www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PoTrA: A framework for Building Power Models For Next Generation Multicore Architectures

Part II: modeling methods

Outline

- Background
- Known pitfalls
- Objectives
- Part I: Decomposable power models: Single Core
- Part II: Decomposable power models: DVFS
- Part III: Decomposable power models: CMP
- Part IV: Decomposable power models on Virtualized Systems

Background: Modeling

- “All models are wrong but some are useful”
- In general, models can be useful for:
  - Prediction: perform estimations
  - Understand better the modeled system
- In our field, power models, are also useful for:
  - Detect power phases
  - Break-down the power consumption of the platform

Background: The big picture

Simulation-Based Models

- **Requirements**:
  - Cirtuit/RTL knowledge
- **Issues:**
  - Unable for online predictions
  - Not portable
- **Advantages**:
  - Detailed and high granularity

Detailed Analytical Models

- **Requirements:**
  - Detailed microachitectural and layout knowledge
  - Performance Counters
- **Tradeoff between:**
  - Simulation and High-level black box
  - Decomposable but more complex

High-Level Black-Box Models

- **Requirements**:
  - High-Level Generic Events
    - Performance Counters
    - OS events
- **Issues**:
  - Unable to breakdown
  - Less accurate
- **Advantages:**
  - Simple and very fast

Background: The big picture

Simulation-Based Models

- **Requirements**:
  - Cirtuit/RTL knowledge
- **Issues:**
  - Unable for online predictions
  - Not portable
- **Advantages**:
  - Detailed and high granularity

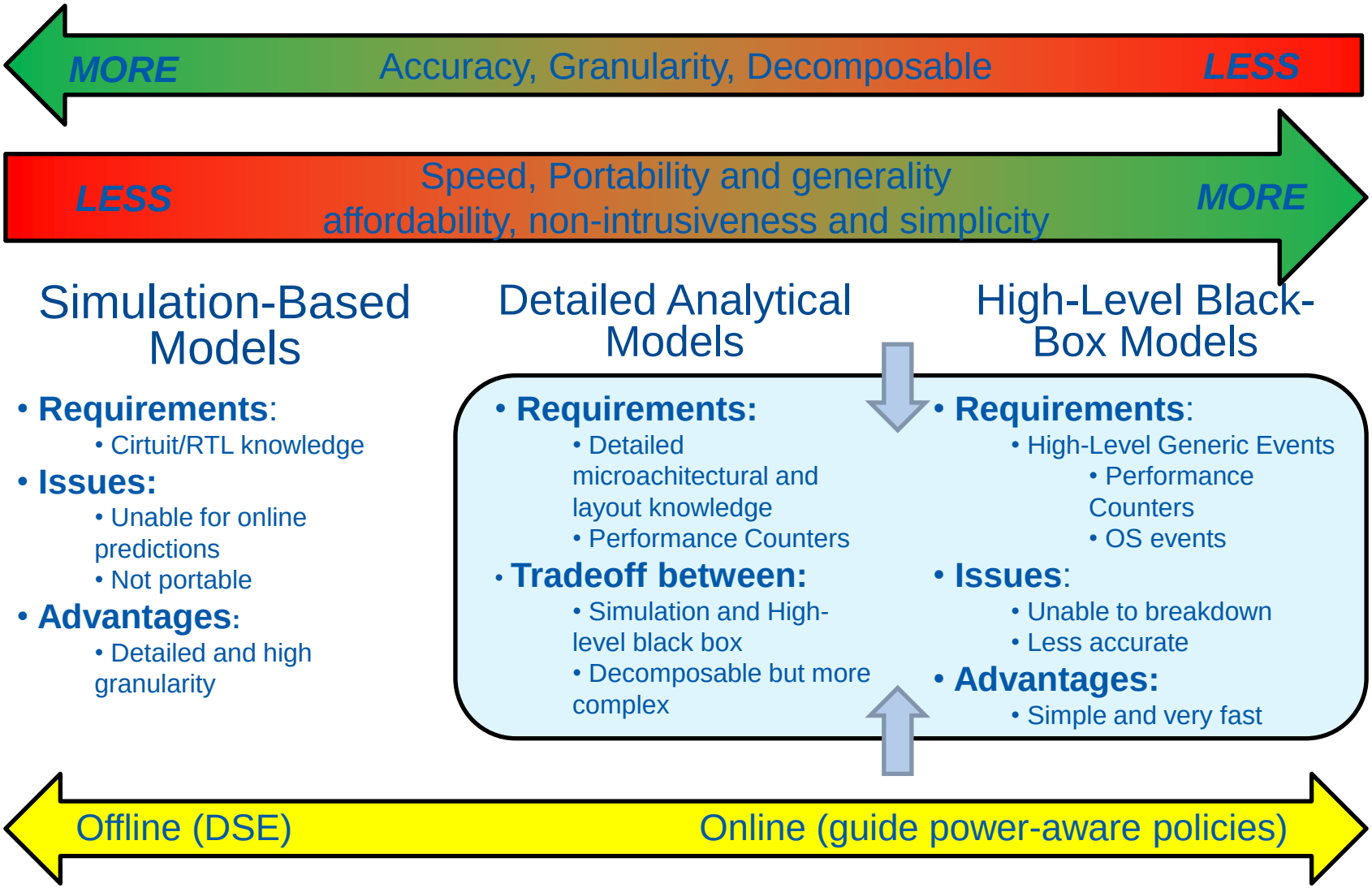
Detailed Analytical Models

- **Requirements:**
  - Detailed microachitectural and layout knowledge
  - Performance Counters
- **Tradeoff between:**
  - Simulation and High-level black box
  - Decomposable but more complex

High-Level Black-Box Models

- **Requirements**:
  - High-Level Generic Events
    - Performance Counters
    - OS events
- **Issues**:
  - Unable to breakdown
  - Less accurate
- **Advantages:**
  - Simple and very fast

Background: The big picture



Background: Interesting model properties

- Accuracy
  - An inaccurate model is useless
  - Error up to X% are accepted by the community
- Fast evaluation
  - Required for on-line application of the model
- Affordable, easy to deploy
  - Quickly target new systems and speed-up research
- Informative (decomposable)
  - Better understanding of the modeled system
- Responsive
  - Detection of power phases
- Robust (generality, workload independent)
  - Valid for extreme situations or for different power modes

Background: Counter-based power models

- Counter-based power model properties (by design):
  - Fast to evaluate → Compute a formula
  - Easy to deploy → Performance counters are common
- Counter-based power models are **empirical models**
  - i.e. the models are trained using real data
- Common methodology:
  - 1.- Design the model:
    - Select the counters
    - Define the "formula" of the model (#inputs)
  - 2.- Gather training data (inputs ←→ power measurements)
  - 3.- Generate the model
    - Multiple linear regression
  - 4.- Validate the model
    - Check average on the validation data set
    - If average error high → fine tune:
      - Redefine the model inputs (apply transformation to model inputs, select other inputs)
      - Piece-wise models (observe data to select splitting point)
      - Manual tuning
- The approach used in each step affects the properties of the model
  - Accuracy? Decomposability? Robustness? Responsiveness?

Background: Common modeling pitfalls

- Pitfall 1: Model the system as a “black-box”
  - Loose of opportunities to gain more insights about the modeled system
    - We know how the modeled system work, why do not use that knowledge to design a more realistic power model?
  - Black-box models tend to be biased towards training set properties
  - Black-box models are difficult to understand by experts and layman, i.e. it is impossible to interpret the model
    - E.g. counter-intuitive model factors. Common: why floating point activity has a negative factor? Is floating point generating energy?

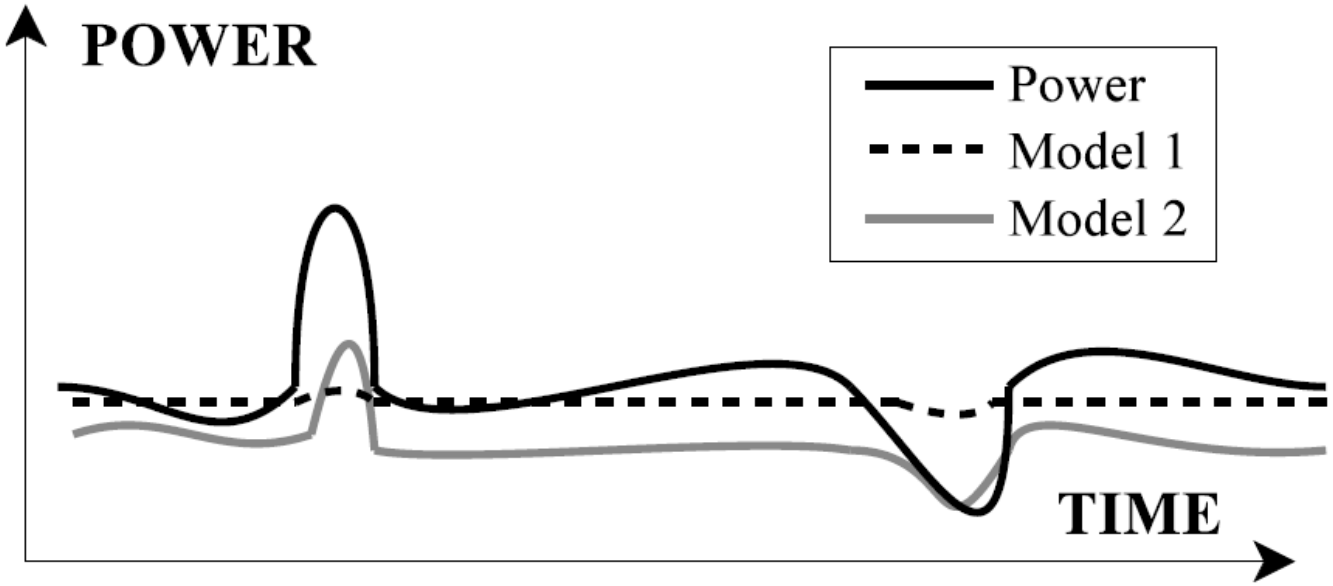
$$Model\ 1 = 547.3 \times AR_{FE} + 456.9 \times AR_{INT} + 598.2 \times AR_{FP} + 1725 \times AR_{BPU} + 982.08 \times AR_{L1} + 23677 \times AR_{L2} + 15214.5 \times AR_{MEM} + 9227$$

$$Model\ 2 = -49.1 \times AR_{FE} + 1263\ \times AR_{INT} + 2779 \times AR_{FP} + 5141 \times AR_{BPU} + 2136 \times AR_{L1} + 34305 \times AR_{L2} + 22688 \times AR_{MEM} + 7865$$

Power model examples. Model 1 and Model 2 exhibit similar average error. However, Model 1 is more acceptable/interpretable.

Background: Common modeling pitfalls

- Pitfall 2: Only validate model prediction accuracy
  - The model responsiveness, i.e. its capacity to react in a similar fashion as power consumption, is key to detect power phases



Power model examples. Model 1 and Model 2 exhibit similar average error. However, Model 2 is more responsive

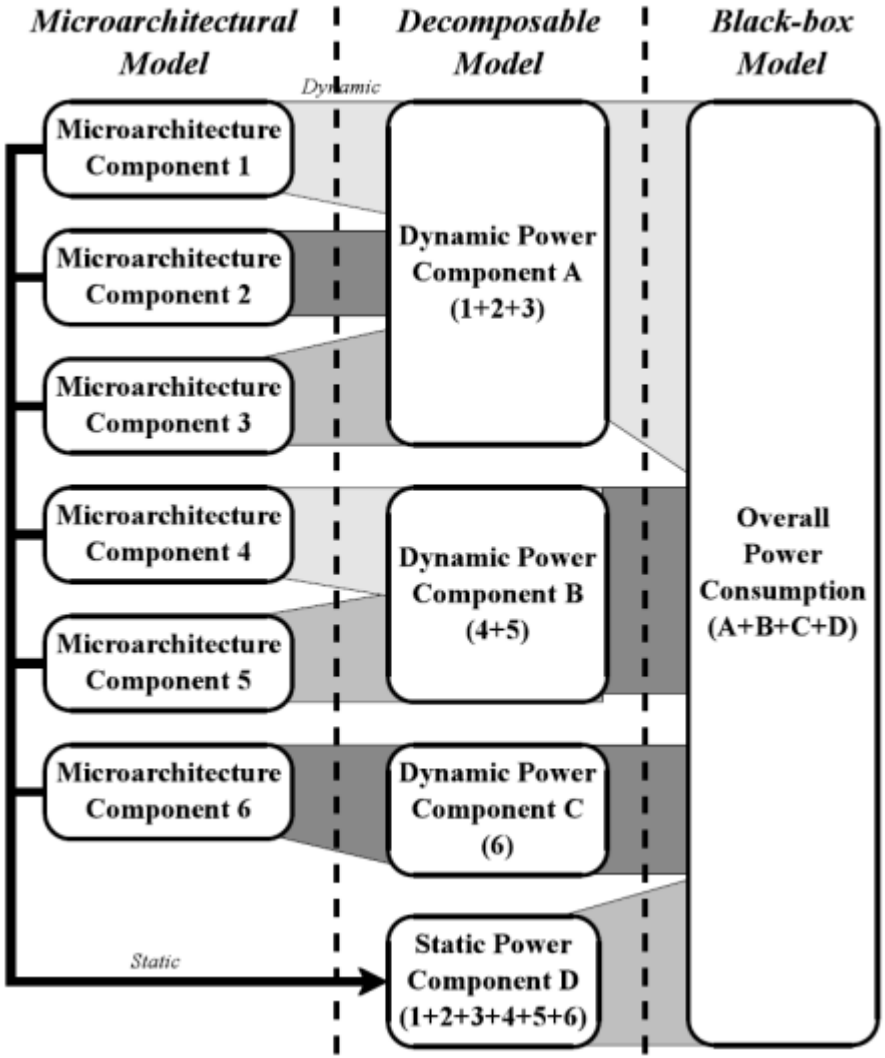
Background: Common modeling pitfalls

- Pitfall 3: Assume workload generality based on K-fold or LOOCV validation
  - Assume data from normal applications as a valid training/validation sets
    - Models biased to the training set properties
  - Lack of generality, training/validation sets do not account for all possible power situations
    - High errors on extreme/not seen situations

- Pitfall 4: Rely on human interaction to improve the model
  - Expert knowledge required to fine tune the model base on **Trial and error** experimental method
    - Time-consuming → Not affordable, not easy to deploy

Objectives

- Maximize:
  - Accuracy
  - Generality and robustness
  - Informativeness (decomposability)
  - Responsiveness
- While keeping:
  - Affordability (simple and easy to deploy)
  - Fast evaluation
- How?
  - Using a simple systematic method (affordability), based on linear regressions (simple), to generate decomposable (informativeness) counter-based (fast-evaluation) power models
  - By design (as well will show), we ensure the rest of properties: generality and robustness, accuracy and responsiveness



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DECOMPOSABLE POWER MODELS: MODELING SINGLE CORE PLATFORMS

Bottom-up modeling methodology: Introduction

- Hypothesis:
  - Power modeling methods guided using basic ***knowledge*** of the modeled system generate models that are more:
    - Accurate and responsive
    - Informative and understandable
    - Robust and general
- Assumptions (***knowledge***)
  - The system is composed of *independent power components*
    - E.g. functional units, memory hierarchy levels, …
  - The sum of the dynamic power consumption of each component in addition to the static power consumption, is the overall power consumption of the system (**Bottom-Up**)
  - The activity on each component is positively and linearly related to its dynamic power consumption
    - More activity → more power consumption
  - The static power consumption (constant) of each component is grouped into a single component (i.e. the intercept)

Bottom-up modeling methodology: Overview

- 1.- Define the system power components and their associated counters (model design/definition)
  - Maximize granularity (number of components) to improve informativeness (decomposability) → (avoid pitfall 1)
  - Use performance counters as inputs to ensure the affordability, easy to deploy and fast on-line evaluation of the generated models
  - Define a model definition algorithm to systematize the process
- 2.- Design the training set
  - Gather training data
- 3.- Derive the marginal effect of each power component to the overall power consumption
    - Use specifically designed training set (avoid pitfalls 3 and 4)
    - Define an algorithm to systematize the process
- 4.- Validate the model

Bottom-up modeling methodology: Overview

- 1.- Define the system power components and their associated counters (model design/definition)
  - Maximize granularity (number of components) to improve informativeness (decomposability) → (avoid pitfall 1)
  - Use performance counters as inputs to ensure the affordability, easy to deploy and fast on-line evaluation of the generated models
  - Define a model definition algorithm to systematize the process
- 2.- Design the training set
  - Gather training data
- 3.- Derive the marginal effect of each power component to the overall power consumption
    - Use specifically designed training set (avoid pitfalls 3 and 4)
    - Define an algorithm to systematize the process
- 4.- Validate the model

Bottom-up modeling methodology: Power component definition - Overview

- What is a ***power component***?
  - A power component represents the power consumption of a part of the modeled system
  - A power component has an associated activity ratio (AR) formula based on performance counters
    - Usually, #events / cycle

- Objective: Systematize power component definition process
  - Maximize the number of power components produce more informative power model
    - Ideally: 1 architecture component ←→ 1 power component
    - Reality: N architecture components ←→ 1 power component
      - Why? Some properties should be fullfilled
  - Define the set of rules that define the power components

Bottom-up modeling methodology: Power component definition - Rules

Constraint 1: Limit availability of performance counters

- Microarchitectural components with not direct performance counters accounting for their activity should be grouped with the most related microarchitectural components with performance counters available.

Constraint 2: Impossibility to decouple the activities of different components

- Power components which activity can not be decoupled from other components should be grouped together.

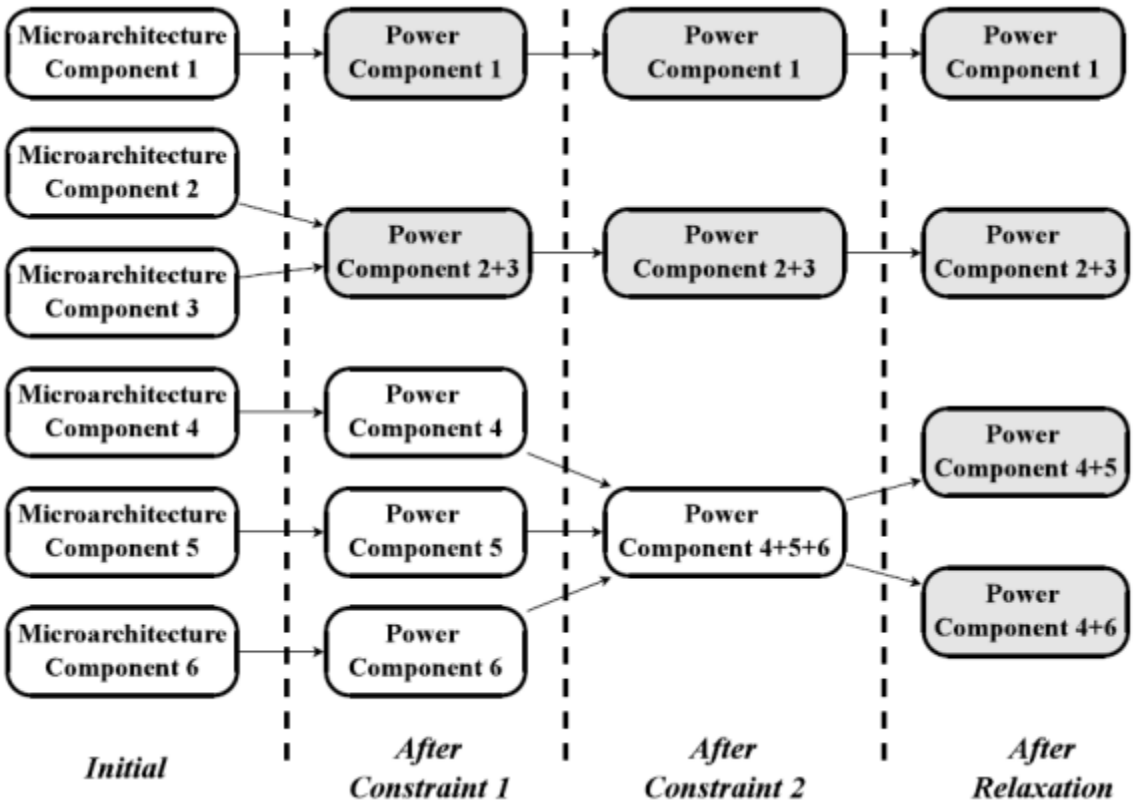
Relaxation 1: Lack of granularity

- Power components defined after the application of Constraint 2 can be split if the activities of the new power components can be decoupled and the activity of the power components causing the coupling is accounted in the activity ratio formula of the each of the new power components

- The activity ratio formula of the new power components defined should be updated to account for the activity (directly or indirectly) of all the microarchitectural components within the power component.

