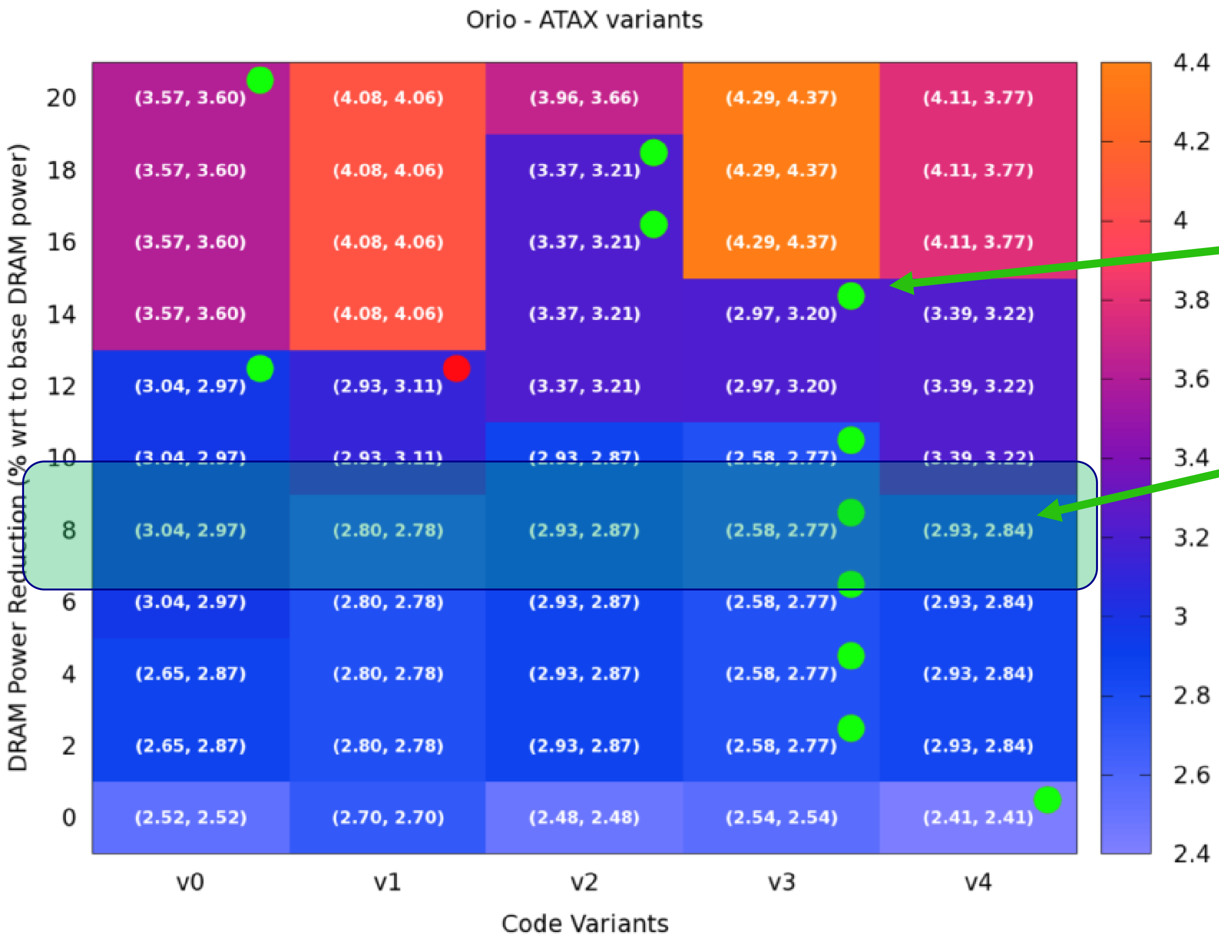
Use-case: Auto-tuning in power-capped environment

- Search-based auto-tuning framework
  - Generate a set of alternative implementations of a given piece of code and select the one that performs the best
  - The code variant that performs the best in base-case (i.e., with no power capping) might not be the best in power-capped environment
  - Model can be used to inform such explorations
- Demonstration using one computation kernel
  - Select 100 random variants, evaluate the performance of those variants in multiple power capping levels

Models identify optimal variant for given power cap

Use-case: Auto-tuning

- Models identify code optimal code variant for a given power budget.
- X-axis shows small subset of 100 code variants, Y-axis different DRAM power reductions relative to base
- Green dot shows fastest code variant for each power bounds
- Red dot only case where models didn't identify fastest code variant



Fastest code variant for 14% power cap

Row is performance for each variant under that % capping

SAN DIEGO SUPERCOMPUTER CENTER

PMaC
Performance Modeling and Characterization

Summary

- Exascale will have multi-core designs and power capped environment that will expose new performance challenges to HPC application developers
- Models help developers and centers enhance their readiness for Exascale systems and beyond;
  - For their key workloads, models can identify code-sections that need to be re-examined to exploit drastic changes in Exascale hardware design
- Presented models that are highly accurate in predicting the performance sensitivity of various HPC computations for power caps DRAM domains as well as reduce per core memory BW
- Presented use cases for both types of models.

Thank you for your attention!

Acknowledgements

This work was supported in part by the DOE Office of Science, Advanced Scientific Computing Research, under award number 62855 “Beyond the Standard Model – Towards an Integrated Modeling Methodology for the Performance and Power”; PNNL lead institution; Program Manager Karen Pao. The authors acknowledge partial support from LLNL under subcontract B600667. This work was also supported in part by the DoD and used elements at the Extreme Scale Systems Center, located at ORNL and funded by the DoD.Partial support also came from the DOE Office of Science through the SciDAC award titled SUPER (Institute for Sustained Performance, Energy and Resilience). Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-666558).

For further details:

1. Tiwari, A., Schulz, M., and Carrington, L. (2015) Predicting Optimal Power Allocation for CPU and DRAM Domains. in *Workshop on Parallel and Distributed Scientific and Engineering Computing*, India
2. Tiwari, A., Gamst, A., Laurenzano, M., Schulz, M., and Carrington, L. (2014) Modeling the Impact of Reduced Memory Bandwidth on HPC Applications. in *EuroPar14* ***nominated top 5 papers***

Questions