Generative and Multi-phase Learning for Computer Systems Optimization

Yi Ding, Nikita Mishra, Henry Hoffmann

THE UNIVERSITY OF CHICAGO

Computer Systems Optimization

- Optimizing modern computer systems requires tradeoffs:
  - Deliver reliable performance
  - Minimize energy consumption

Computer Systems Optimization

- Optimizing modern computer systems requires tradeoffs:
  - Deliver reliable performance
  - Minimize energy consumption

- Resource management via system configuration:
  - Resources have complex, non-linear effects on performance and energy
  - Resource interactions create local optima

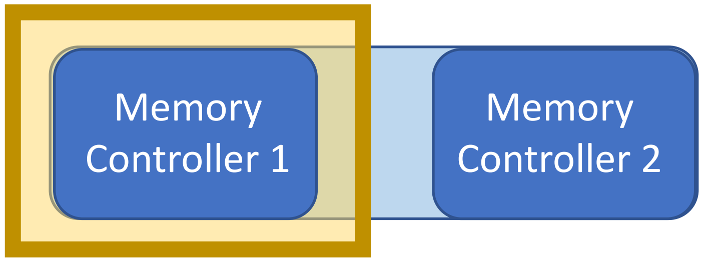
Computer Systems Optimization

- Optimizing modern computer systems requires tradeoffs:
  - Deliver reliable performance
  - Minimize energy consumption
- Resource management via system configuration:
  - Resources have complex, non-linear effects on performance and energy
  - Resource interactions create local optima
- How to find the optimal system configuration?

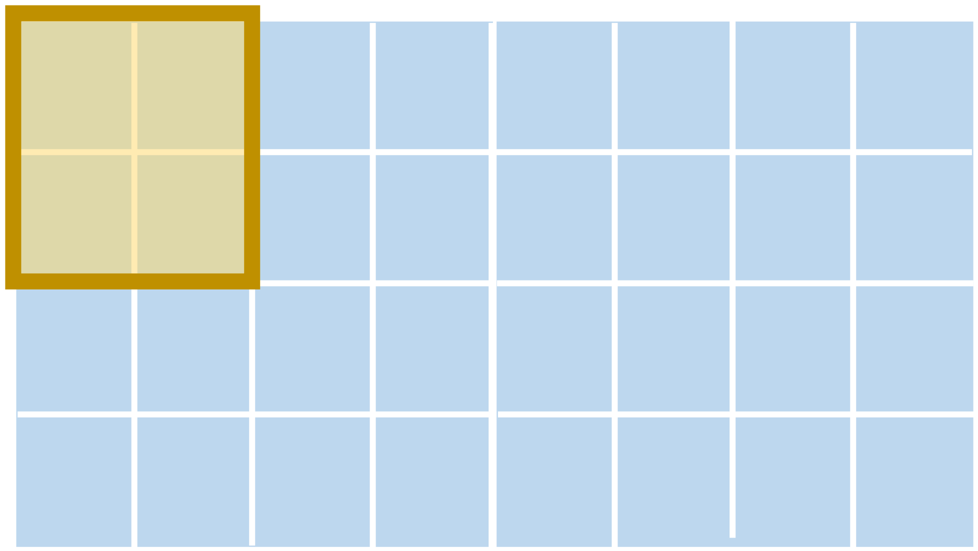
Example of a Configuration Space $\boldsymbol{C}$

$$\mathcal{C} \leftarrow \{\text{Core assignment}\} \times \{\text{Clock speed assignment}\} \times \{\text{Memory controller}\}$$