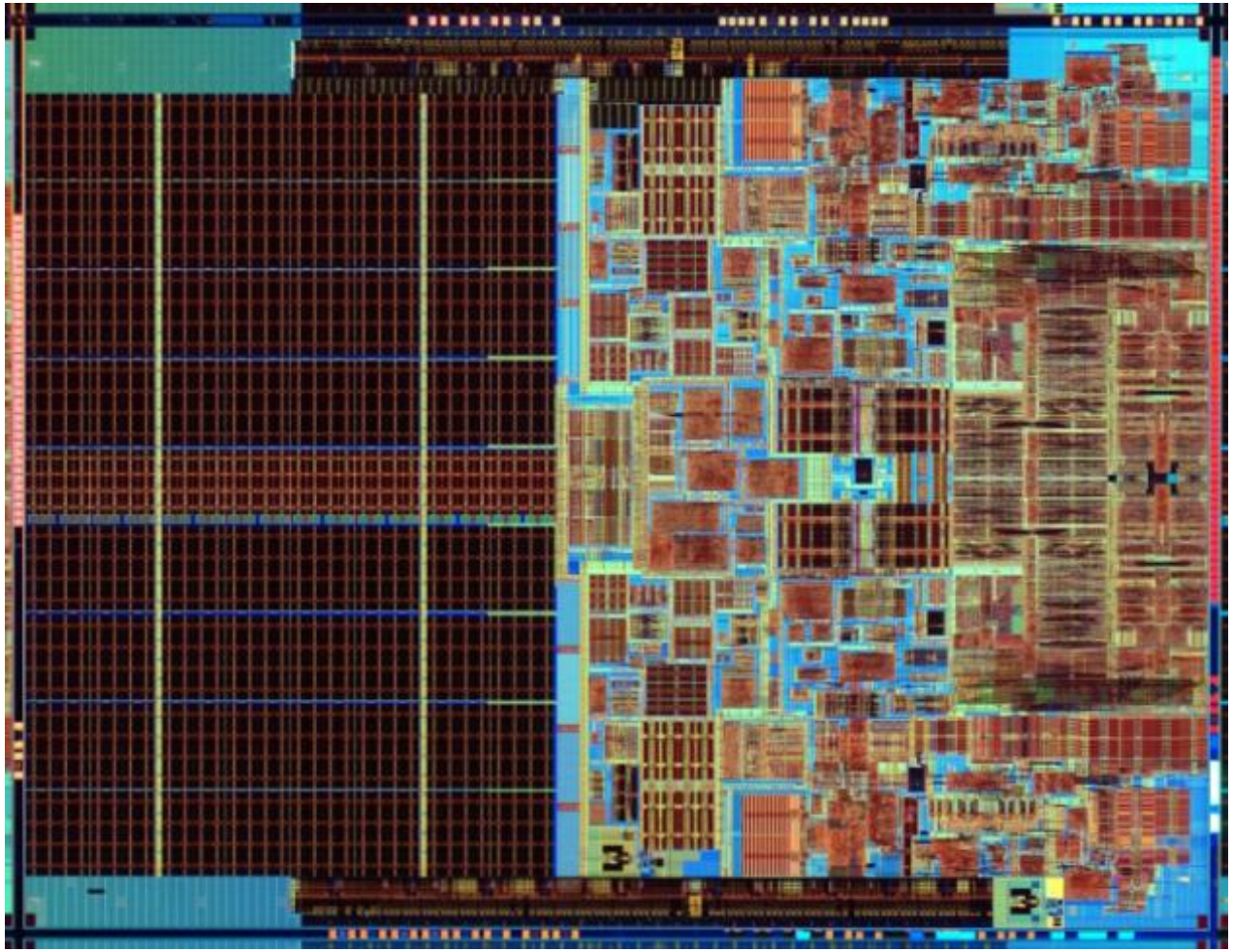
Bottom-up modeling methodology: Power component definition - Algorithm

1) Define a power component for each microarchitecture component
2) Apply Constraint 1: join component without counters
3) Apply Constraint 2: join component that can not be decoupled
4) Apply Relaxation 1: split components



Bottom-up modeling methodology: Power component definition – Intel Core 2

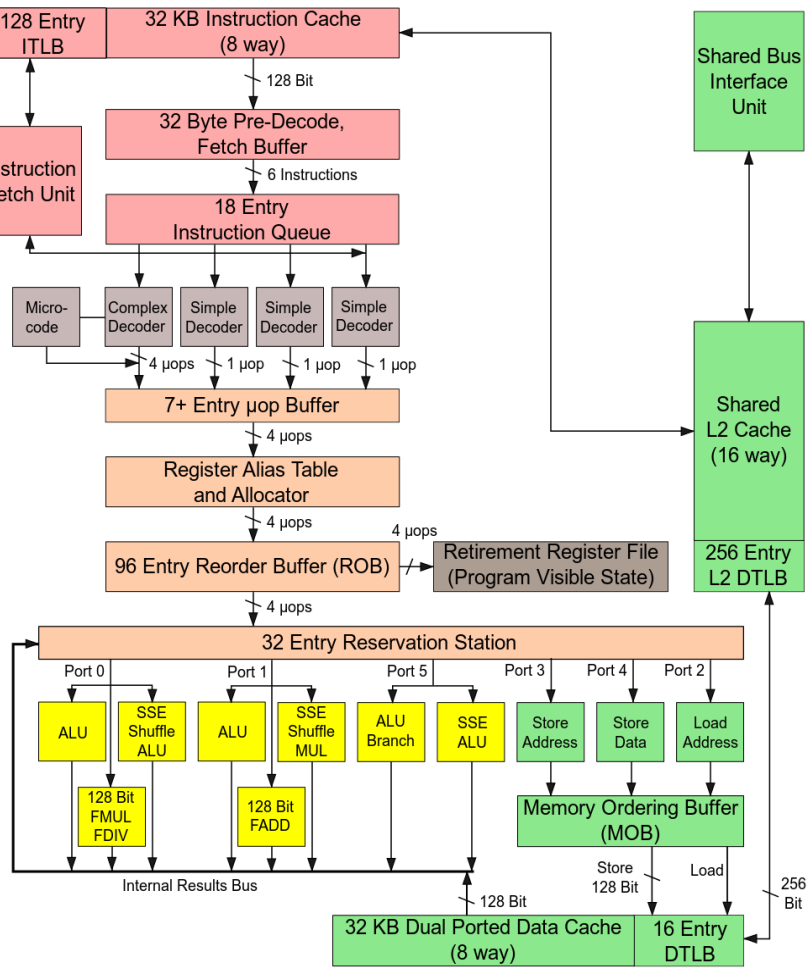
Intel Core 2 processor floorplan



Bottom-up modeling methodology: Power component definition – Intel Core 2

- > 30 microarchitecture components
- In-order pipeline:
  - I-Cache, ITLB, IFU, Pre-Decode, IQ, Microcode ROM, Decoders, uOP buffer, RAT, ROB, BPU
- Out-of-Order pipeline:
  - ALUs, SSEs, FMUL, FDIV, FADD
- Memory hierarchy
  - AGUs, MOB, L1, L1-DTLB, L2, L2-DTLB, FSB/MEM

Intel Core 2 pipeline



Intel Core 2 Architecture

Bottom-up modeling methodology: Power component definition – Intel Core 2

- Power components from the in-order pipeline:
  - BPU:
    - The Branch prediction unit activity can be decoupled from the rest and have counters accounting for their activity (# Branches instructions decoded)
      - AR formula: BR_INST_DECODED/CPU_CLK_UNHALTED
  - FRONTEND (FE):
    - Includes the rest of the microarchitecture components because:
      - Activities can not be decoupled
        - » Activity in Stage N ~ Activity in Stage N+1
      - Most components do not have performance counters accounting directly for their activity
    - AR formula: UOPS_RETIRED:ANY/CPU_CLK_UNHALTED

Bottom-up modeling methodology: Power component definition – Intel Core 2

- Power components from the out-of-order pipeline:
  - FP:
    - Includes all the floating point units because:
      - There is only a generic counter (FP_COMP_OPS_EXE) accounting for the FP operation executed (there is not a counter for each unit)
      - Moreover, most the FP instructions can go to different FP units and hence, it is impossible control/decouple their activities.
    - AR formula: FP_COMP_OPS_EXE/CPU_CLK_UNHALTED
  - SIMD:
    - Includes all the SIMD units because:
      - There is only a generic counter (SIMD_UOPS_EXE) accounting for the SIMD operation executed (there is not a counter for each unit)
      - Moreover, most the SIMD instructions can go to different SIMD units and hence, it is impossible control/decouple their activities.
    - AR formula: SIMD_UOPS_EXE/CPU_CLK_UNHALTED
  - INT:
    - Include all the integer units because:
      - Most of the integer instructions can go to different integer units, hence it is impossible to control/decouple their activities.
    - Integer units do no have counters accounting for their direct activity. However their activity can be derived from ALL activity minus the FP, SIMD and Branch activity.
    - AR formula: (RS_UOPS_DISPATCHED_CYCLES:PORT_0 + RS_UOPS_DISPATCHED_CYCLES:PORT_1 + RS_UOPS_DISPATCHED_CYCLES:PORT_5 – FP_COMP_OPS_EXE – SIMD_UOPS_EXEC – BR_INST_RETIRED:ANY)/CPU_CLK_UNHALTED

Bottom-up modeling methodology: Power component definition – Intel Core 2

- Power components from the cache hierarchy:
  - L1:
    - Includes LD/ST execution units, MOB, L1 cache, L1 DTLB, L2 DTLB
      - Some units without counters accounting for their activity
      - it is impossible control/decouple their activities
    - AR formula: L1D ALL REF/ CPU CLK UNHALTED
  - L2:
    - Includes the L2 cache
      - Although L2 activity implies L1 activity, the contribution of the L2 can be derived incrementally after knowing the contribution of the L1 component.
    - AR formula: L2 RQSTS/ CPU CLK UNHALTED
  - Main memory
    - Includes then: FSB (Front Side Bus) and main memory
      - Although FSB/main memory activity implies L1 activity, the contribution of the main memory component can be derived incrementally after knowing the contribution of the L1/L2 components.
    - AR formula: BUS DRDY CLOCKS/ CPU CLK UNHALTED

Bottom-up modeling methodology: Overview

- 1.- Define the system power components and their associated counters (model design/definition)
  - Maximize granularity (number of components) to improve informativeness (decomposability) → (avoid pitfall 1)
  - Use performance counters as inputs to ensure the affordability, easy to deploy and fast on-line evaluation of the generated models
  - Define a model definition algorithm to systematize the process
- 2.- Design the training set
  - Gather training data
- 3.- Derive the marginal effect of each power component to the overall power consumption
    - Use specifically designed training set (avoid pitfalls 3 and 4)
    - Define an algorithm to systematize the process
- 4.- Validate the model

Bottom-up modeling methodology: Design of the training set for training the model

- The rule of thumb:

  "the broader the type of situations used to train the model, the more general and accurate the model will be"

  - This implies:
    - Generate micro-benchmarks stressing different combinations of the power components defined
      - Stress only one unit or various
    - Cover all the range of possible activities
      - E.g. stress the floating point unit from IPC 0.05 to IPC 4 (if possible)
- To ensure the decomposability:
  - Generate micro-benchmarks decoupling the activity between the component
    - Minimize the colinearity between component activities (inputs of the model)

Bottom-up modeling methodology: Training set: Intel Core 2

Microbench-mark set	#	FE Activity	INT Activity	FP Activity	SIMD Activity	BPU Activity	L1 Activity	L2 Activity	FSB Activity
FE	1	1	0	0	0	0	0	0	0
INT	13	1-3.45	1-3	0	0	0	0	0	0
FP	9	0.2-1.98	0	0.2-1	0	0	0	0	0
SIMD	12	1.85-3.29	0	0	0.99-2.63	0	0	0	0
BPU	5	0.42-1.14	0	0	0	0.46-1	0	0	0
L1	16	1-2.97	0	0	0	0	0.66-2	0	0
L2	12	0.12-0.42	0	0	0	0	0.11-0.22	0.11-0.21	0
MEM	18	0.02-0.14	0	0	0	0	0.02-0.04	0.02-0.04	0.58-0.71
RANDOM	11	1.63-3.95	0-1	0-0.8	0-1.97	0-0.34	0-1.97	0-0.07	0-0.34
TOTAL	97	0.02-3.95	0-3	0-1	0-2.63	0-1	0-2	0-0.21	0-0.71

~100 micro-benchmarks stressing the different power components defined at different activity ratio

Bottom-up modeling methodology: Overview

- 1.- Define the system power components and their associated counters (model design/definition)
  - Maximize granularity (number of components) to improve informativeness (decomposability) → (avoid pitfall 1)
  - Use performance counters as inputs to ensure the affordability, easy to deploy and fast on-line evaluation of the generated models
  - Define a model definition algorithm to systematize the process
- 2.- Design the training set
  - Gather training data
- 3.- Derive the marginal effect of each power component to the overall power consumption
  - Use specifically designed training set (avoid pitfalls 3 and 4)
  - Define an algorithm to systematize the process
- 4.- Validate the model

Bottom-up modeling methodology: Modeling the power components

- The overall power is the addition of the power consumption of each power component defined

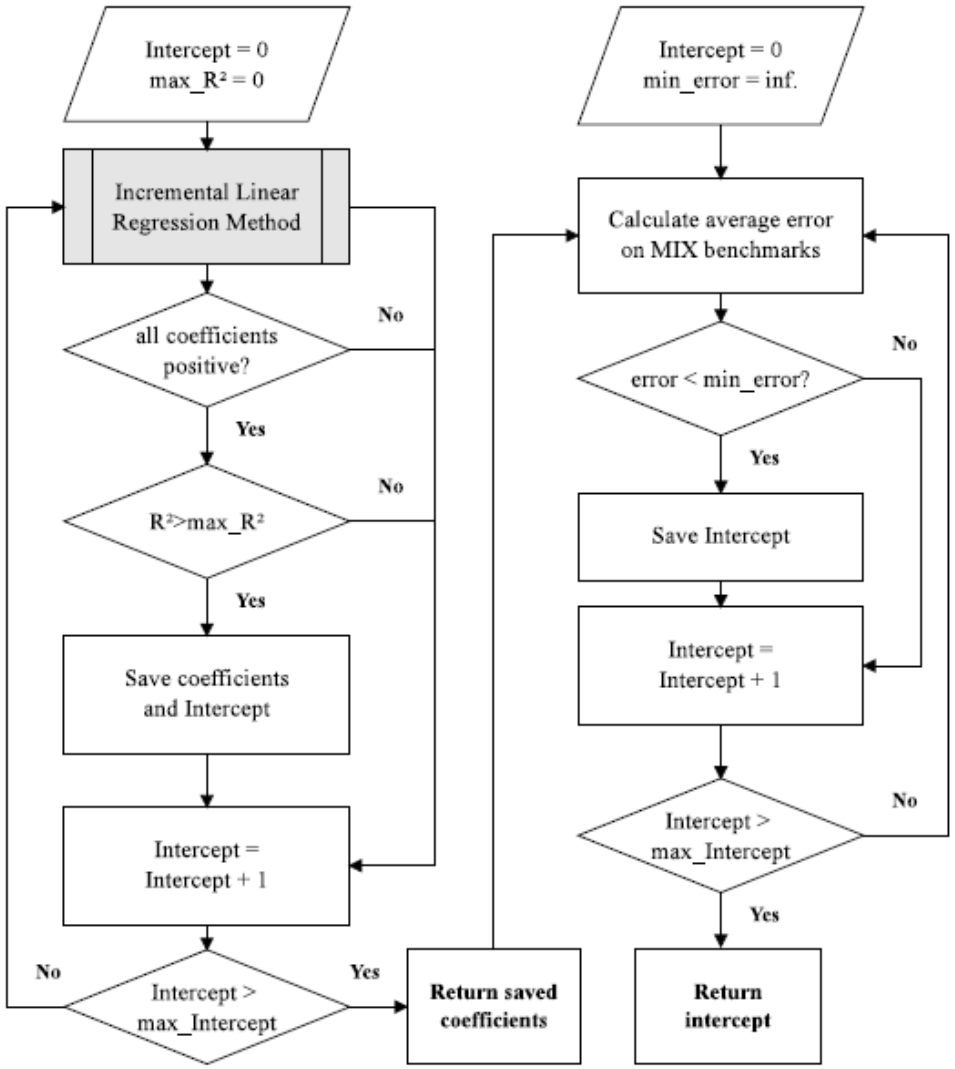
$$Power = \left(\sum_{i=0}^{n} AR_i \times P_i\right) + P_{Static}$$

- Where:
  - n is the numbers of components defined
  - AR_i is the activity ratio of the component i
  - P_i is the power weight of the component i
    - The power weights should be positive
  - P_{Static} is the static power consumption

- Approach: model each Power weight separately
  - use the specifically designed training set
  - Based on linear regression

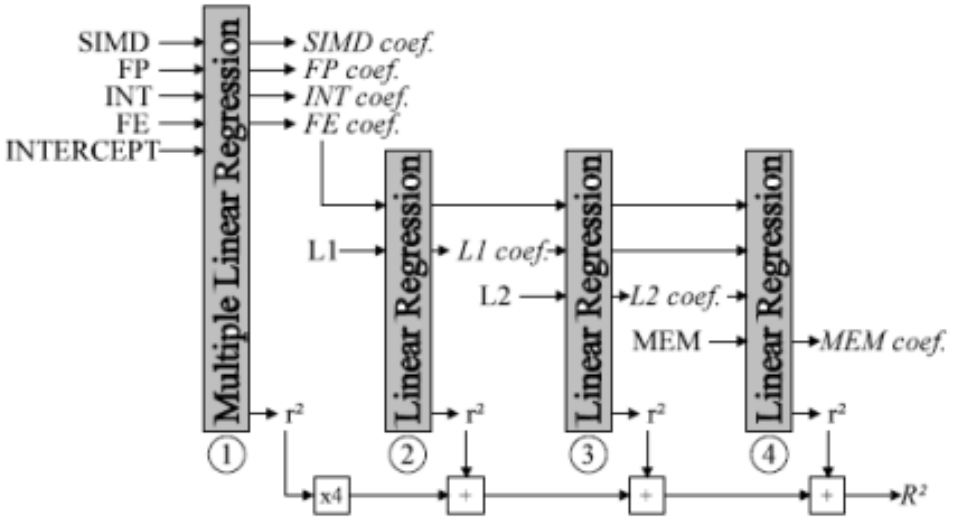
Bottom-up modeling methodology: Modeling the power components

- 1st step: model the weights of the power components
  - Apply incremental linear regression method (next slide)
  - Check all weights positive
  - Maximize correlation coefficient
- 2nd step: tune the P_{Static} component
  - Use the random micro-benchmark set
  - Avoid sub-estimating P_{Static} due to energy saving techniques
    - E.g. clock-gating
- The method requires specifically designed training data to find a solution
- The method does not require human intervention
  - Systematic



Bottom-up modeling methodology: Modeling the power components

- Apply incremental linear regression method:
  - Apply a sequential number of linear regression (one for each component defined) using the intercept provided
    - Force intercept to be zero
    - Model of component i+1 is trained using the residuals of applying the previous (0..i) models to the micro-benchmark set stress that component
  - Return the sum of the correlation coefficient of each linear regression and the weights assigned to each component
- Intel Core 2 example:



Bottom-up modeling methodology: Overview

- 1.- Define the system power components and their associated counters (model design/definition)
  - Maximize granularity (number of components) to improve informativeness (decomposability) → (avoid pitfall 1)
  - Use performance counters as inputs to ensure the affordability, easy to deploy and fast on-line evaluation of the generated models
  - Define a model definition algorithm to systematize the process
- 2.- Design the training set
  - Gather training data
- 3.- Derive the marginal effect of each power component to the overall power consumption
    - Use specifically designed training set (avoid pitfalls 3 and 4)
    - Define an algorithm to systematize the process
- 4.- Validate the model

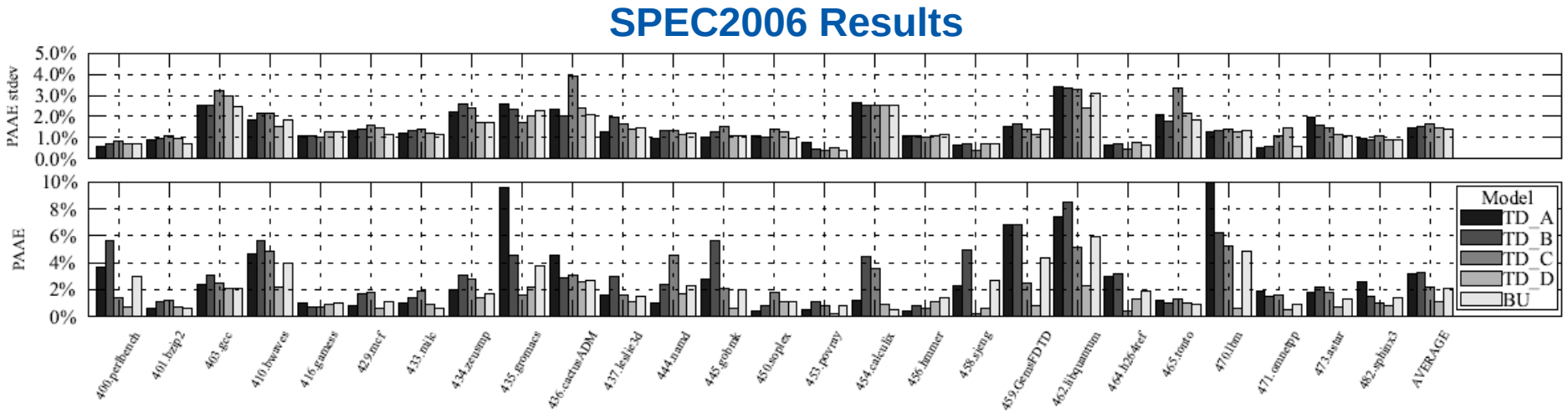
Bottom-up modeling methodology: Validation

- Metrics to validate:
  - Accuracy :
    - Difference between power estimations and real measurements
      - PAAE: percentage absolute average error
  - Responsiveness: Capacity to detect phases
    - Apply the same phases detection algorithm to estimations and the real measurement and compare the results
      - %Accuracy → check if the mode is able to detect phases
        - » ((# of phases correctly predicted)/(total # of phases))*100
      - %False positives → check that the model do not over-react
        - » ((# of non-existent phases predicted)/(total # of phases))*100
  - Robustness (generality, workload independent):
    - Apply the generate model on a wide set of application types to check its generality
      - CPU workloads: SPEC2006
      - MEM workloads: NAS Parallel Benchmarks
      - OS System : LMBENCH Suite

Bottom-up modeling methodology: Validation: Intel Core 2

- Top-down (black-box) models generated for comparison purposes:
  - TD_A: Simplest Top-down model.
    - Linear regression: P = f(IPC)
    - Training set: SPEC2006
  - TD_B: Simple Top-down model.
    - Linear regression: P = f(IPC,MEM)
    - Training set: SPEC2006
  - TD_C: Complex Top-down model.
    - Use of parameter selection techniques
    - Linear regression: P = f(IPC,FP,MEM,STALLS)
    - Training set: SPEC2006
  - TD_D: Optimal Top-down model.
    - Use the same inputs as the BU model
    - Linear regression: P = f(FE,INT,FP,SIMD,BPU,L1,L2,MEM)
    - Training set: SPEC2006

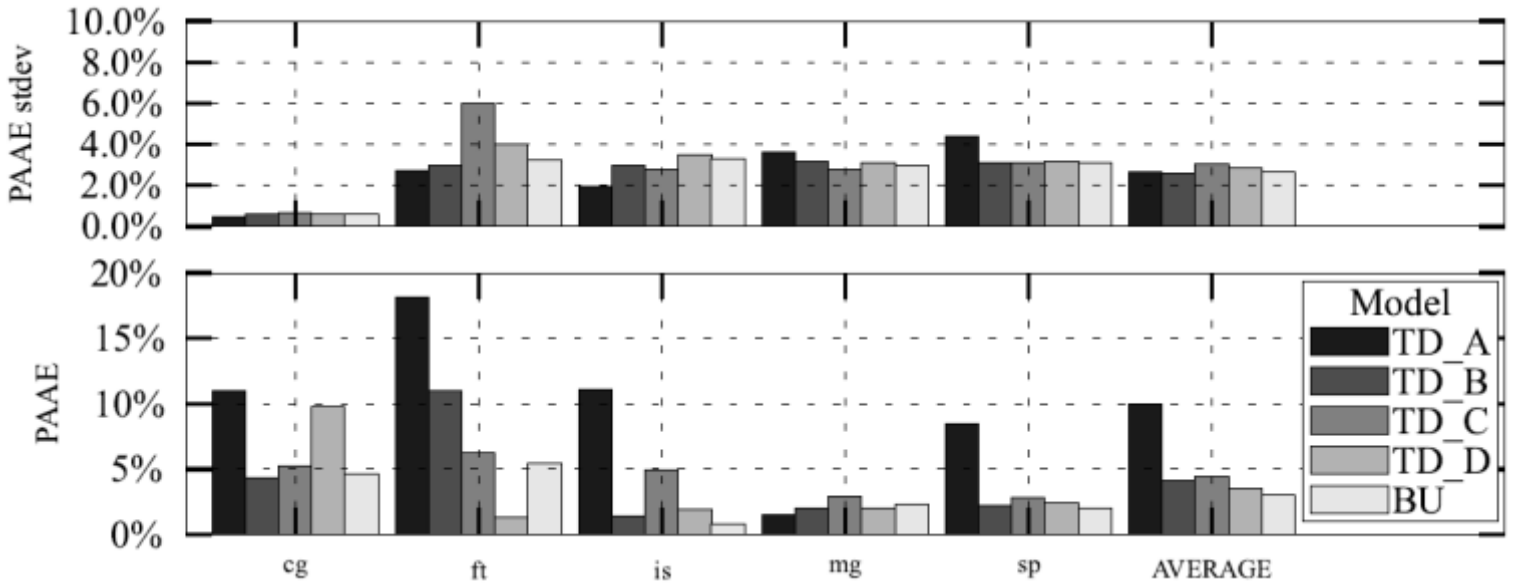
Bottom-up modeling methodology: Validation: Intel Core 2 - Accuracy



- All models show similar average results on average
  - All models show low average errors
    - The error of simplest TD_A model is high in some cases
  - In general, even simpler approaches perform well
  - TD_D outperforms the other models
- **BU model shows similar results even it was not trained using the SPEC2006 data as the training set**
  - All the other models are over-trained for this validation suite

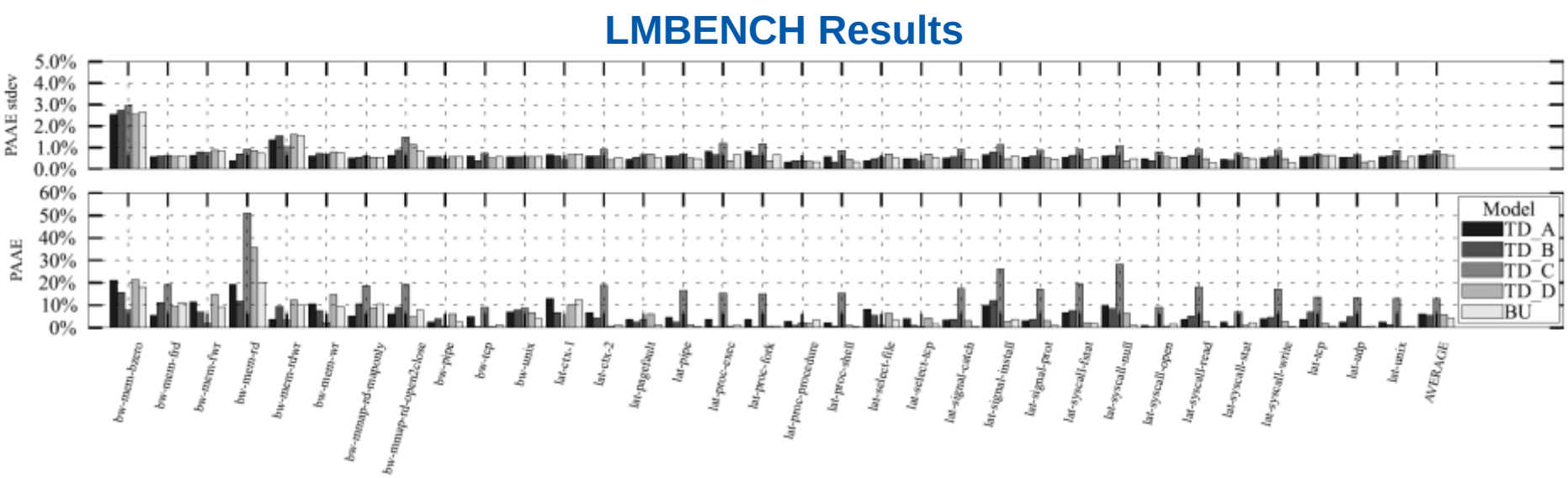
Bottom-up modeling methodology: Validation: Intel Core 2 - Accuracy

NAS Parallel Benchmarks Results



- Benchmarks configured to stress the main memory system
- All models including the memory component show similar average results on average
  - All models show low average errors (<5%)
- TD_A, not including the memory component, show high errors
  - **This remarks the importance of modeling all the components of the architecture**
- **BU model slightly outperforms the rest of models**
  - None of the models is over-trained for this validation suite

Bottom-up modeling methodology: Validation: Intel Core 2 - Accuracy



- Benchmarks stressing different OS/System characteristics
  - Memory bandwidth, context switch, page faults, system calls, signals, …
  - Different behavior than normal applications
- All models show reasonable average error results (expect TD_C)
  - All models show low average errors (<5%),
  - Some outliers (bandwidth test kernels → low processor activity → high error)
- TD_C model shows much higher errors than the simpler TD_A and TD_B
  - Parameter selection techniques fail → tailor models to specific training set characteristics
  - **This remarks the importance of selecting model inputs based on the components of the architecture**
- **BU model slightly outperforms the rest of models**
  - None of the models is over-trained for this validation suite

Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

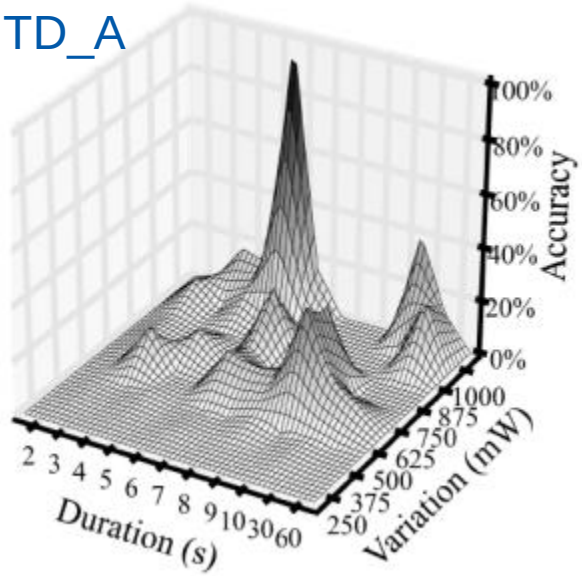
- Apply the same phase detection algorithm the predicted power and the real power
  - Algorithm: First Pivot Clustering:
    - A new phase is defined if the power values is above/below a given threshold
  - Phases classified by:
    - Duration: how long they last
    - Variation: how big was the power variation with respect to the previous phases

- Intel Core 2:
  - FPC algorithm applied on the SPEC 2006 traces
    - Fine-grain threshold: 250mW

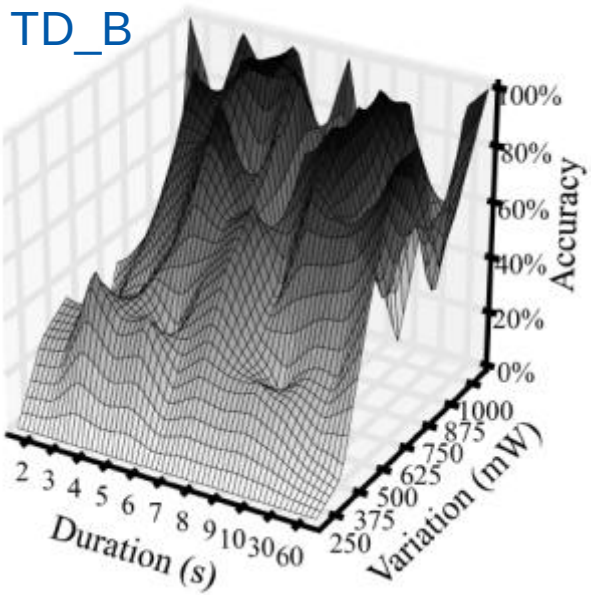
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

Responsiveness Accuracy

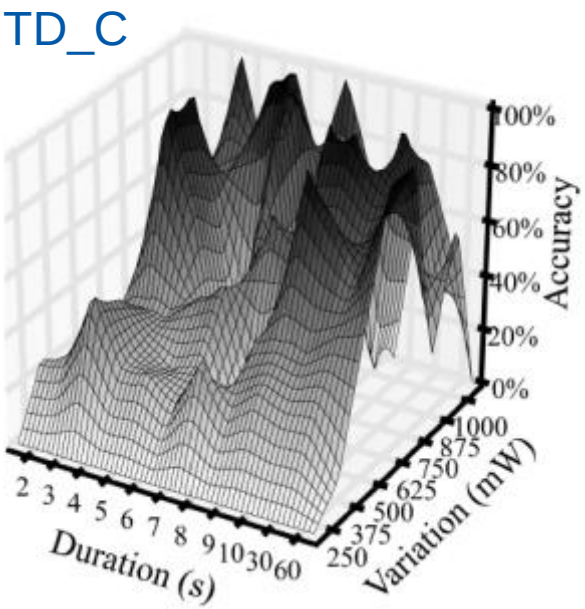
TD_A



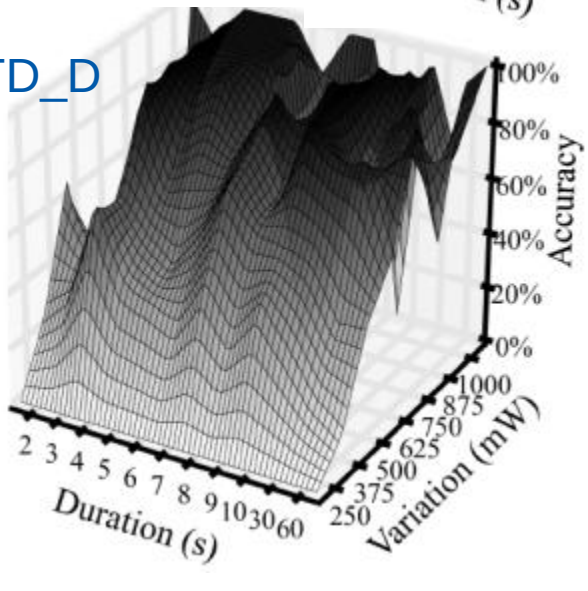
TD_B



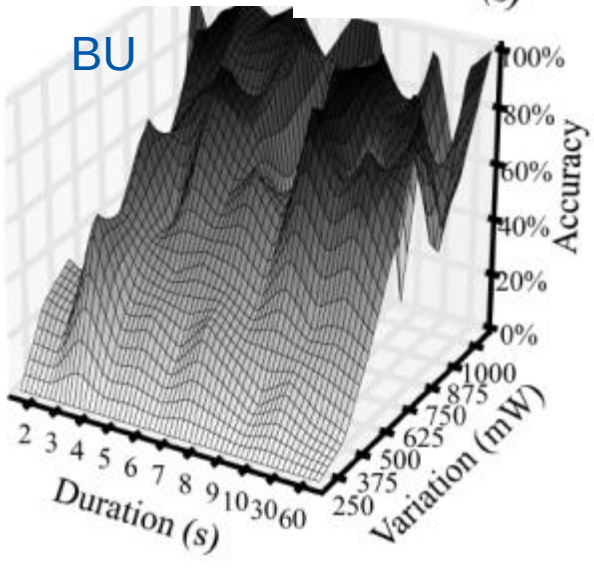
TD_C



TD_D



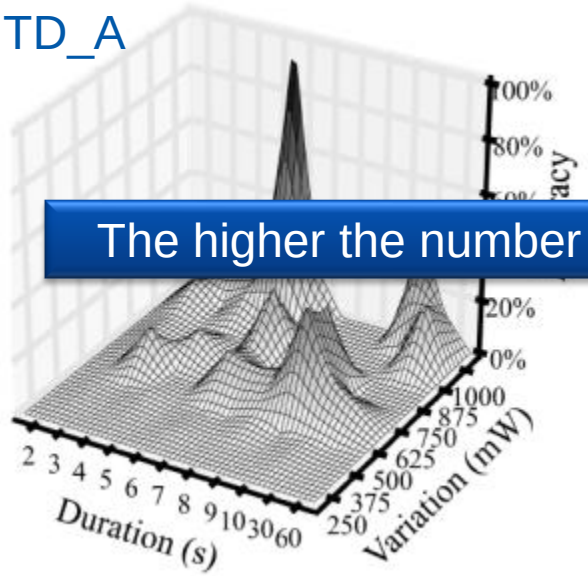
BU



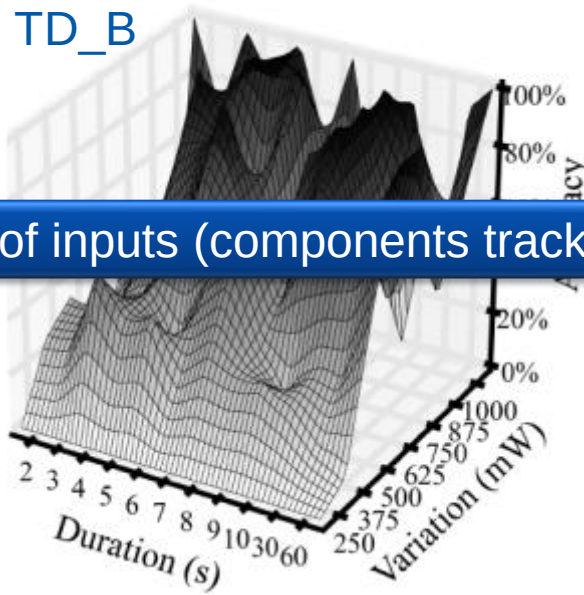
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

Responsiveness Accuracy

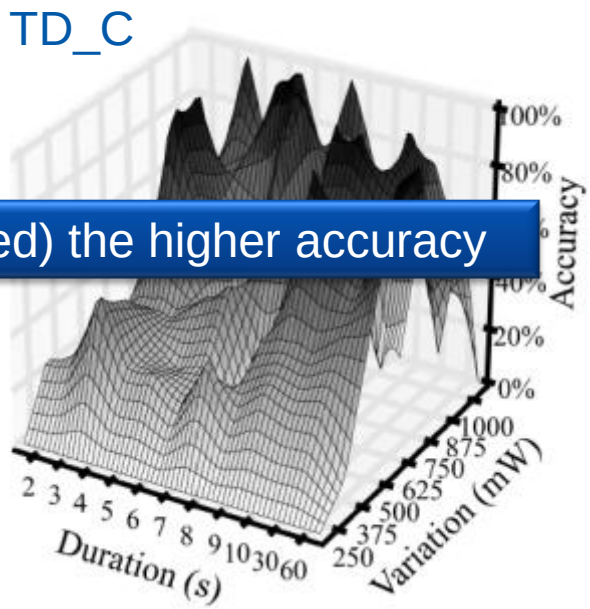
TD_A



TD_B

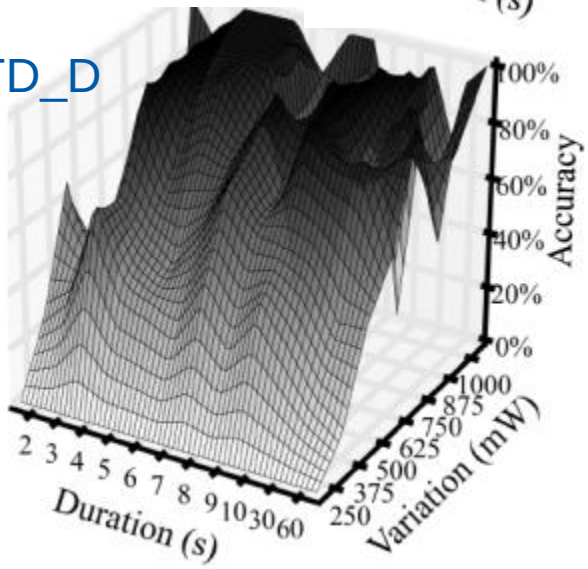


TD_C

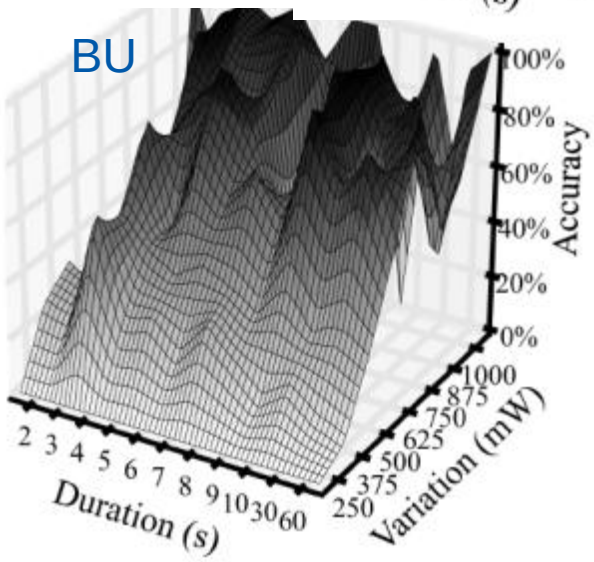


The higher the number of inputs (components tracked) the higher accuracy

TD_D



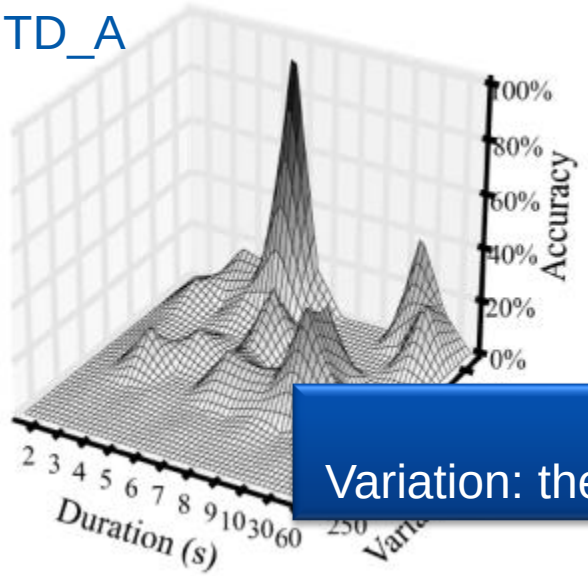
BU



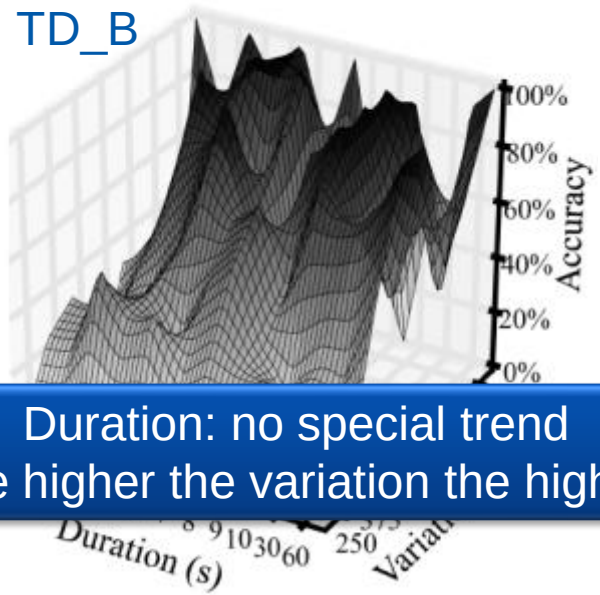
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

Responsiveness Accuracy

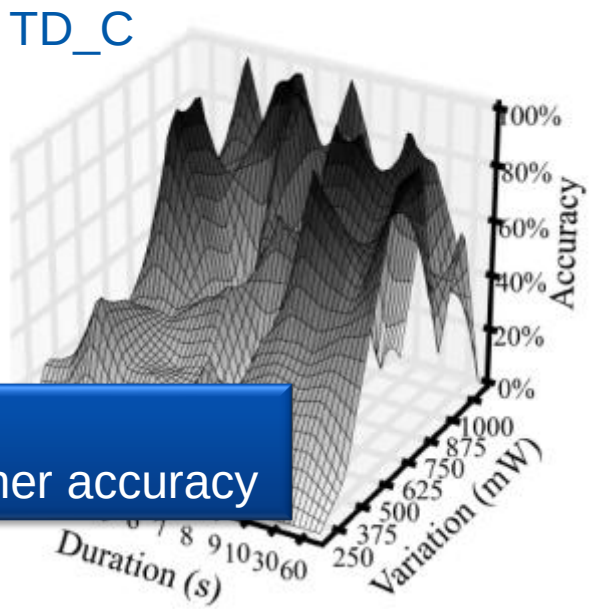
TD_A



TD_B

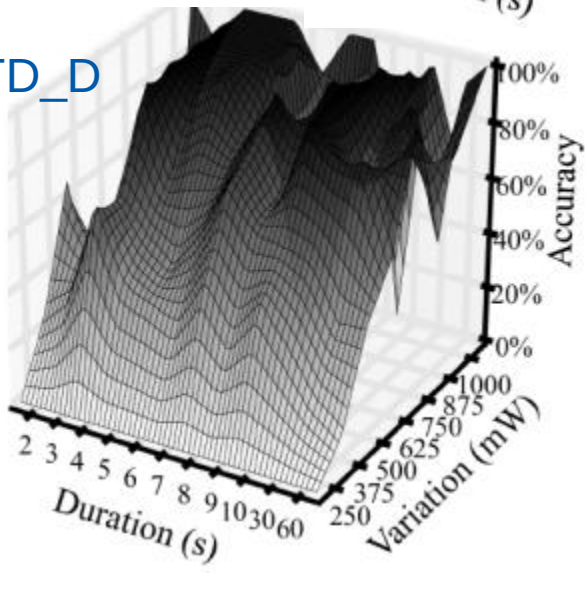


TD_C

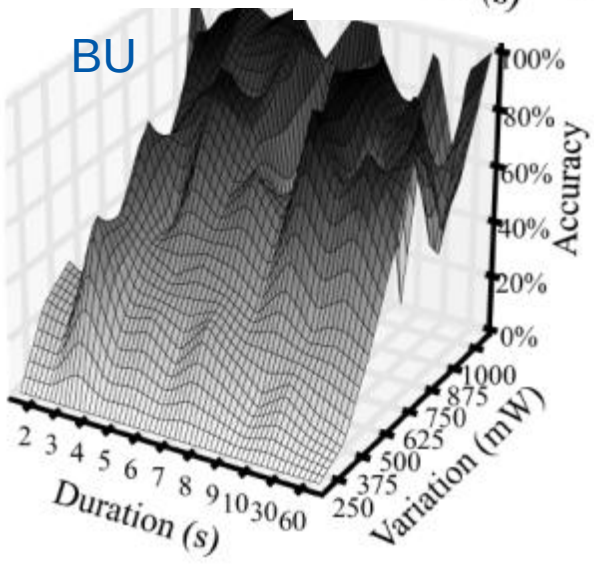


Duration: no special trend
Variation: the higher the variation the higher accuracy

TD_D



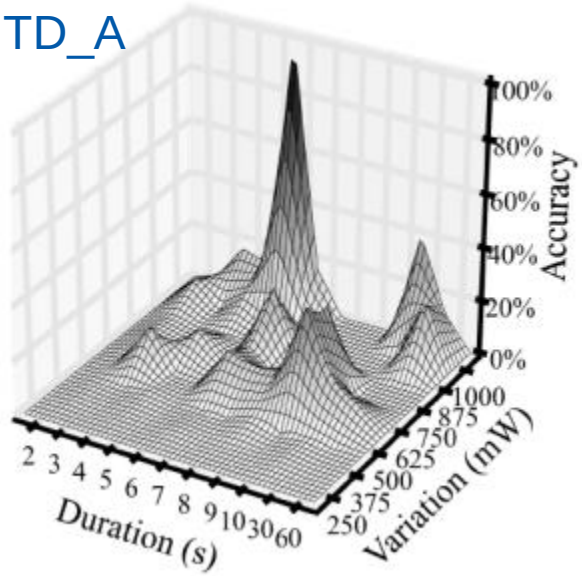
BU



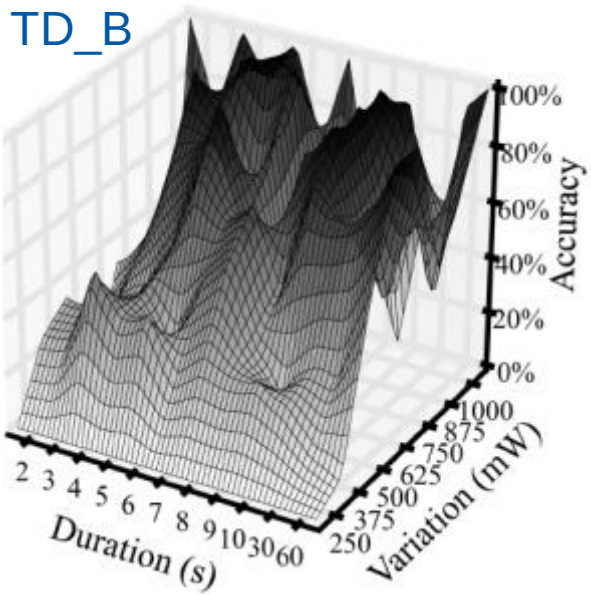
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