Clock Speed



Memory controller



Cores

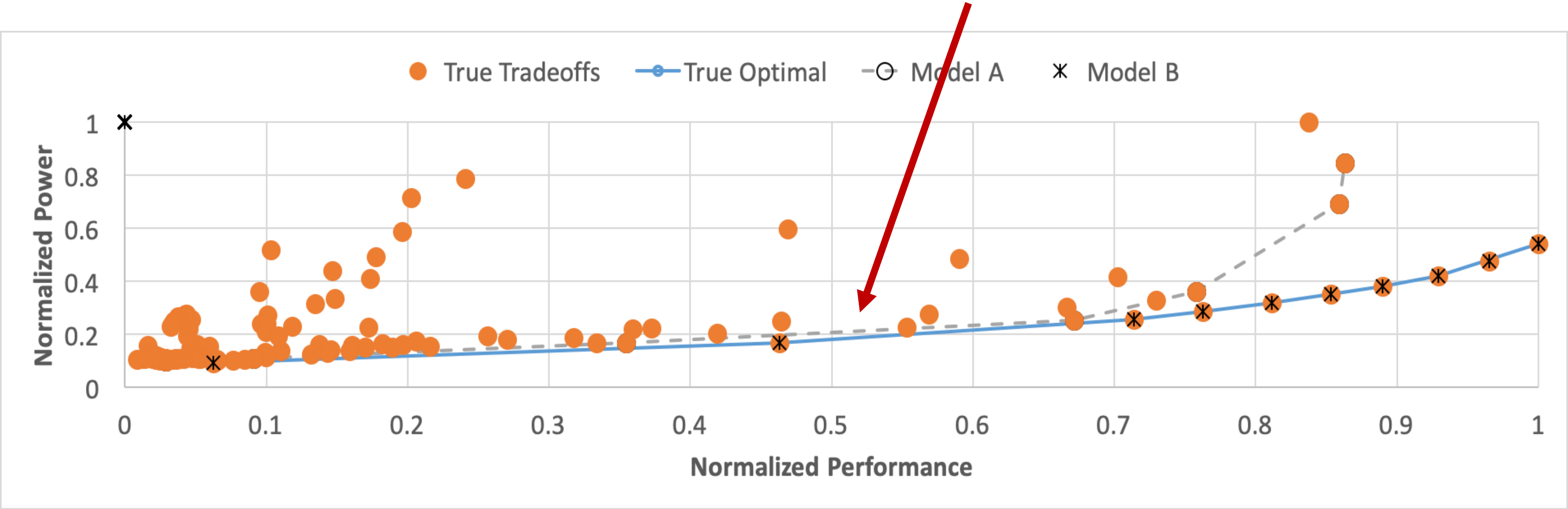
Machine Learning to the Rescue

Machine Learning to the Rescue

- But…
  - Scarce data: expensive collection, limited range behavior

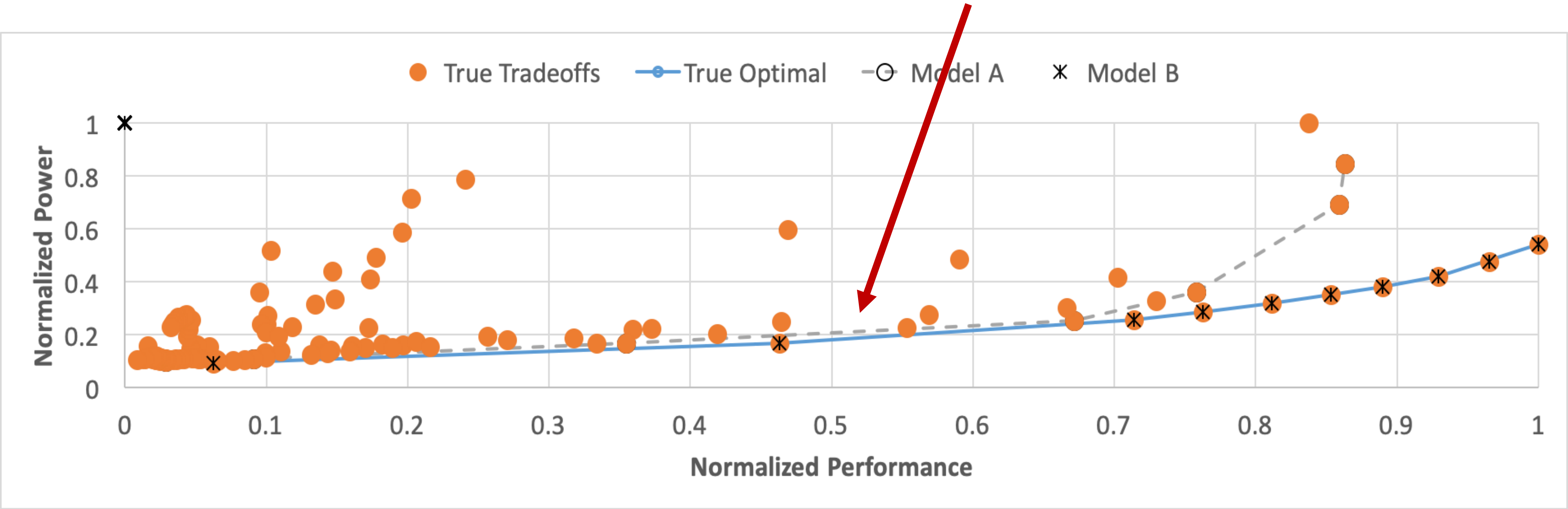
Machine Learning to the Resue

- But…
  - Scarce data: expensive collection, limited range behavior
  - Asymmetric benefits: only configs on **optimal frontier** useful



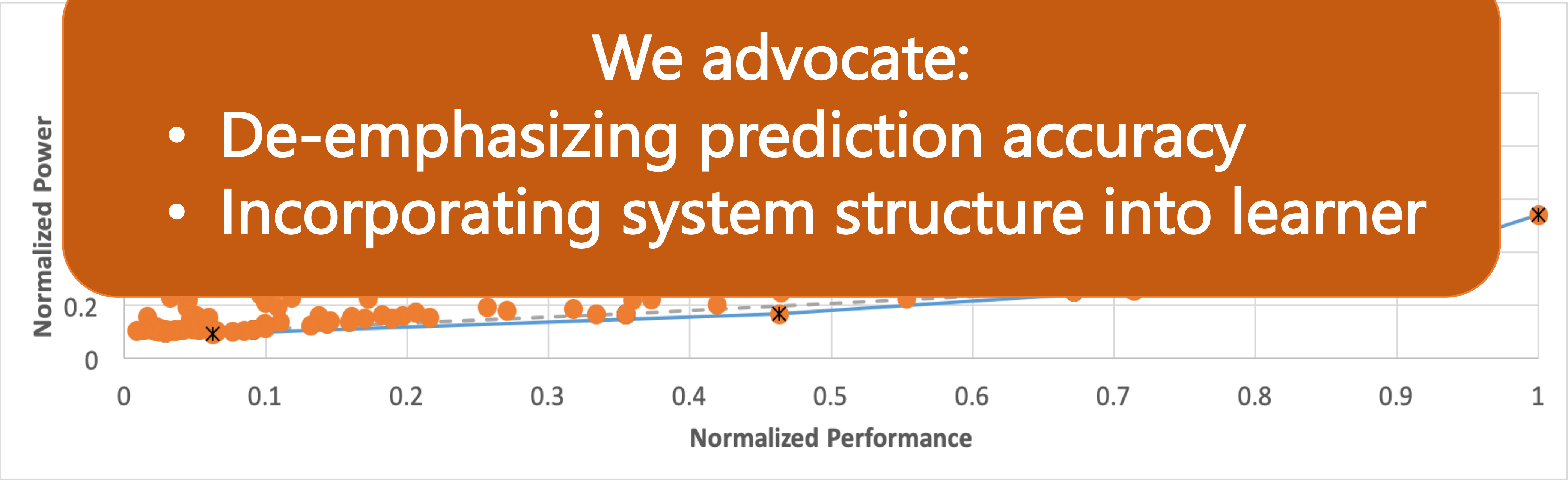
Machine Learning to the Resue

- But…
  - Scarce data: expensive collection, limited range behavior →Generative model
  - Asymmetric benefits: only configs on **optimal frontier** useful →Multi-phase sampling



Machine Learning to the Resue

- But…
  - Scarce data: expensive collection, limited range behavior →Generative model
  - Asymmetric benefits: only configs on **optimal frontier** useful →Multi-phase sampling



We advocate:
- De-emphasizing prediction accuracy
- Incorporating system structure into learner