Responsiveness Accuracy

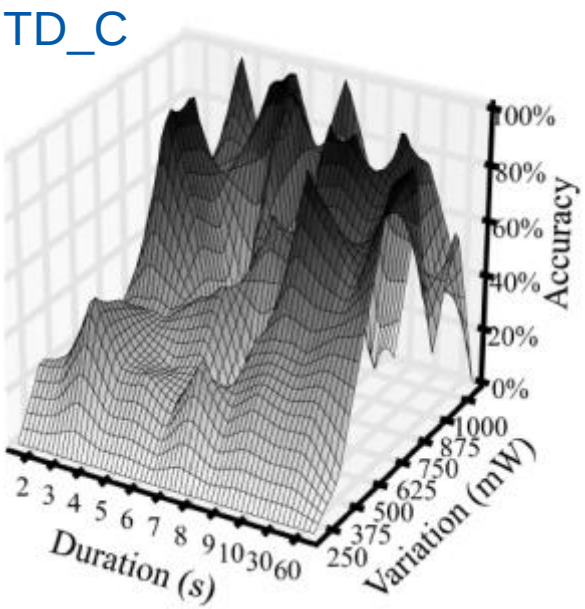
TD_A



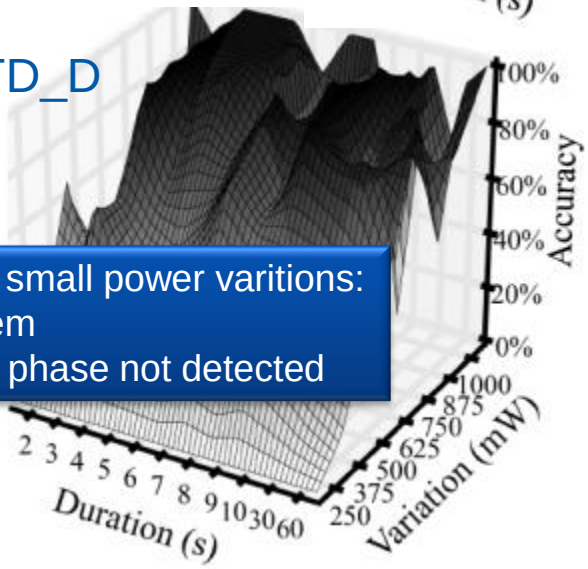
TD_B



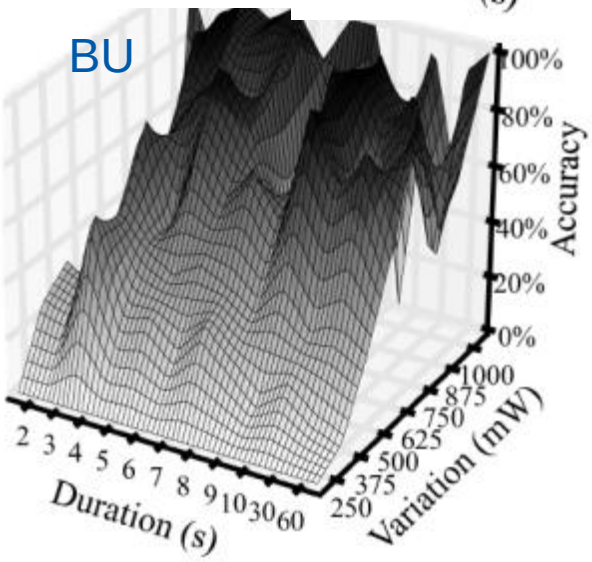
TD_C



TD_D



BU

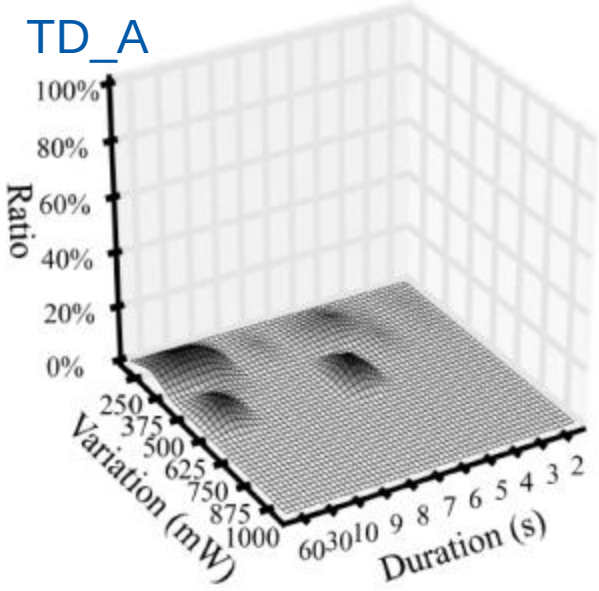


Low accuracy for small power varitions:
- Several of them
- small error → phase not detected

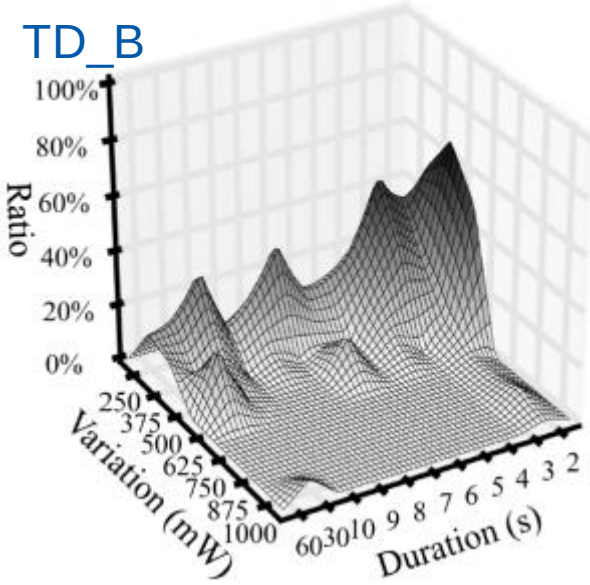
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

- False positives

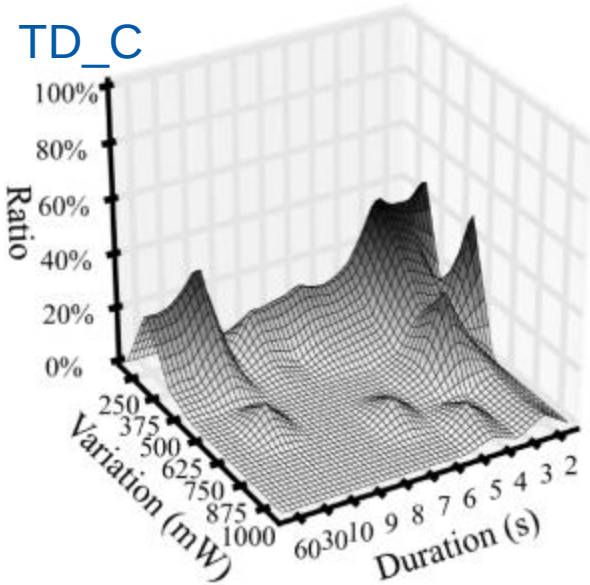
TD_A



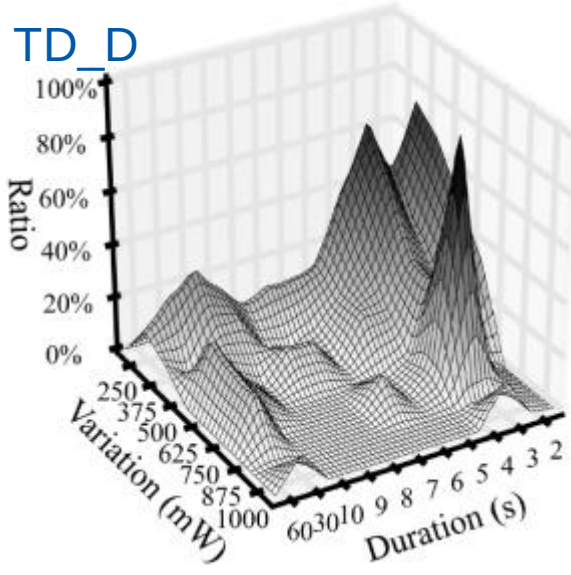
TD_B



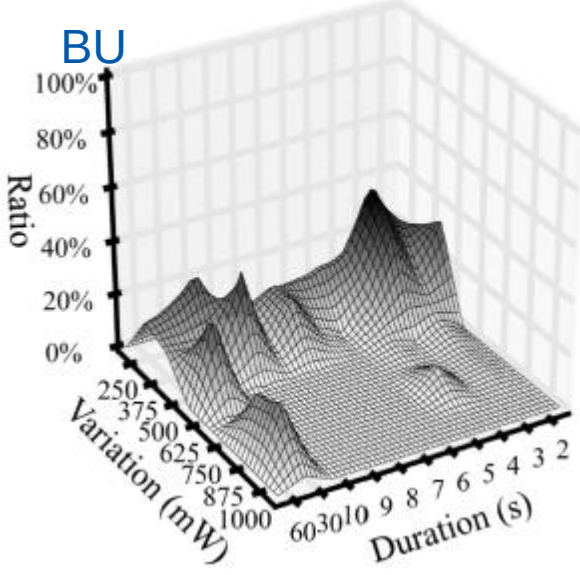
TD_C



TD_D



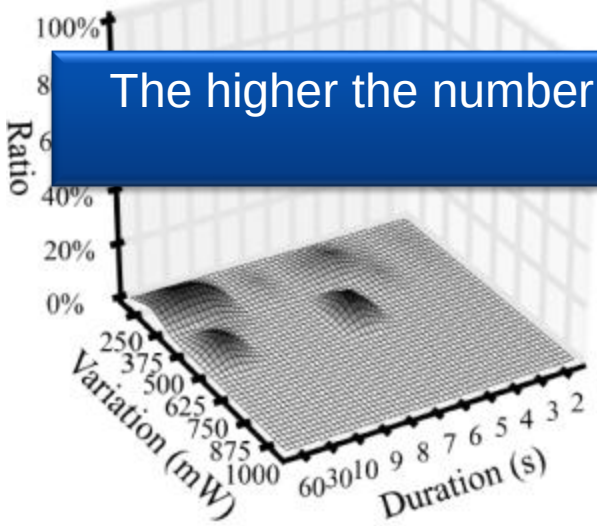
BU



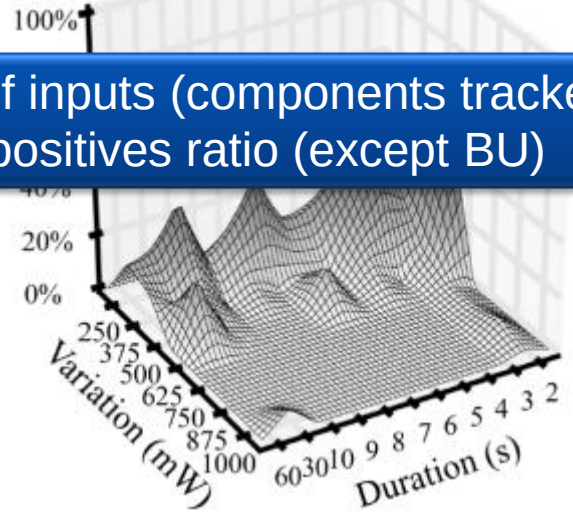
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

- False positives

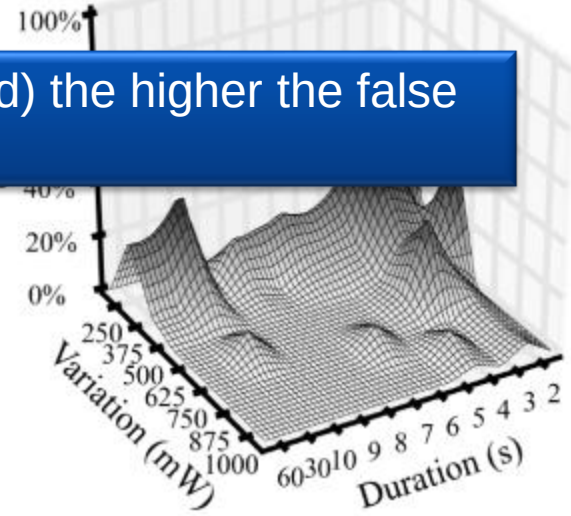
TD_A



TD_B

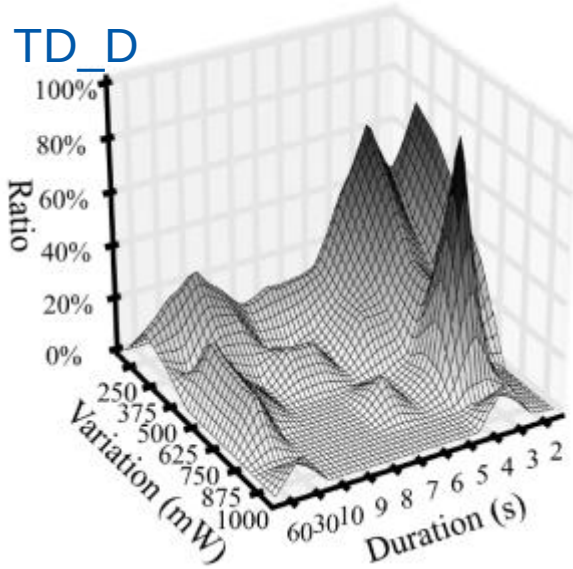


TD_C

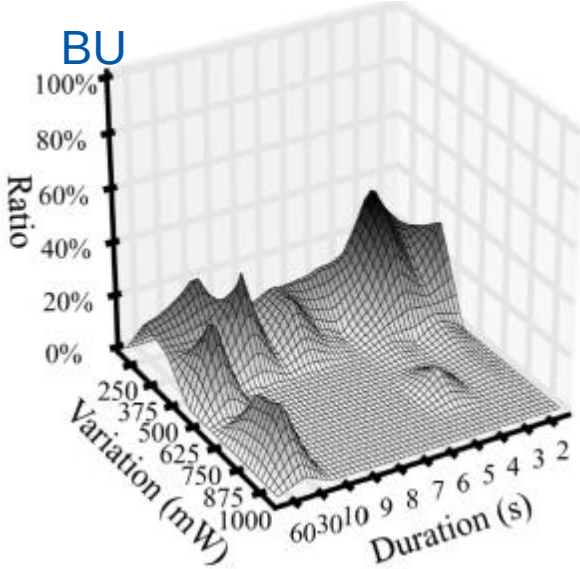


The higher the number of inputs (components tracked) the higher the false positives ratio (except BU)

TD_D



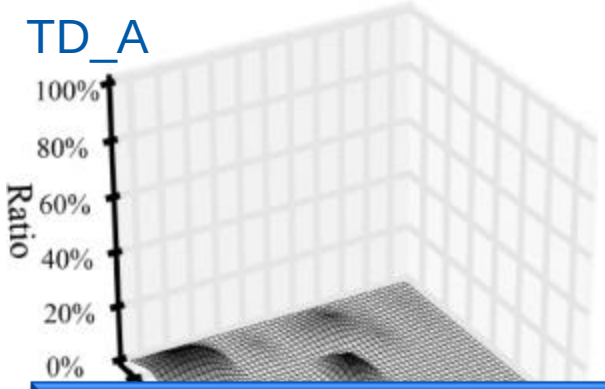
BU



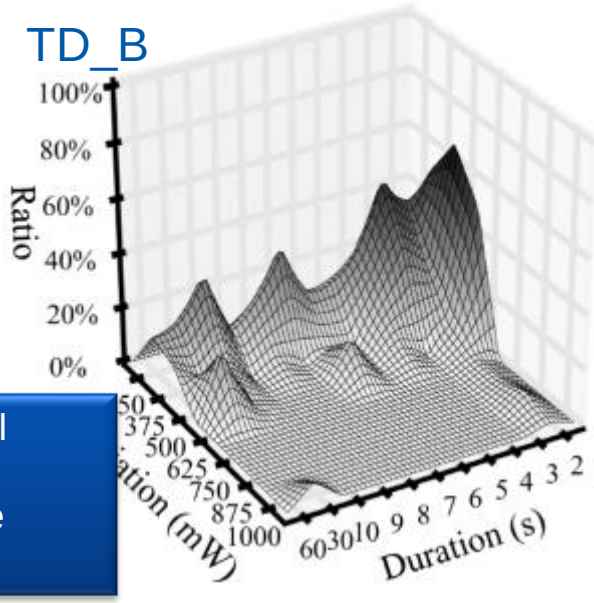
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

False positives

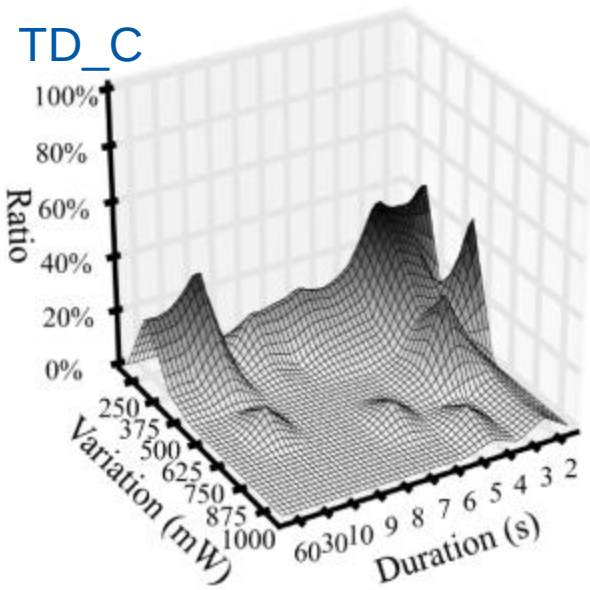
TD_A



TD_B



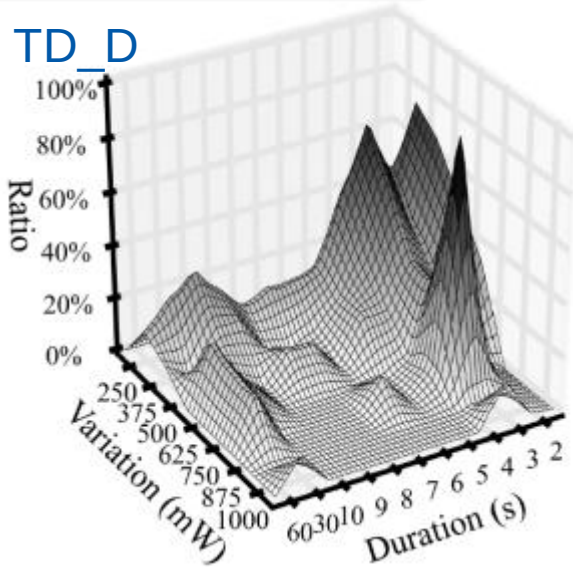
TD_C



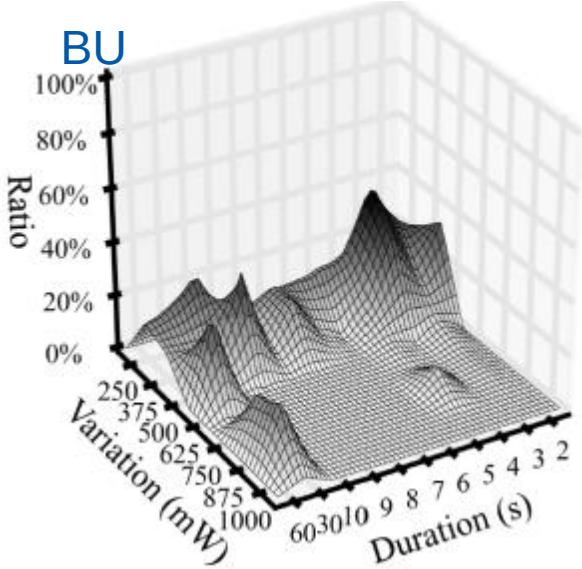
High false positives ratio for small varitations:

- small error → inexistent phase detected

TD_D



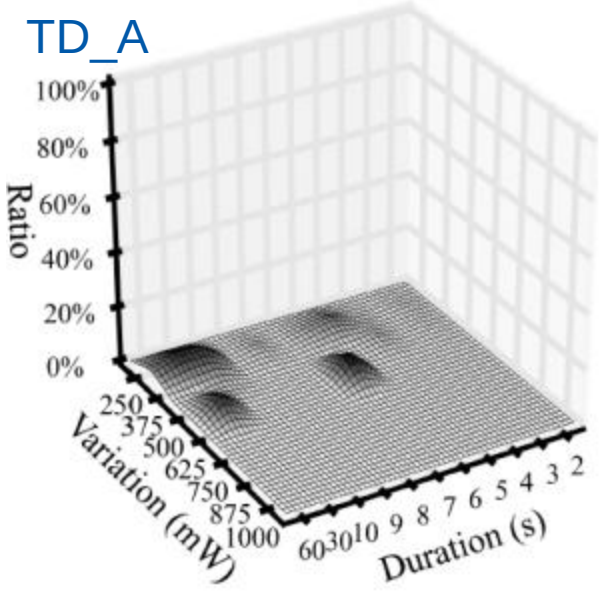
BU



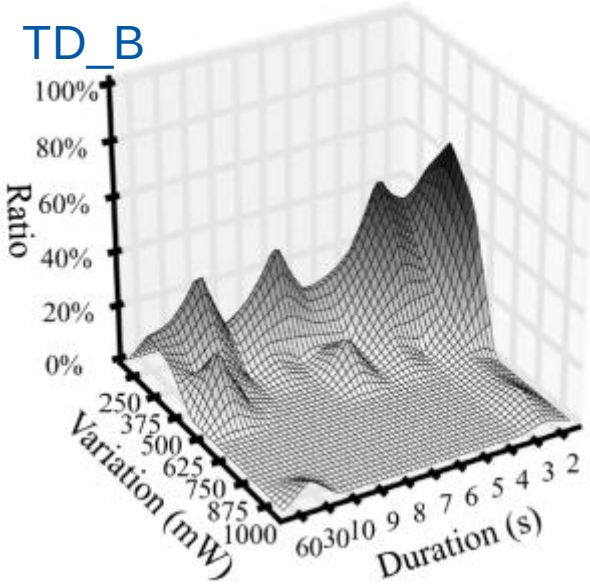
Bottom-up modeling methodology: Validation: Intel Core 2 - Responsiveness

- False positives

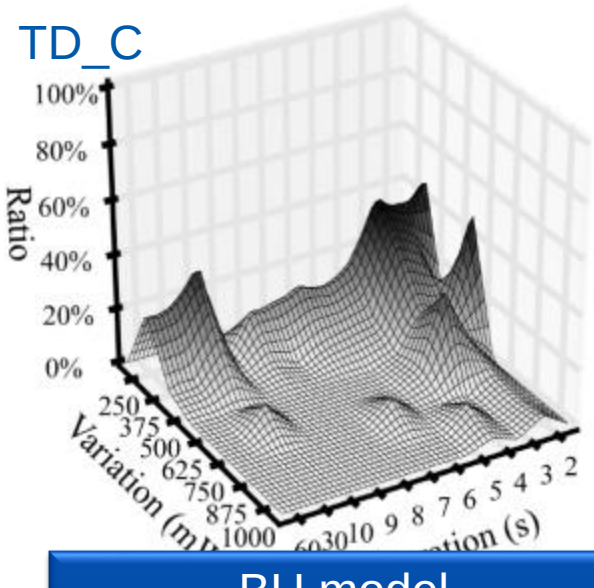
TD_A



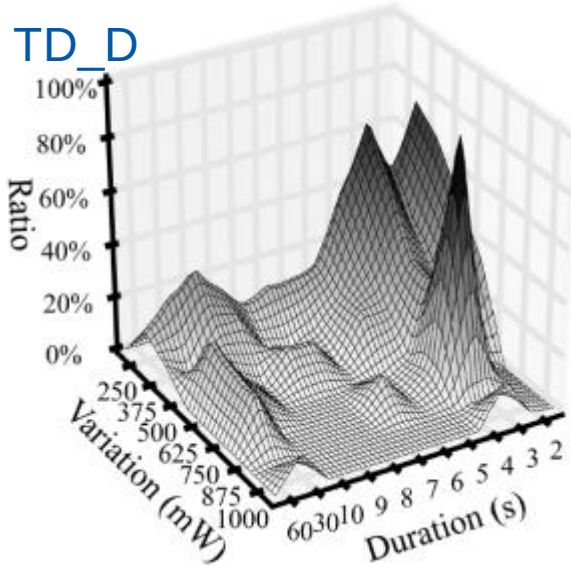
TD_B



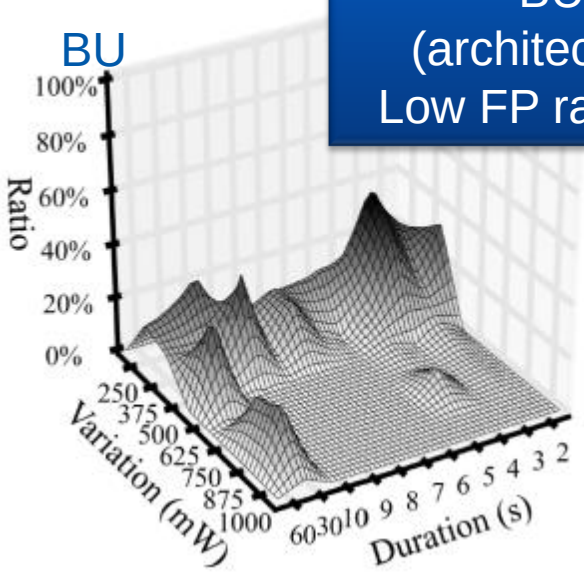
TD_C



TD_D

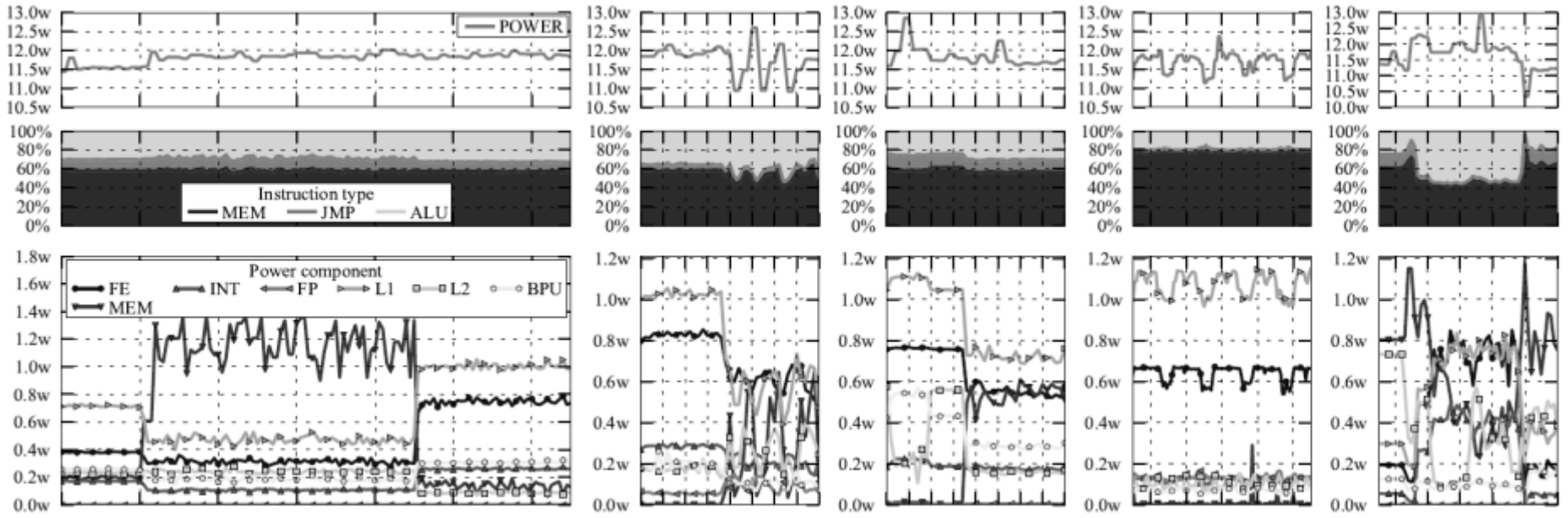


BU



BU model
(architecture aware)
Low FP ratio in general

Bottom-up modeling methodology: Applicability



- Bottom-up power model allows to decompose the power consumption among the components defined to gain insights about power consumption
  - changes in data locality (and component breakdown) but the global power consumption remains almost constant on such changes (cases a, c)
  - show power variations due to program phase changes (cases b, e)
  - shows the case in which the overall power and the major power components exhibit a similar pattern (cases d)

Bottom-up modeling methodology: Conclusions

- Hypothesis: Power modeling methods guided using basic ***knowledge*** of the modeled system generate models that are more:
  - Accurate and responsive
  - Informative and understandable
  - Robust and general
- The validation confirms the hypothesis:
  - Bottom-up power model provides the best trade-off between accuracy and responsiveness
  - Bottom-up power model provides the break-down of power consumption → informative and understandable
  - Bottom-up power model exhibits more consistent results across three different benchmark suites → robust and general
  - Bottom-up power modeling methodology does not required human intervention → Systematic process

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DECOMPOSABLE POWER MODELS: MODELING DVFS ENABLED PLATFORMS

Bottom-up modeling methodology under DVFS Introduction

- Dynamic Voltage Frequency Scaling
  - Allows to select the frequency of each core
  - Control the system power consumption
  - Extensively used to implement power aware policies
- Problem: solutions based on counter-based power models require a different model for each DVFS state combination
  - Sometimes the # DVFS state range is continuous

Cores	DVFS states	Models required (per chip DVFS)	Models required (per core DVFS)
2	3	**6**	**9**
2	18	**36**	**189**
8	3	**24**	**164**
8	18	**144**	**1562274**
16	3	**48**	**968**
16	18	**288**	**2203961429**

Bottom-up modeling methodology under DVFS Introduction

- Objective: define a power model for any DVFS state
  - DVFS state (frequency) should be an input
- Possible solutions:
  - Instead of using AR based on cycles use time for normalize them
    - Implicitly take into account frequency
    - E.g. instruction / second
    - Pitfall: Same # of event per second have different power consumption if the DVFS state is different
  - Add frequency/voltage as an extra variable of the model
    - Pitfall: loose the decomposability of the model
      - Which component represents the frequency?
  - Direct scale model coefficients using DVFS information
    - If we know the real values of frequency and voltage, we can apply the well-known power formula:

$$P = C \times V^2 \times f$$

    - Pitfall: this information usually not know or if so, it is processor dependent

Bottom-up modeling methodology under DVFS Methodology

- Observation: there is a relation between the model coefficients.
- Hypothesis: they can be modeled as function of frequency
  - Integer weight = f(freq) ?
- Methodology:
  - Generate a power model for each DVFS state
  - Model each component weight as a function of DVFS state (frequency)
    - Assume linear/exponential relation (choose the best)

$$f_i(F) = \begin{cases} (\alpha_i \times F) + \beta_i & if(R_{lin} \geq R_{exp}) \\ (\gamma_i \times \delta_i^F) + \epsilon_i & otherwise \end{cases}$$

  - Define power for any DVFS state (frequency) as:

$$P_{Total} = \left(\sum_{i=0}^{n} AR_i \times f_i(F) \right) + f_{intercept}(F)$$

Bottom-up modeling methodology under DVFS Methodology

- Intel Core 2 models

TD_A

Frequency	P_{fe}	P_{static}
0.80	-193.57	4157.56
0.90	-253.67	4414.45
1.00	-251.37	4821.96
1.10	-233.92	5180.92
1.2A	-219.64	5626.13
1.2B	-230.18	5834.09
1.40	-191.55	6735.29
1.50	-151.36	7037.26
1.60	-35.95	7315.41
1.70	-66.65	7777.85
1.80	5.26	8191.21
1.90	89.53	8492.11
2.00	151.99	8835.31
2.10	193.58	9090.53
2.20	349.25	9412.35
2.30	456.32	9868.19
2.40	577.35	10407.05
2.50	697.98	10833.35

TD_B

Frequency	P_{fe}	P_{mem}	P_{static}
0.80	180.86	25388.82	3410.59
0.90	195.85	24672.10	3563.78
1.00	222.98	23151.61	3944.57
1.10	296.29	23535.51	4221.27
1.2A	343.33	22292.35	4629.82
1.2B	311.25	21249.39	4838.78
1.40	421.51	20299.49	5623.59
1.50	476.92	19220.77	5917.65
1.60	573.28	18739.92	6222.46
1.70	623.05	18543.21	6582.08
1.80	684.45	17460.45	7021.12
1.90	787.18	17437.07	7295.54
2.00	877.76	17295.24	7599.86
2.10	870.97	15660.27	7946.11
2.20	1022.51	14954.08	8283.66
2.30	1116.05	14410.77	8765.84
2.40	1235.41	13894.70	9316.69
2.50	1324.78	12973.37	9796.22

Bottom-up modeling methodology under DVFS Methodology

- Intel Core 2 models

TD_C

Frequency	P_{fe}	P_{fp}	P_{stalls}	P_{mem}	P_{static}
0.80	46.29	671.84	-935.07	162216.25	4029.96
0.90	19.58	632.30	-1113.60	178157.72	4340.77
1.00	18.90	595.31	-1342.52	194949.46	4880.27
1.10	81.94	918.99	-1337.55	212998.25	5143.00
1.2A	50.05	863.75	-1666.86	225439.54	5841.55
1.2B	64.46	493.67	-1596.85	212185.23	5977.86
1.40	112.52	821.07	-2039.87	248071.15	7052.94
1.50	141.00	817.60	-2163.06	255157.66	7448.08
1.60	149.29	1002.09	-2334.26	261052.45	7954.79
1.70	172.18	1014.54	-2645.99	280181.17	8516.66
1.80	182.42	1099.87	-2980.37	293441.75	9195.57
1.90	270.62	1178.68	-2924.50	303847.48	9456.39
2.00	308.33	1253.70	-3166.10	319494.52	9955.21
2.10	295.93	1053.45	-3185.88	309139.94	10340.29
2.20	367.00	1240.98	-3517.34	315069.36	10949.01
2.30	409.10	1159.49	-3778.15	335379.08	11645.96
2.40	482.60	1312.54	-3921.65	344795.50	12322.14
2.50	473.86	1246.39	-4483.73	362280.25	13237.21

Bottom-up modeling methodology under DVFS Methodology

- Intel Core 2 models

BU

Frequency	P_{fe}	P_{int}	P_{fp}	P_{bpu}	P_{L1}	P_{L2}	P_{mem}	P_{static}
0.80	112.58	77.33	135.62	341.65	185.69	4986.51	16900.37	3197.00
0.90	113.11	93.47	21.04	272.08	202.11	5254.65	16877.96	3369.00
1.00	152.52	85.14	163.13	459.74	226.94	6307.34	16871.35	3689.00
1.10	165.00	116.73	174.16	548.07	271.98	6995.07	17097.06	3963.00
1.2A	216.64	103.87	164.34	539.39	301.79	8124.28	17145.63	4346.00
1.2B	212.45	101.25	172.82	556.75	296.97	8361.99	14312.48	4547.00
1.40	260.22	129.46	202.25	747.88	366.28	9758.93	14053.44	5305.00
1.50	295.59	154.45	153.96	750.05	395.15	11032.66	14515.71	5570.00
1.60	282.34	225.67	292.35	858.69	487.11	12385.24	14539.24	5858.00
1.70	340.78	195.91	235.81	924.24	497.19	13442.19	14542.80	6209.00
1.80	367.52	233.61	319.15	1069.23	562.56	14625.96	14743.76	6577.00
1.90	384.55	223.13	305.47	1081.20	560.63	14895.63	14553.59	6974.00
2.00	440.26	246.73	289.26	1108.87	622.46	16400.00	14715.39	7250.00
2.10	427.91	284.24	405.86	1281.89	682.95	17719.12	14649.53	7474.00
2.20	487.09	253.33	310.00	1254.77	670.67	18678.92	14554.42	7951.00
2.30	521.06	318.98	428.93	1472.57	774.25	20278.90	14845.86	8324.00
2.40	524.59	365.23	537.51	1594.74	859.96	20613.48	14791.74	8902.00
2.50	547.30	456.90	598.26	1725.30	982.08	23677.18	15214.53	9227.00

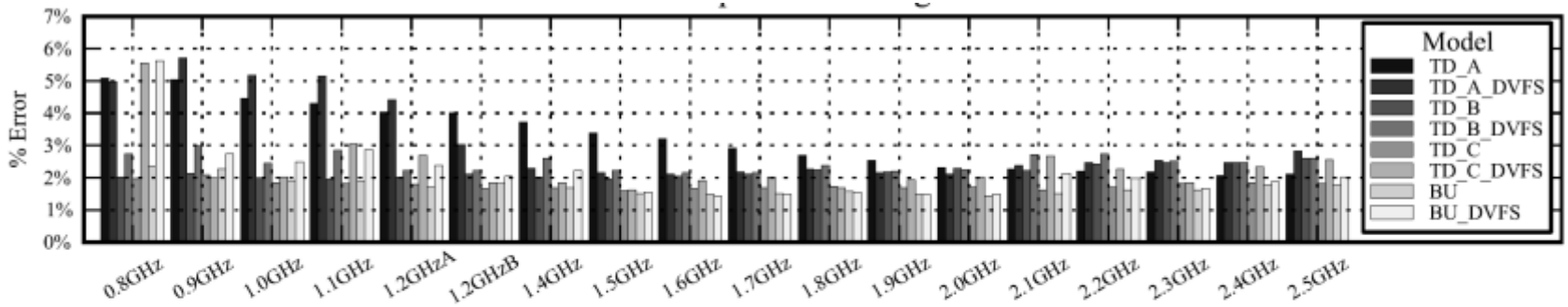
Bottom-up modeling methodology under DVFS Methodology

Models generated

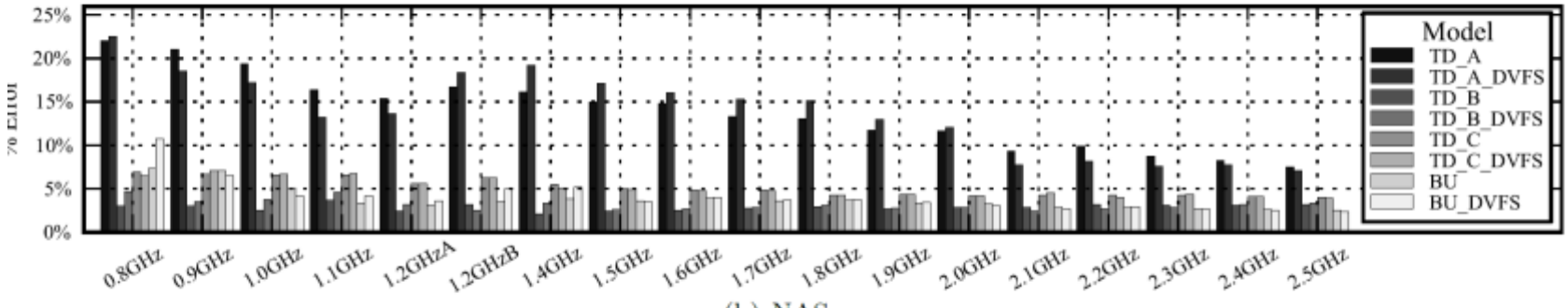
$$
\begin{aligned}
TD\_A\_DVFS &= (2.27 \times (12.5^{F}) - 263) \times AR_{ipc} + ((3895 \times F) + 1040) \\
TD\_B\_DVFS &= (76 \times (3.28^{F})) \times AR_{ipc} + (34549 \times (0.68^{F})) \times AR_{mem} + ((3721 \times F) + 268) \\
TD\_C\_DVFS &= ((279 \times F) - 257) \times AR_{ipc} + ((421 \times F) + 273) \times AR_{fp} + ((-1946 \times F) + 694) \times AR_{stalls} \\
&+ ((5266 \times F) - 423) + ((110437 \times F) + 85828) \times AR_{L2} \\
BU\_DVFS &= ((269 \times F) - 117) \times AR_{ipc} + (35.5 \times (2.66^{F})) \times AR_{int} + ((250 \times F) - 139) \times AR_{fp} \\
&+ ((794 \times F) - 385) \times AR_{bpu} + (94.5 \times (2.55^{F})) \times AR_{L1} + ((10637 \times F) - 4517) \times AR_{L2} + ((3567 \times F) + 173) \\
&+ \begin{cases} ((562 \times F) + 13553) \times AR_{mem} & if\ (F \geq 1.2GHz), \\ ((709 \times F) + 16268) \times AR_{mem} & otherwise \end{cases}
\end{aligned}
$$

- BU piece-wise model in the memory component
  - Rationale: memory component do not scale at the same pace as the core components
    - 100GHz for core frequency <= 1.2GHz, 200Hz for core frequency >= 1.2GHz
- High correlation coefficient in general
  - Corroborates hypothesis
  - TD_D
- Are the models accurate? Responsive?

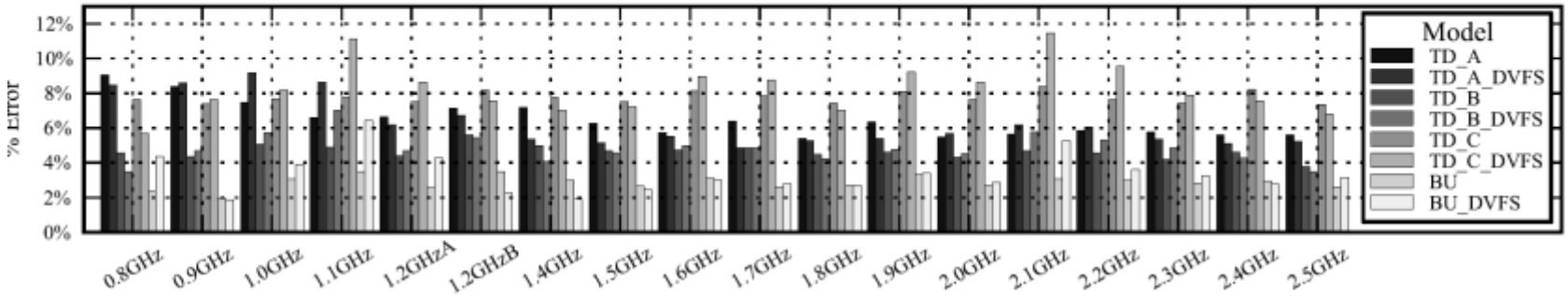
Bottom-up modeling methodology under DVFS Validation: Accuracy