Problem Formulation

Meet latency constraints with minimal energy via system configs

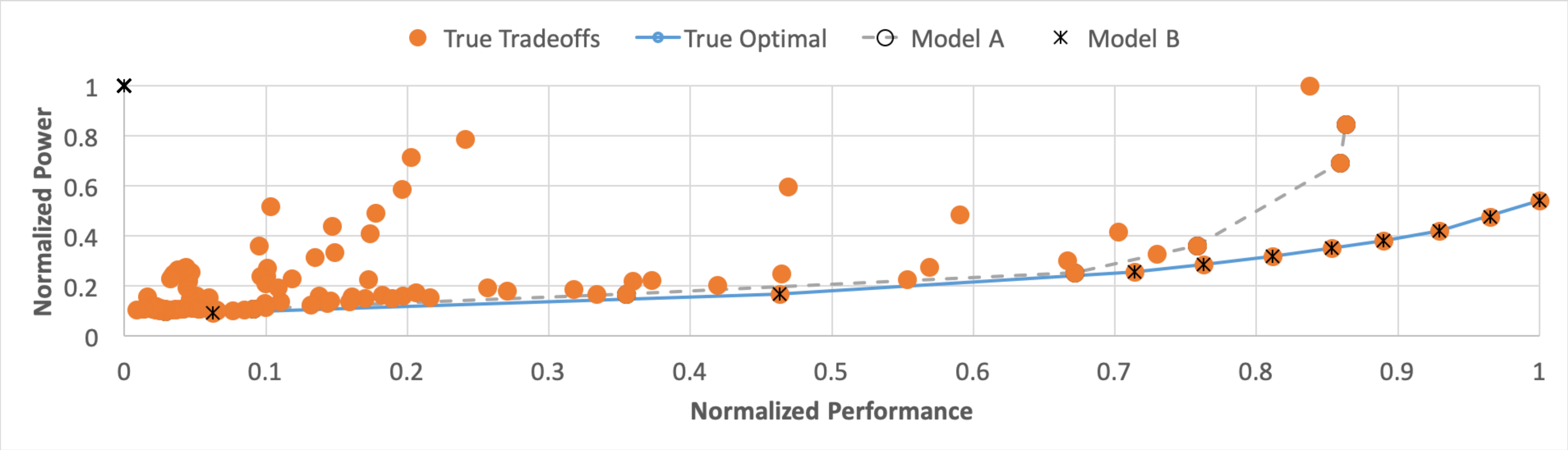
Config: an allocation of hardware resources to an application

Require the power and performance profile for applications

Expensive

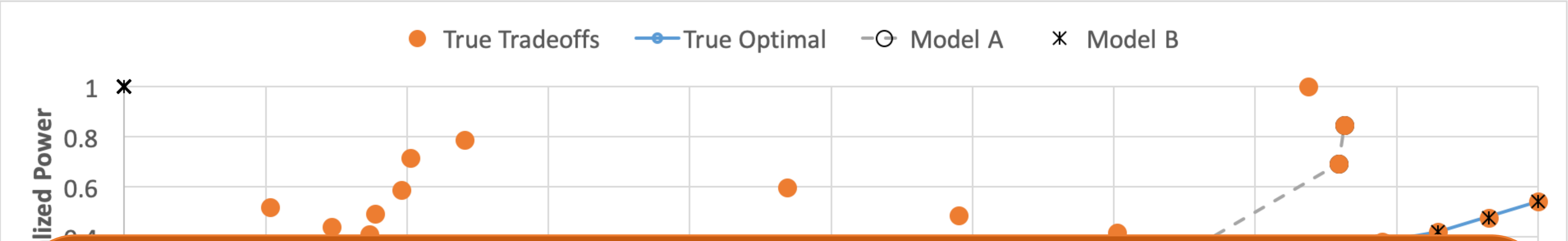
Learn to estimate these values

SRAD on ARM big.LITTLE system



	Model A	Model B
Optimal points	Just far enough	True data
Non-optimal points	True data	Very far
Goodness of fit	99%	0
Energy over optimal	22% ❌	0 ✅