(a) SPECcpu2006

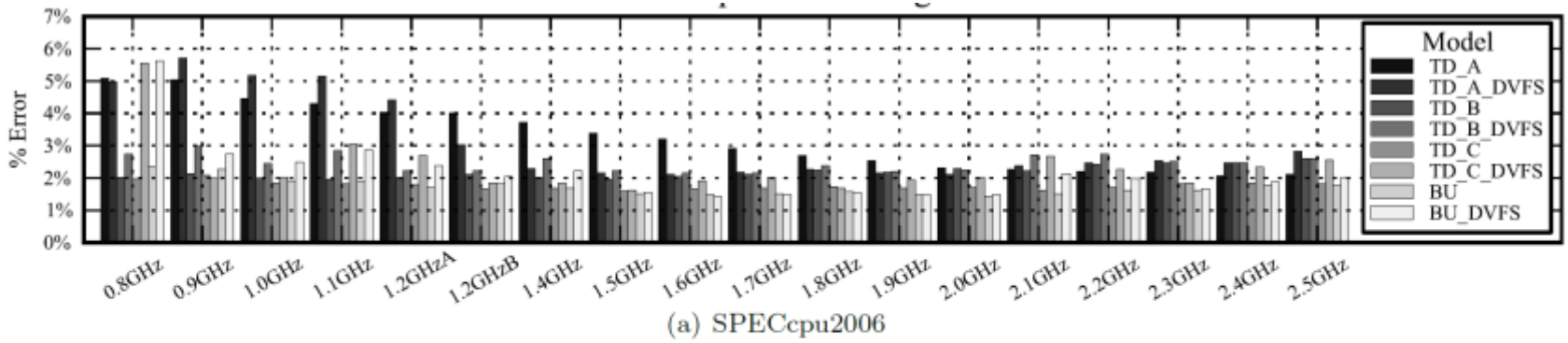


(b) NAS

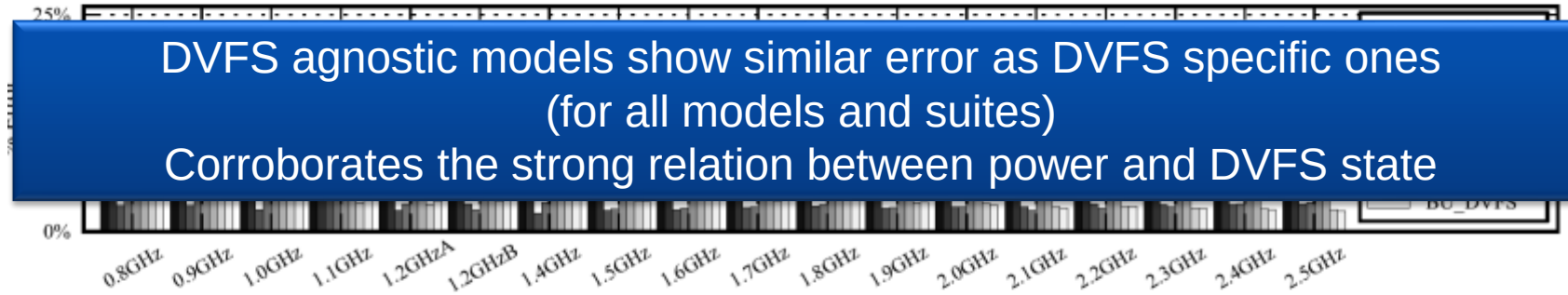


(c) LMBENCH

Bottom-up modeling methodology under DVFS
Validation: Accuracy

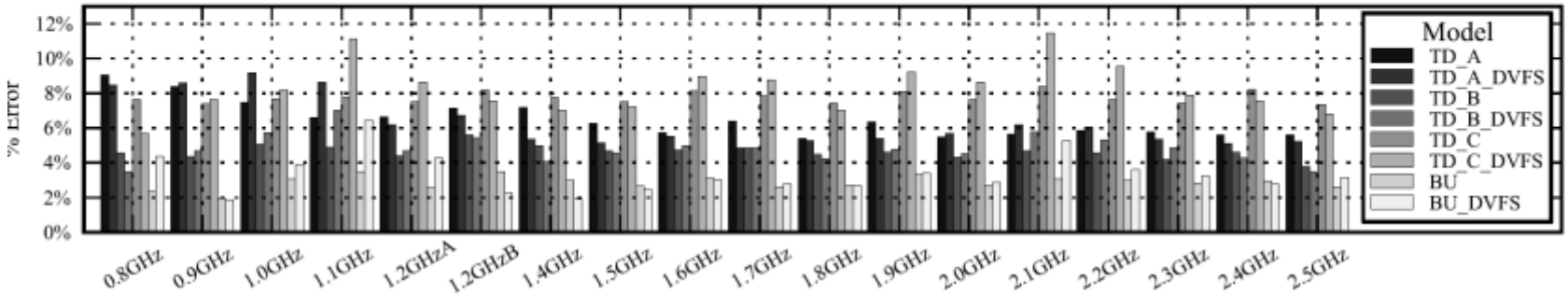


(a) SPECcpu2006



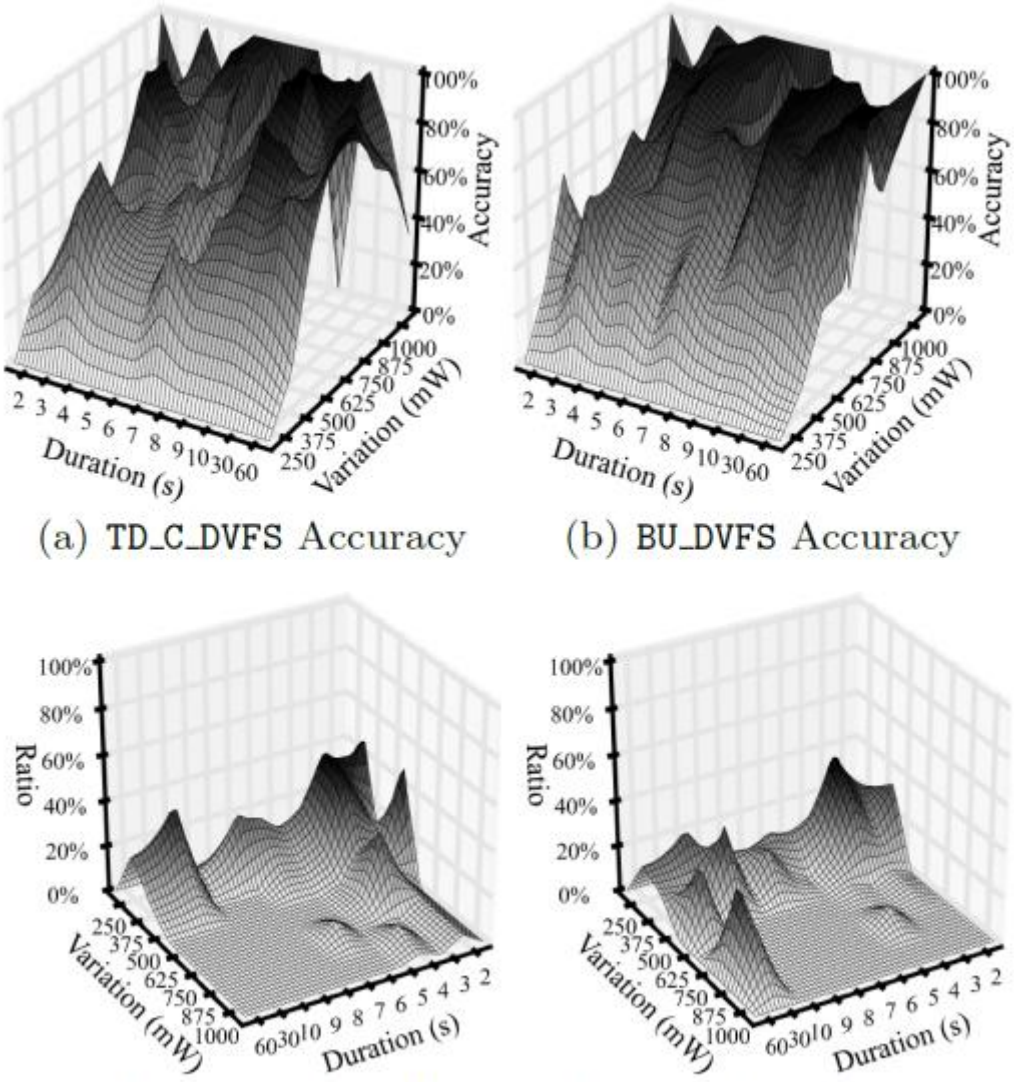
(b) NAS

DVFS agnostic models show similar error as DVFS specific ones
(for all models and suites)
Corroborates the strong relation between power and DVFS state



(c) LMBENCH

Bottom-up modeling methodology under DVFS
Validation: Responsiveness



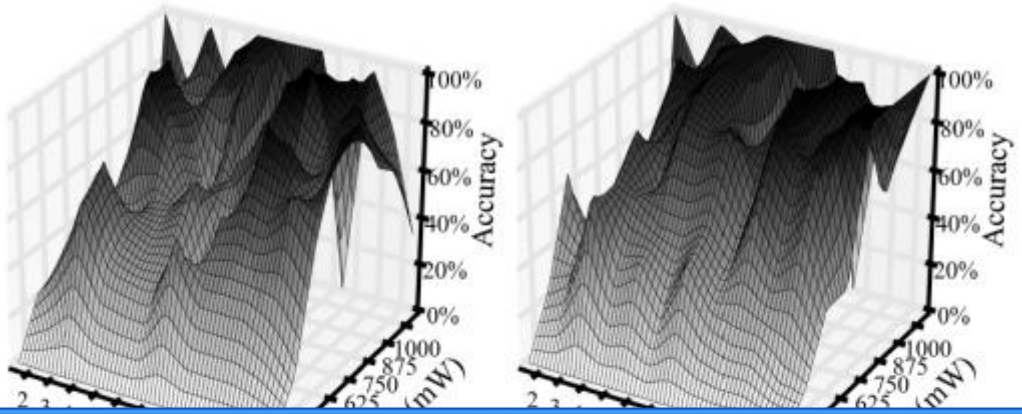
(a) TD_C_DVFS Accuracy

(b) BU_DVFS Accuracy

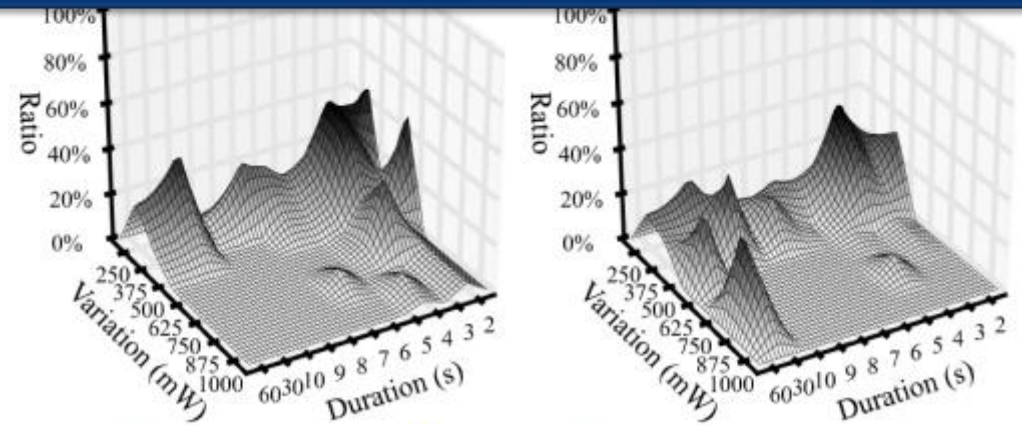
(c) TD_C_DVFS False Positives

(d) BU_DVFS False Positives

Bottom-up modeling methodology under DVFS
Validation: Responsiveness



DVFS agnostic models show the same responsiveness as the DVFS specific ones
(for all models and suites)
Corroborates the strong relation between power and DVFS state



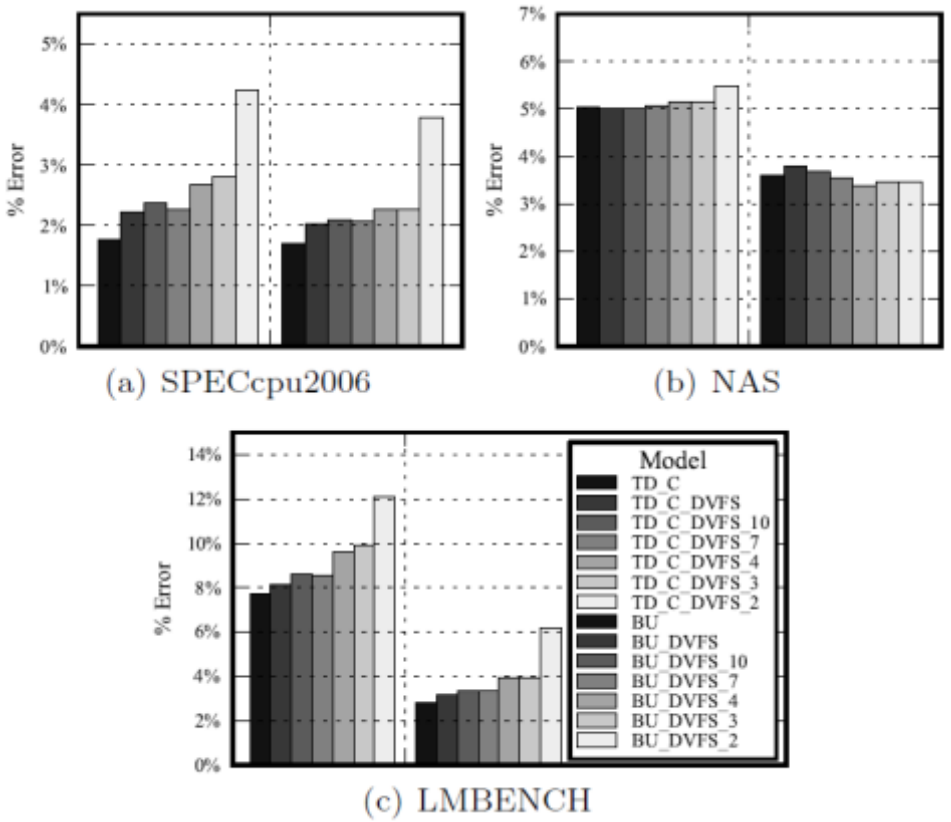
(c) TD_C_DVFS False Positives (d) BU_DVFS False Positives

Bottom-up modeling methodology under DVFS
Modeling time reduction

- Generating a DVFS agnostic model requires a to generate a model for each DVFS state
- Time consuming process (impractical)
  - Large number of cores
  - Large number of DVFS states
- Hypothesis: it is possible to reduce the number of training DVFS specific models and derive a DVFS agnostic one without affecting accuracy
- Methodology:
  - Study how affects the reduction of the number of training DVFS states to the accuracy and responsiveness of the model

Bottom-up modeling methodology under DVFS
Modeling time reduction

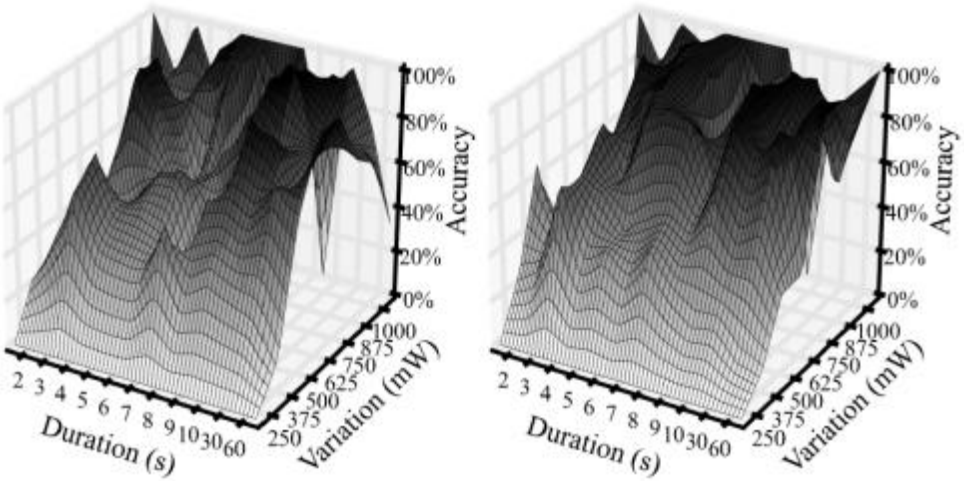
- Reducing the # of training DVFS states from 18 to 3, increment average error <2 percentage points
  - Strong robustness to DVFS
- Using only 2 DVFS states (max and min), show higher error
  - Too simple → linear model
- NAS Parallel benchmarks less affected
  - Memory bound



(a) SPECcpu2006 (b) NAS

(c) LMBENCH

Bottom-up modeling methodology under DVFS
Modeling time reduction

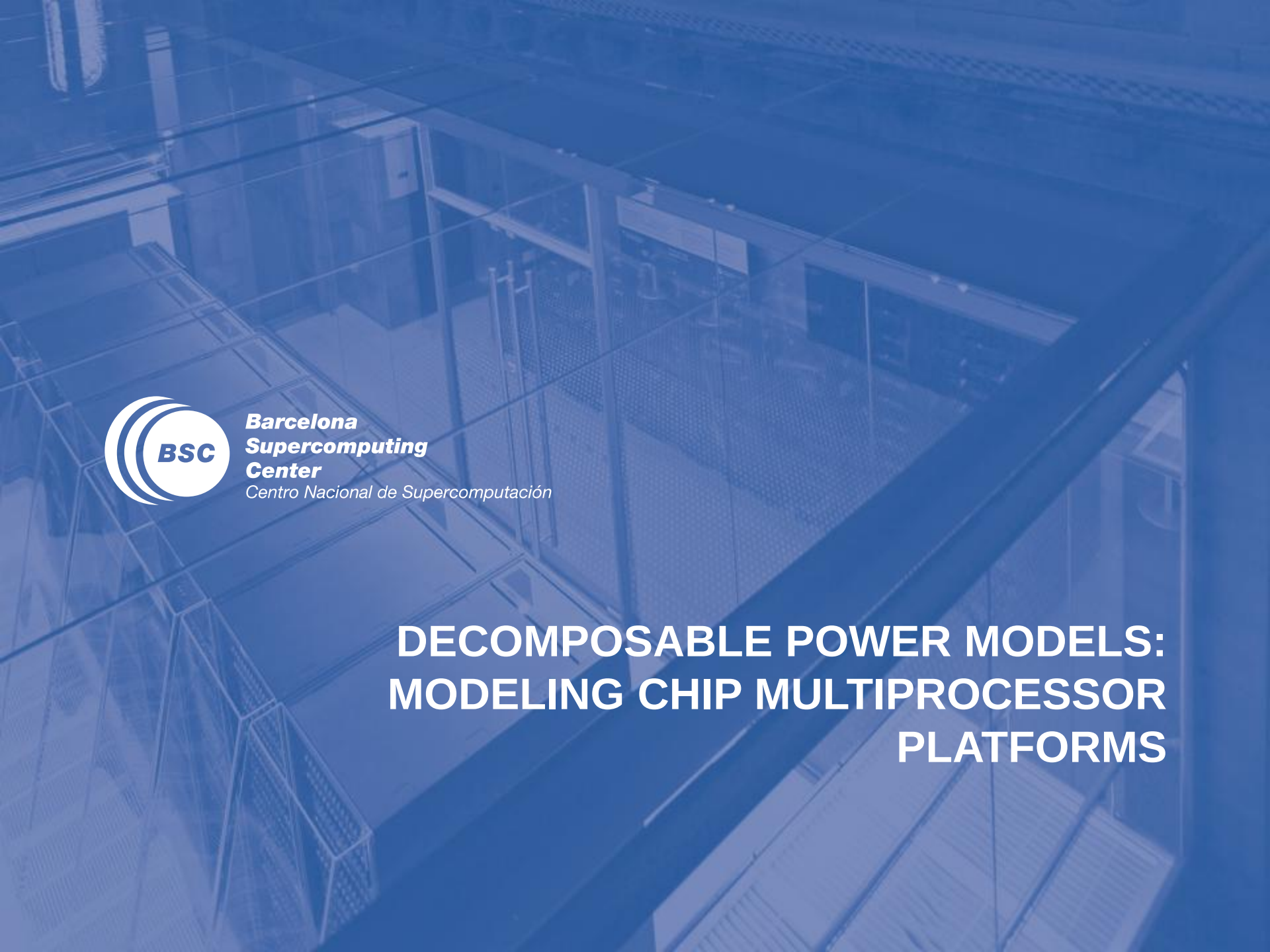


(a) TD_C_DVFS_3 Accuracy (b) BU_DVFS_3 Accuracy

- Reducing the # of training DVFS states from 18 to 3 does not affect the responsiveness of the model
  - Responsiveness depend on the model design: the inputs and the BU generation method (orthogonal to DVFS)

Bottom-up modeling methodology under DVFS Conclusions

- Novel 2-step methodology proposed for deriving DVFS agnostic models from DVFS specific ones
  - Keeps the accuracy of the models
  - Keeps the properties of the models
    - Decomposability
    - Responsiveness
- The number of DVFS required to generate the DVFS agnostic models can be reduced up to 3
  - There exist a strong relation between coefficients (DVFS state) and power consumption



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DECOMPOSABLE POWER MODELS: MODELING CHIP MULTIPROCESSOR PLATFORMS

Bottom-up modeling methodology: CMP extension Introduction

- Challenge: CMP architectures are here
  - Intel Core 2 : 2 cores
  - Intel Core i7: 8 cores
- How to extend the bottom-up power modeling method to account the CMP effect?
  - Follow the same bottom-up/incremental approach
  - Use basic ***knowledge*** to guide the modeling process
- Assumptions (***knowledge***):
  - The overall power consumption is composed by the dynamic power of each HW thread running at each core, the static power consumption of each core enabled and the uncore power consumption (intercept)
  - Model each HW thread using the Bottom-Up modeling approach
  - Model the intercept as a function of the number of cores enabled

Bottom-up modeling methodology: CMP extension Methodology

- Gather empirical data for each number of cores enabled
  - Only the data from the random mix is required
- For each number of cores enabled
  - Apply the intercept tuning pass of the Bottom-Up method
- The overall power is the addition of the power consumption of BU model of each core plus the intercept tuned for that number of cores

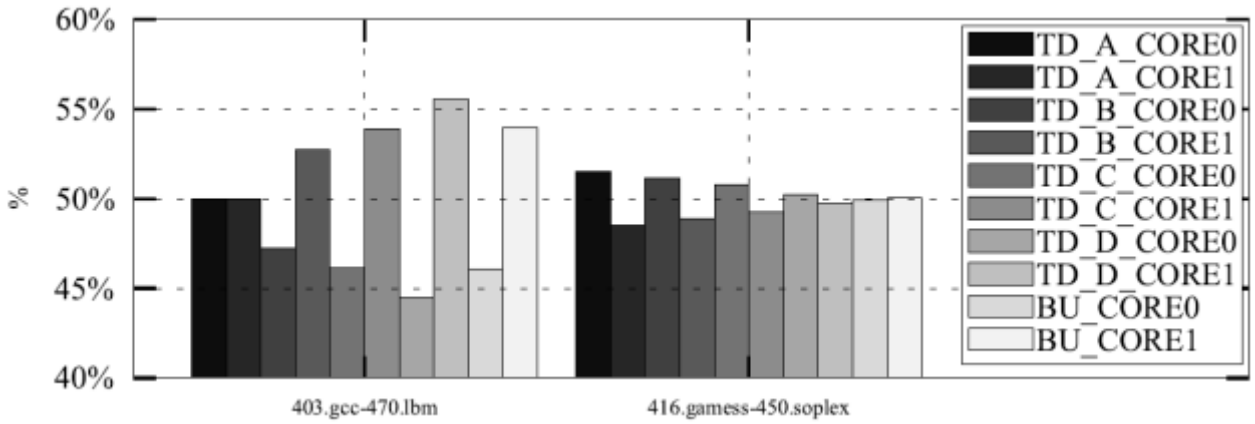
$$Power = \sum_{j=1}^{cores} \left(\left(\sum_{i=0}^{n} AR_{i^j} \times P_i \right) \right) + P_{Static^j}$$

Bottom-up modeling methodology: CMP extension Validation

Accuracy:

Benchmark *cpu0–cpu1*	Power (mW)	Model MICRO %err	 σ	Model FE %err	 σ	Model +FSB %err	 σ	Processor (mw)	Core 0 (mw)	Core 1 (mw)
400.perlbench-436.cactusADM	18540.20	4.10	1.85	4.16	1.38	3.56	1.41	19298.02	10707.80(55.49%)	8590.22(44.51%)
400.perlbench-453.povray	19686.18	6.08	1.09	0.52	0.53	0.60	0.58	20882.35	10762.56(51.54%)	10119.79(48.46%)
400.perlbench-454.calculix	20462.40	4.38	1.52	2.33	0.97	5.10	1.94	21352.66	10768.96(50.43%)	10583.71(49.57%)
400.perlbench-462.libquantum	24034.74	6.80	1.78	17.46	1.21	13.54	1.77	22398	10954.22(48.91%)	11443.79(51.09%)
400.perlbench-473.astar	19377.92	5.62	3.11	2.71	1.89	1.91	1.33	20444.1	10776.56(52.71%)	9667.53(47.29%)
462.libquantum-436.cactusADM	22661.80	9.22	2.08	13.78	1.54	8.96	2.12	20569.68	11432.35(55.58%)	9137.33(44.42%)
462.libquantum-453.povray	23896.88	9.02	1.38	16.37	1.25	13.34	1.55	21739.6	11508.96(52.94%)	10230.64(47.06%)
462.libquantum-454.calculix	24716.67	10.15	1.45	15.12	1.73	9.31	2.01	22206.72	11495.94(51.77%)	10710.77(48.23%)
462.libquantum-473.astar	23353.39	9.26	2.00	17.01	1.69	11.70	2.97	21188.89	11399.75(53.80%)	9789.13(46.20%)
473.astar-436.cactusADM	18964.17	4.40	3.76	3.01	1.80	3.04	2.15	18131.4	9523.93(52.53%)	8607.47(47.47%)
473.astar-453.povray	20049.61	3.12	2.87	3.21	3.35	3.62	3.40	19776.85	9619.77(48.64%)	10157.08(51.36%)
473.astar-454.calculix	20798.19	3.04	3.59	3.01	2.73	3.78	1.79	20206.38	9628.47(47.65%)	10577.91(52.35%)
436.cactusADM-453.povray	18477.38	1.47	1.09	5.09	1.50	4.36	1.50	18708.42	8590.34(45.92%)	10118.09(54.08%)
436.cactusADM-454.calculix	19269.42	1.68	1.17	6.06	2.41	8.81	3.08	19143.36	8569.32(44.76%)	10574.04(55.24%)
453.povray-454.calculix	20163.70	2.84	0.89	3.46	1.30	6.78	2.52	20732.2	10142.99(48.92%)	10589.20(51.08%)
AVERAGE		**4.63**	**2.9**	**5.12**	**6.31**	**6.09**	**4.15**			

Applicability:



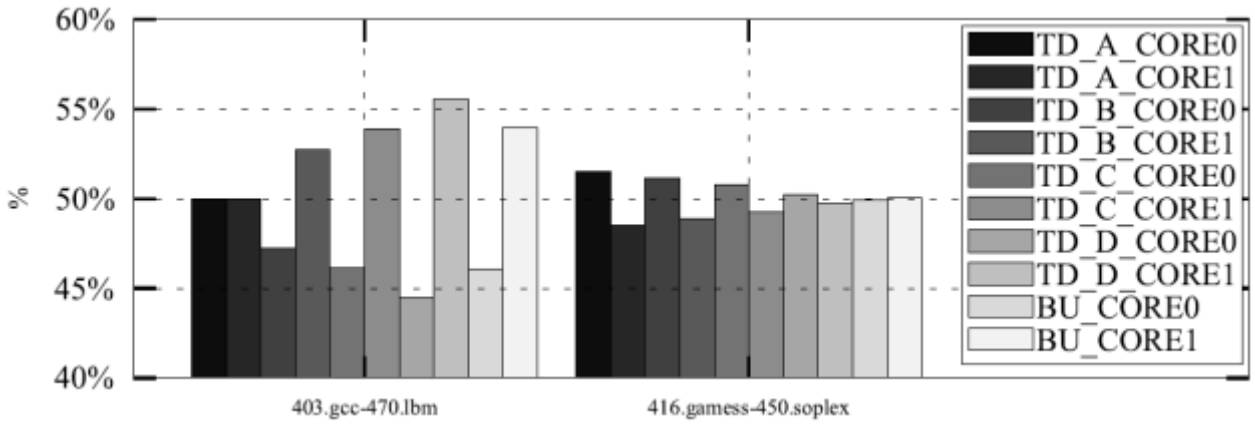
Bottom-up modeling methodology: CMP extension Validation

Accuracy:

Benchmark cpu0–cpu1	Power (mW)	Model MICRO %err	Model MICRO σ	Model FE %err	Model FE σ	Model +FSB %err	Model +FSB σ	Processor (mw)	Core 0 (mw)	Core 1 (mw)
400.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]51%)
400										.46%)
400										.57%)
400										.09%)
400.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	29%)
462.libquantum-436.cactusADM	22661.80	9.22	2.08	13.78	1.54	8.96	2.12	20569.68	11432.35(55.58%)	9137.33(44.42%)
462.libquantum-453.povray	23896.88	9.02	1.38	16.37	1.25	13.34	1.55	21739.6	11508.96(52.94%)	10230.64(47.06%)
462.libquantum-454.calculix	24716.67	10.15	1.45	15.12	1.73	9.31	2.01	22206.72	11495.94(51.77%)	10710.77(48.23%)
462.libquantum-473.astar	23353.39	9.26	2.00	17.01	1.69	11.70	2.97	21188.89	11399.75(53.80%)	9789.13(46.20%)
473.astar-436.cactusADM	18964.17	4.40	3.76	3.01	1.80	3.04	2.15	18131.4	9523.93(52.53%)	8607.47(47.47%)
473.astar-453.povray	20049.61	3.12	2.87	3.21	3.35	3.62	3.40	19776.85	9619.77(48.64%)	10157.08(51.36%)
473.astar-454.calculix	20798.19	3.04	3.59	3.01	2.73	3.78	1.79	20206.38	9628.47(47.65%)	10577.91(52.35%)
436.cactusADM-453.povray	18477.38	1.47	1.09	5.09	1.50	4.36	1.50	18708.42	8590.34(45.92%)	10118.09(54.08%)
436.cactusADM-454.calculix	19269.42	1.68	1.17	6.06	2.41	8.81	3.08	19143.36	8569.32(44.76%)	10574.04(55.24%)
453.povray-454.calculix	20163.70	2.84	0.89	3.46	1.30	6.78	2.52	20732.2	10142.99(48.92%)	10589.20(51.08%)
AVERAGE		**4.63**	**2.9**	**5.12**	**6.31**	**6.09**	**4.15**			

Similar levels of accuracy
Lower standard deviation in errors → robustness across workloads

Applicability:



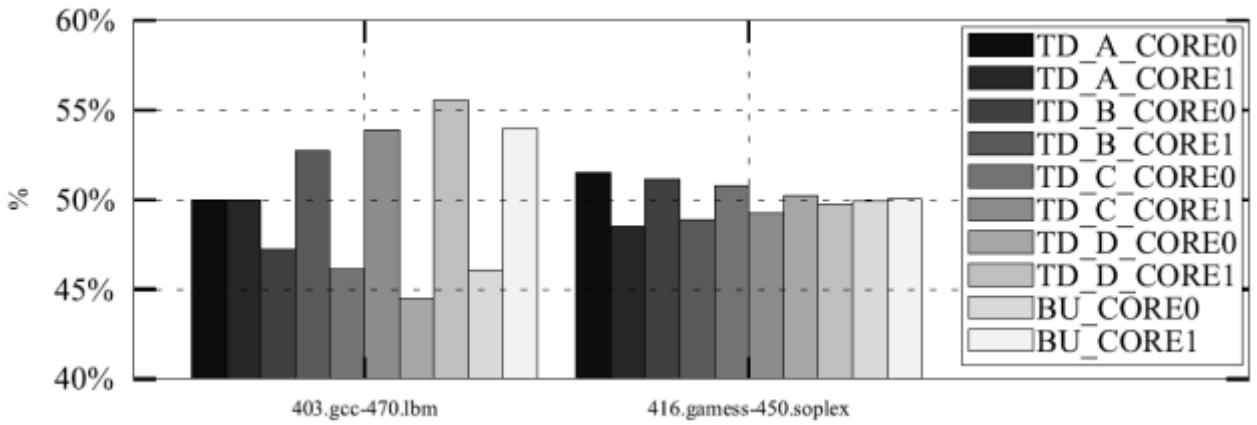
Bottom-up modeling methodology: CMP extension Validation

Accuracy:

Benchmark *cpu0–cpu1*	Power (mW)	Model MICRO %err	Model MICRO σ	Model FE %err	Model FE σ	Model +FSB %err	Model +FSB σ	Processor (mw)	Core 0 (mw)	Core 1 (mw)
400.perlbench-436.cactusADM	18540.20	4.10	1.85	4.16	1.38	3.56	1.41	19298.02	10707.80(55.49%)	8590.22(44.51%)
400.perlbench-453.povray	19686.18	6.08	1.09	0.52	0.53	0.60	0.58	20882.35	10762.56(51.54%)	10119.79(48.46%)
400.perlbench-454.calculix	20462.40	4.38	1.52	2.33	0.97	5.10	1.94	21352.66	10768.96(50.43%)	10583.71(49.57%)
400.perlbench-462.libquantum	24034.74	6.80	1.78	17.46	1.21	13.54	1.77	22398	10954.22(48.91%)	11443.79(51.09%)
400.perlbench-473.astar	19377.92	5.62	3.11	2.71	1.89	1.91	1.33	20444.1	10776.56(52.71%)	9667.53(47.29%)
462.libquantum-436.cactusADM	22661.80	9.22	2.08	13.78	1.54	8.96	2.12	20569.68	11432.35(55.58%)	9137.33(44.42%)
462.libquantum-453.povray	23896.88	9.02	1.38	16.37	1.25	13.34	1.55	21739.6	11508.96(52.94%)	10230.64(47.06%)
462.libquantum-454.calculix	24716.67	10.15	1.45	15.12	1.73	9.31	2.01	22206.72	11495.94(51.77%)	10710.77(48.23%)
462.libquantum-473.astar	23353.39	9.26	2.00	17.01	1.69	11.70	2.97	21188.89	11399.75(53.80%)	9789.13(46.20%)
473.astar-436.cactusADM	18964.17	4.40	3.76	3.01	1.80	3.04	2.15	18131.4	9523.93(52.53%)	8607.47(47.47%)
473.astar-453.povray	20049.61	3.12	2.87	3.21	3.35	3.62	3.40	19776.85	9619.77(48.64%)	10157.08(51.36%)
473.astar-454.calculix	20798.19	3.04	3.59	3.01	2.73	3.78	1.79	20206.38	9628.47(47.65%)	10577.91(52.35%)
436.cactusADM-453.povray	18477.38	1.47	1.09	5.09	1.50	4.36	1.50	18708.42	8590.34(45.92%)	10118.09(54.08%)
436.cactusADM-454.calculix	19269.42	1.68	1.17	6.06	2.41	8.81	3.08	19143.36	8569.32(44.76%)	10574.04(55.24%)
453.povray-454.calculix	20163.70	2.84	0.89	3.46	1.30	6.78	2.52	20732.2	10142.99(48.92%)	10589.20(51.08%)
AVERAGE		**4.63**	**2.9**	**5.12**	**6.31**	**6.09**	**4.15**			

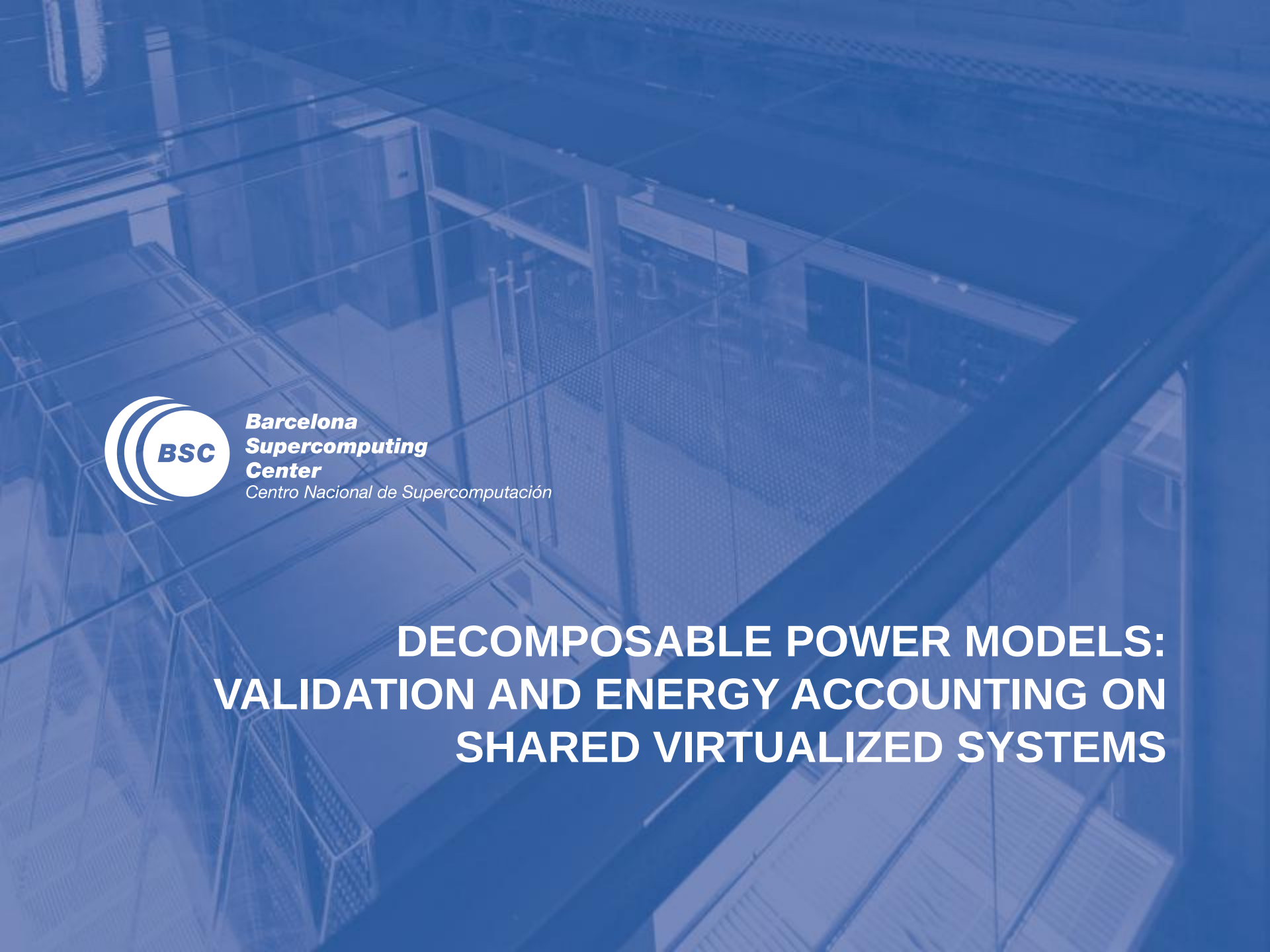
Applicability:

Higher accuracy in the per-core power decomposition



Bottom-up modeling methodology: CMP extension Conclusions

- Bottom-up power modeling methodology can be extended to multi-cores
  - Keeps the same level of accuracy
  - Allows the decompose the power consumption between the cores
- The extension is simple
  - Gather data for N cores using the random micro-benchmarks
  - Re-tune the intercept
  - No need of human intervention



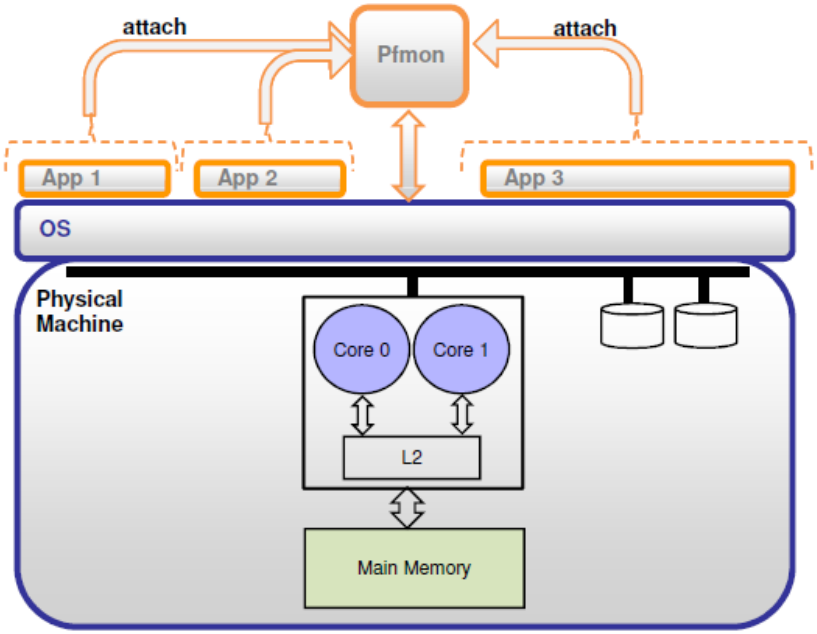
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

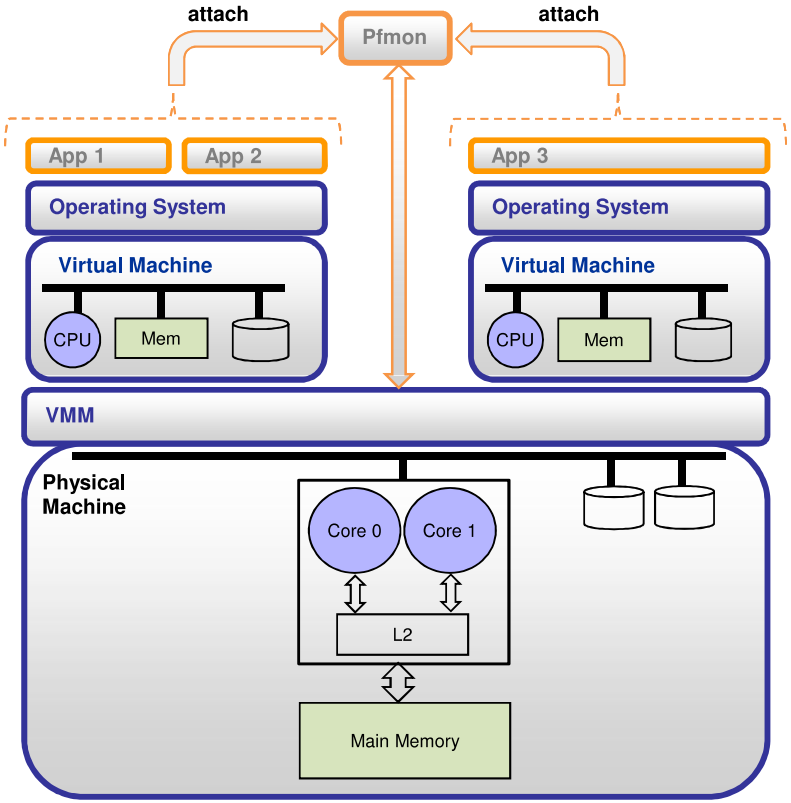
DECOMPOSABLE POWER MODELS: VALIDATION AND ENERGY ACCOUNTING ON SHARED VIRTUALIZED SYSTEMS

Bottom-up modeling methodology under Virtualization Introduction

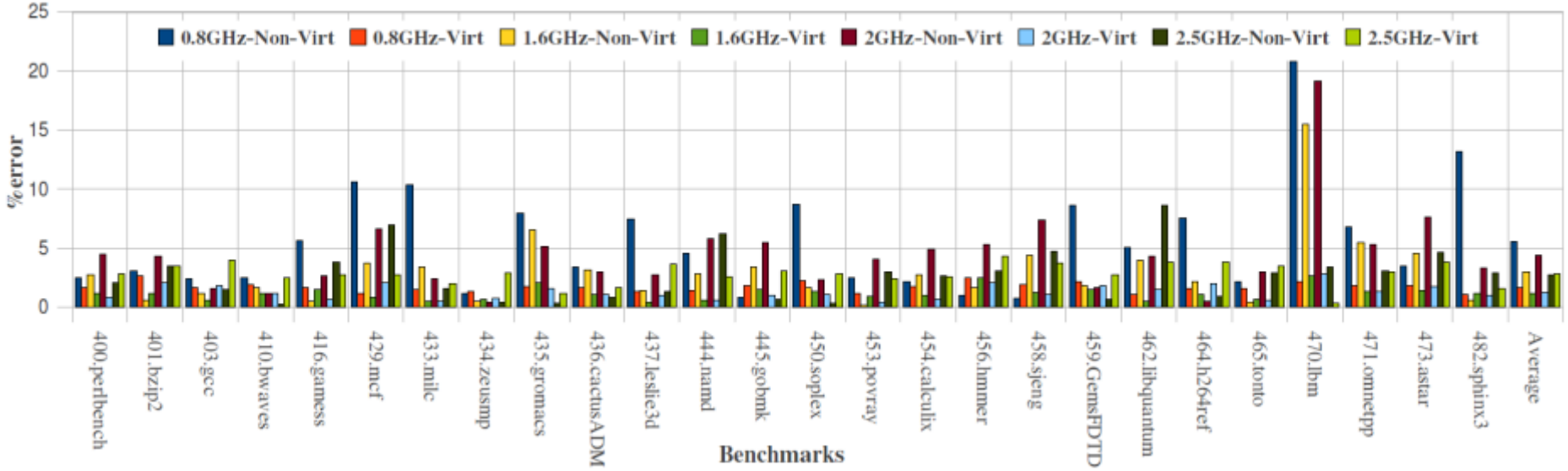
- Hypothesis: The counter-based power model can be used on to account the activity of virtualized environments
  - Then, they can be used to perform energy accounting
- Environment:
  - Platform: Intel Core 2
  - Experiment: run various VMs on the platform and apply validate the model
- Method:
  - Model Validation
    - Apply the model on virtualized and non-virtualized environments
  - Accounting Validation:
    - Estimate the power consumption each VM running on the system
    - The sum of the power consumption of each VM running on the system ~ the overall platform power consumption

Bottom-up modeling methodology under Virtualization Environment



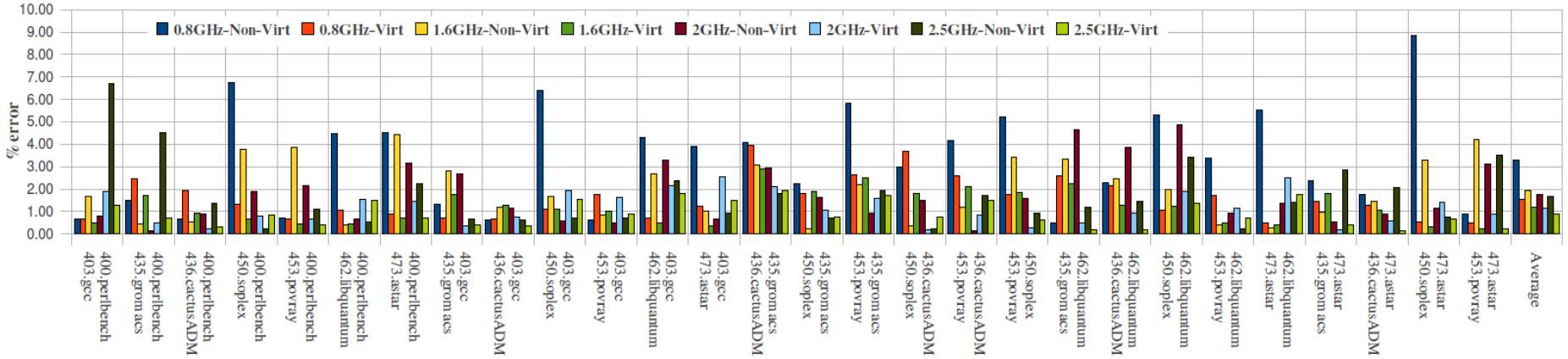


Bottom-up modeling methodology under Virtualization
Model Validation – Single core



- Models valid for virtualized/not-virtualized environments
- Models valid for the different frequencies studied
- Average errors below 5%
  - 470.lbm, memory bound → high error in non virtualized, low error in virtualized
    - pathological case removed due to virtualization