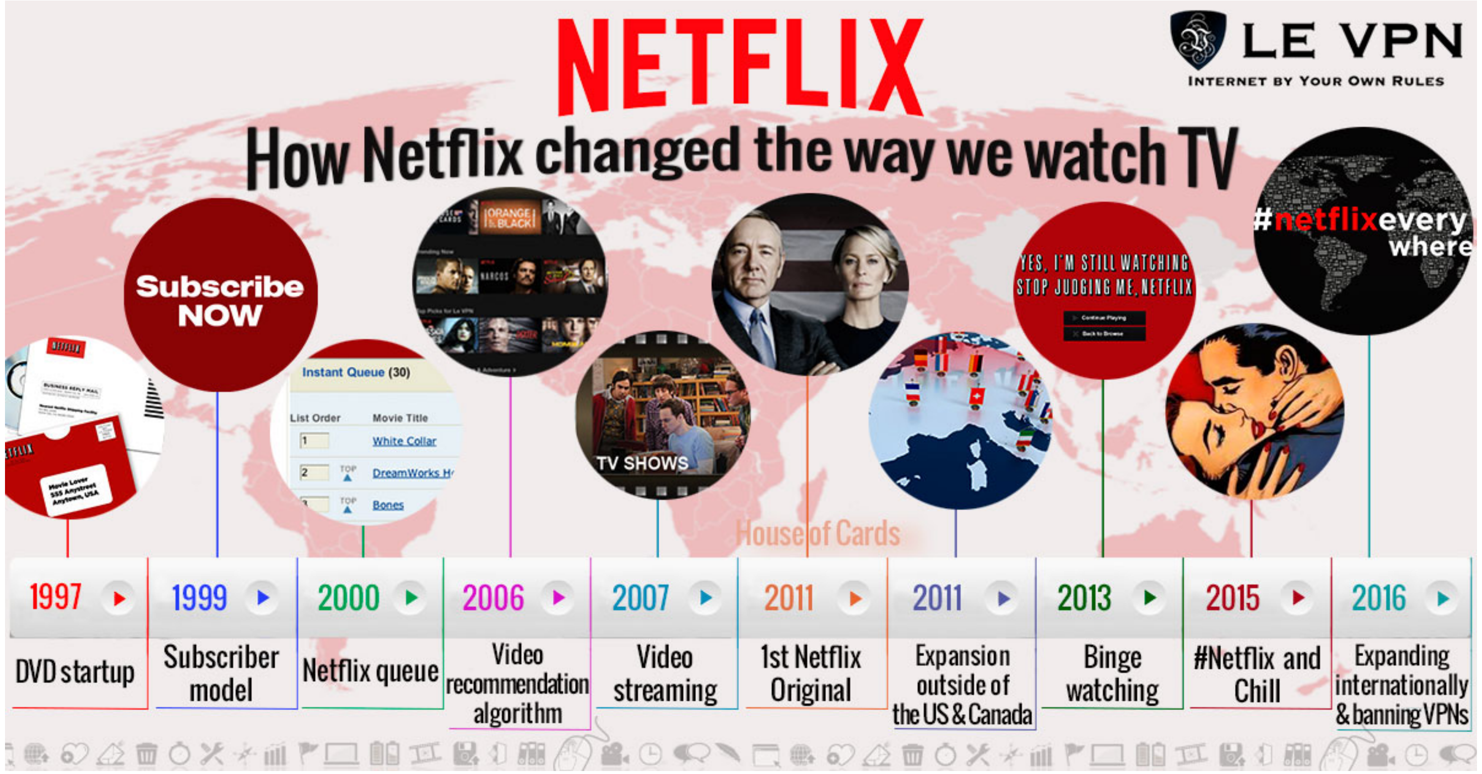
SRAD on ARM big.LITTLE system



Key Insight:
High accuracy != good system results

Optimal points	Just far enough	True data
Non-optimal points	True data	Just far enough
Goodness of fit	99%	0
Energy over optimal	22% ❌	0 ✅

Recommender Systems -> Learning by Examples



1. Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters. Christina Delimitrou and Christos Kozyrakis. (ASPLOS 2013)
2. Quasar: Resource-Efficient and QoS-Aware Cluster Management. Christina Delimitrou and Christos Kozyrakis (ASPLOS 2014)

https://www.le-vpn.com/netflix-10-innovations-changed-world/
https://www.thrillist.com/entertainment/nation/the-netflix-prize

An Analogy

Users

Movies

1	3	?	5	5			4
	5	4	4	2		1	2
2		4		1	2	3	
4	3		5		2	4	5
	4	2	4	2	4		1
2		4	1		3	2	3
3	4		2	2		5	3
1	3	3		2	4	2	