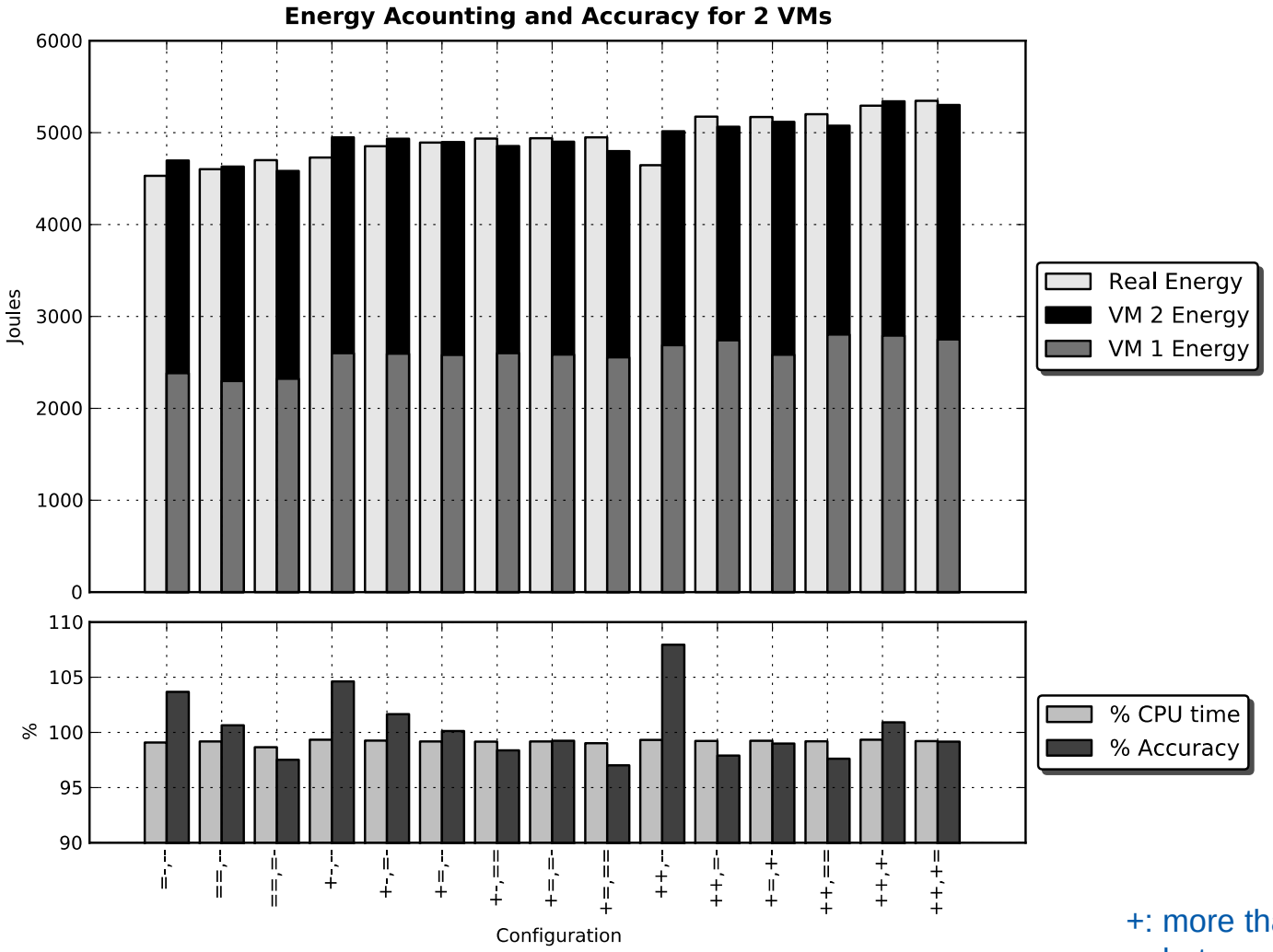
Bottom-up modeling methodology under Virtualization
Model Validation – Dual core



- Models valid for virtualized/not-virtualized environments
- Models valid for the different frequencies studied
- Average errors below 5%

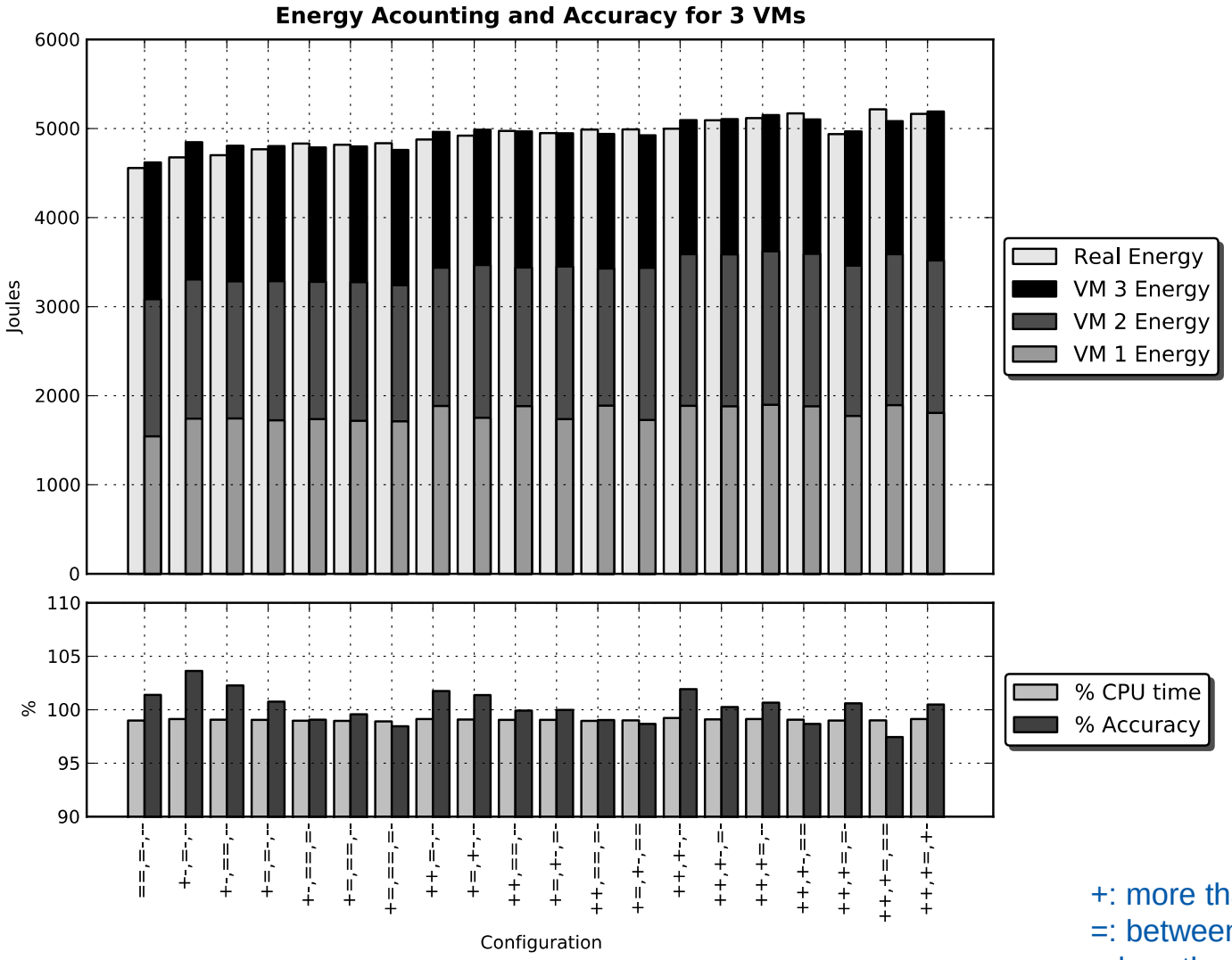
Counter-based power models are orthogonal to DVFS and to the virtualization technology

Bottom-up modeling methodology under Virtualization
Energy accounting validation



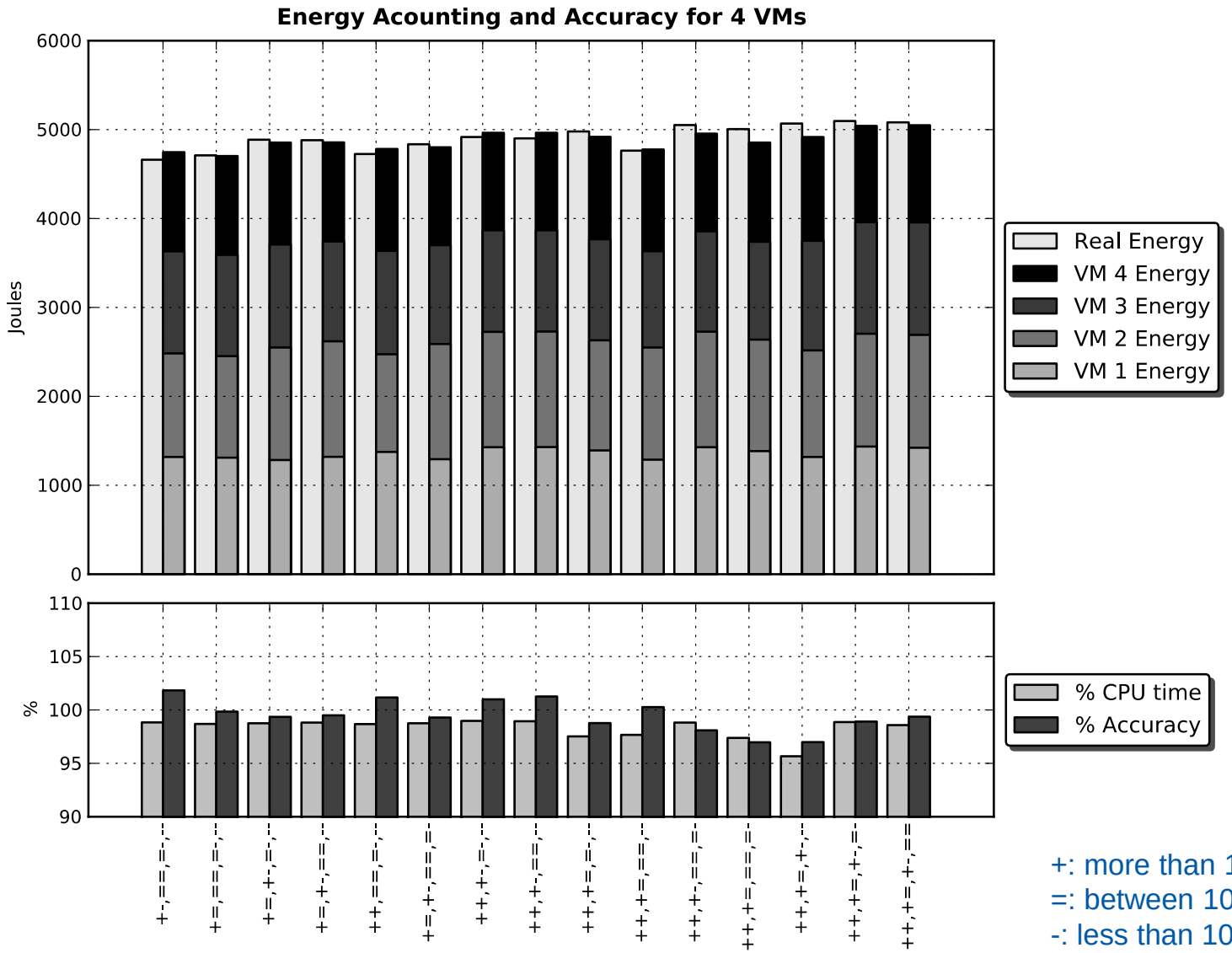
+: more than 11W
=: between 10W and 11W
-: less than 10W

Bottom-up modeling methodology under Virtualization
Energy accounting validation



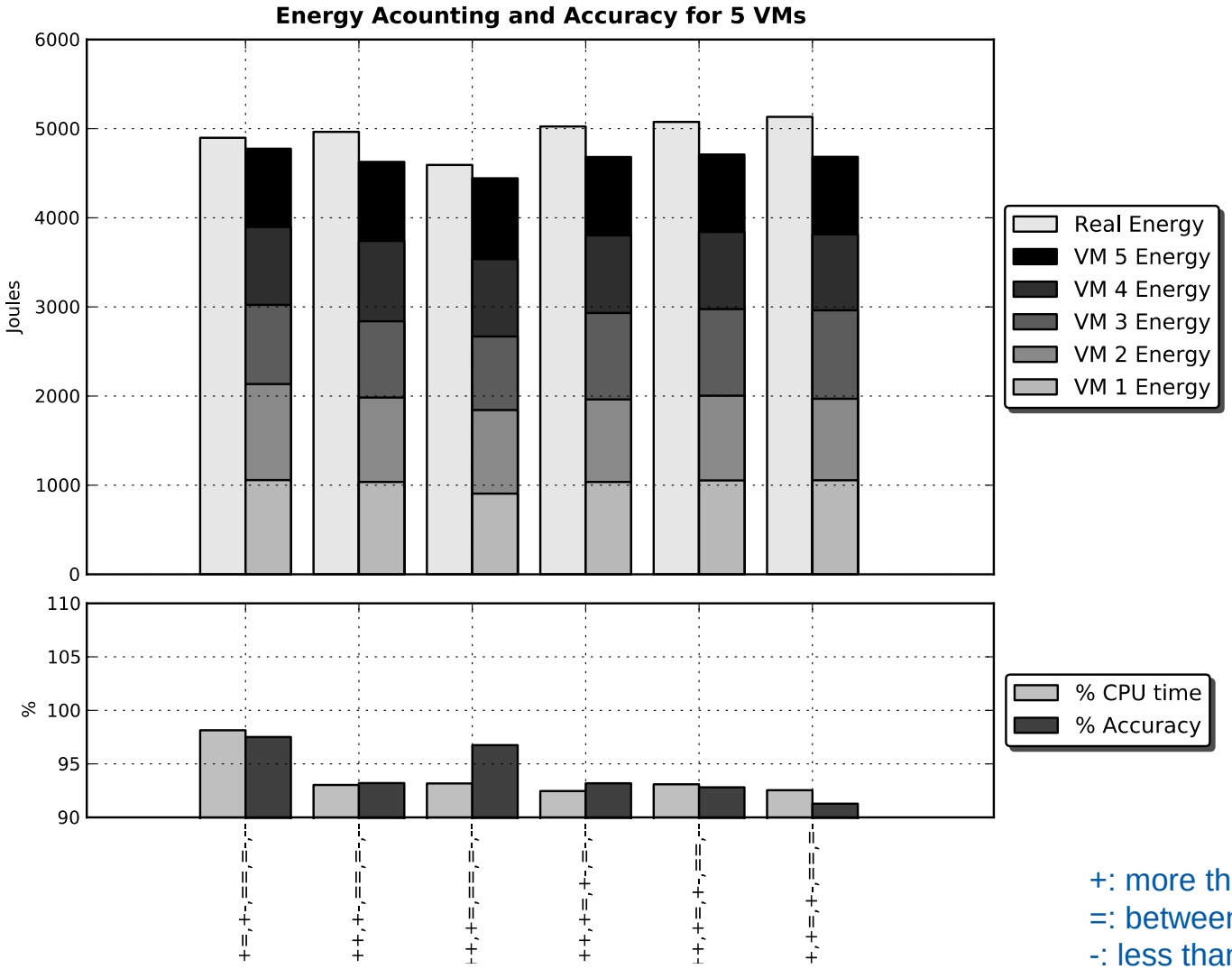
+: more than 11W
=: between 10W and 11W
-: less than 10W

Bottom-up modeling methodology under Virtualization
Energy accounting validation



+: more than 11W
=: between 10W and 11W
-: less than 10W

Bottom-up modeling methodology under Virtualization
Energy accounting validation

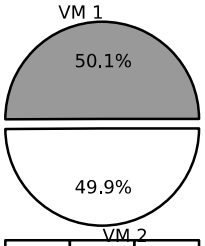


+: more than 11W
=: between 10W and 11W
-: less than 10W

Bottom-up modeling methodology under Virtualization
Energy accounting: applicability

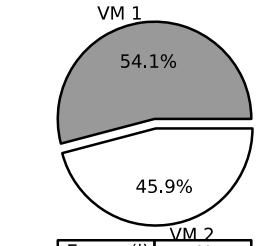
- Not every application consumes the same amount of energy per equal time of execution
  - Observed differences ranging from 17% up to 30%
    - Even they use a similar amount of CPU cycles

Time Breakdown



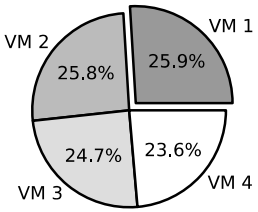
	Workload	Time(s)	%
VM 1	++	298.21	
VM 2	=-	297.11	0.37

Energy Breakdown



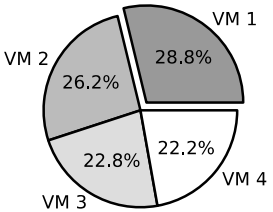
	Energy(j)	%
VM 1	2739.36	
VM 2	2324.63	17.84

Time Breakdown



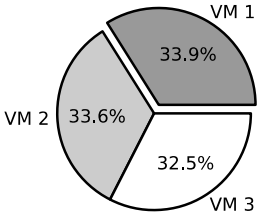
	Workload	Time(s)	%
VM 1	++	153.78	
VM 2	+-	152.98	0.52
VM 3	==	146.24	5.15
VM 4	=-	139.92	9.91

Energy Breakdown



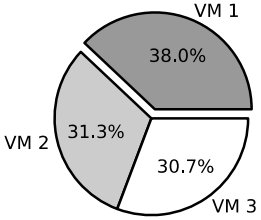
	Energy(j)	%
VM 1	1426.95	
VM 2	1298.90	9.86
VM 3	1128.78	26.42
VM 4	1099.23	29.81

Time Breakdown



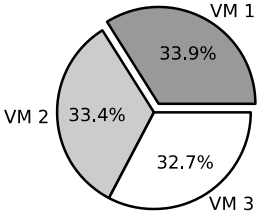
	Workload	Time(s)	%
VM 1	++	201.75	
VM 2	=-	199.73	1.01
VM 3	--	193.26	4.39

Energy Breakdown



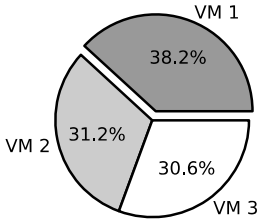
	Energy(j)	%
VM 1	1884.46	
VM 2	1551.88	21.43
VM 3	1524.94	23.58

Time Breakdown

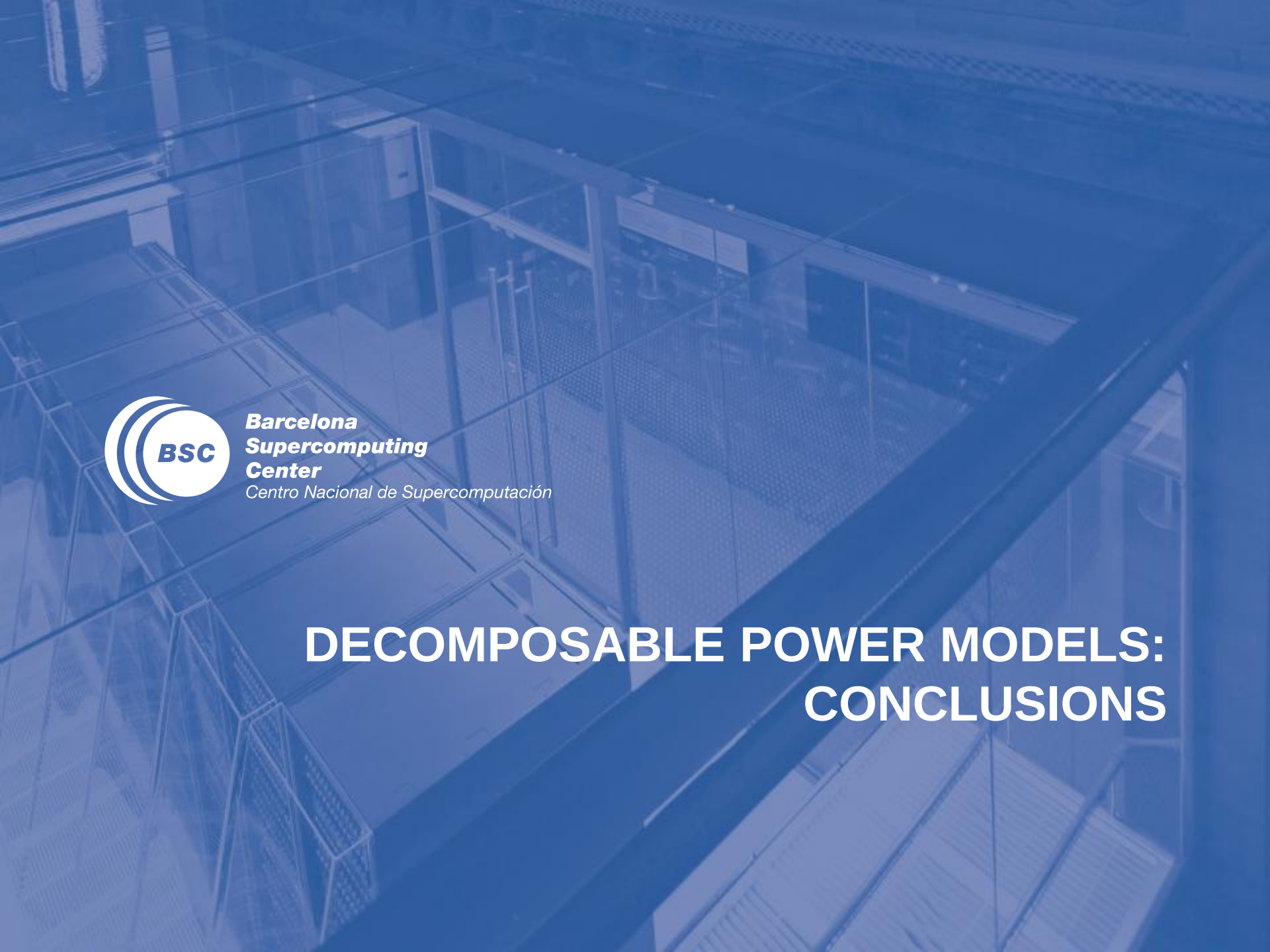


	Workload	Time(s)	%
VM 1	++	201.41	
VM 2	==	198.29	1.57
VM 3	=-	194.06	3.78

Energy Breakdown



	Energy(j)	%
VM 1	1886.43	
VM 2	1540.90	22.42
VM 3	1511.77	24.78



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DECOMPOSABLE POWER MODELS: CONCLUSIONS

Conclusions

- Methodology to generate Bottom-Up counter-based power models
  - Systematic
  - Accurate and general
  - Responsive
- Methodology to extend them to DVFS aware systems
  - Derive DVFS agnostic from DVFS specific
  - Maintain model properties
    - DVFS orthogonal
  - Reduce the number of training states
    - Modeling time reduction
- Methodology to extend BU models to CMP systems
- Validation on virtualized systems
  - Orthogonal virtualization technology

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PUBLICATIONS

Selected Publications

- R. Bertran et al. *"PoTrA: A Framework for Building Power Models for Next Generation Multicore Architectures"*, SIGMETRICS'12, June 11-15, 2012, London, England, UK
- F. Bellosa. *"The benefits of event: driven energy accounting in power-sensitive systems"*, In EW 9'00, Kolding, Denmark, 2000.
- C. Isci et al. *"Runtime power monitoring in high-end processors: Methodology and empirical data"*, In MICRO'03, San Diego, CA, USA, 2003.
- R. Bertran et al. *"Decomposable and responsive power models for multicore processors using performance counters"*, In ICS'10, Tsukuba, Ibaraki, Japan, June 2010
- R. Bertran et al. *"Accurate Energy Accounting for Shared Virtualized Environments using PMC-based Power Modeling Techniques"*, In GRID'10, Brussels, Belgium, October 2010
- R. Bertran et al. *"A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs"*. IEEE Transactions on Computers, 2012.
- R. Bertran et al. *"Energy accounting for shared virtualized environments under DVFS using pmc-based power models"*, Future Generation Computer Systems, pp457-468, 2012


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
QUESTIONS?