Ratings between 1 to 5

Unknown ratings

An Analogy

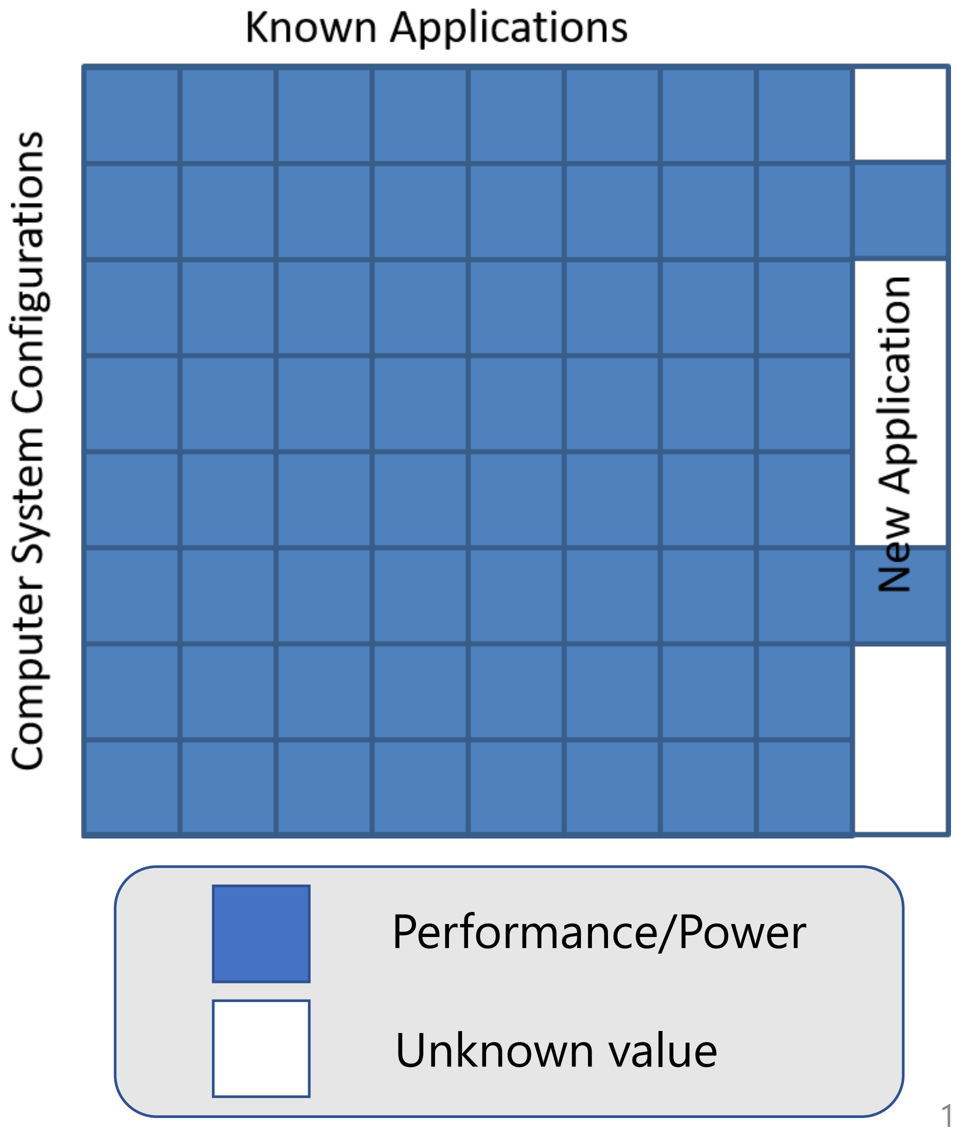
Users

Movies

1	3	?	5	5			4
	5	4	4	2		1	2
2		4		1	2	3	
4	3		5		2	4	5
	4	2	4	2	4		1
2		4	1		3	2	3
3	4		2	2		5	3
1	3	3		2	4	2	

Ratings between 1 to 5

Unknown ratings



Outline

- Motivation
- Methods
- Experimental Results
- Conclusion

Generating Data for Accuracy

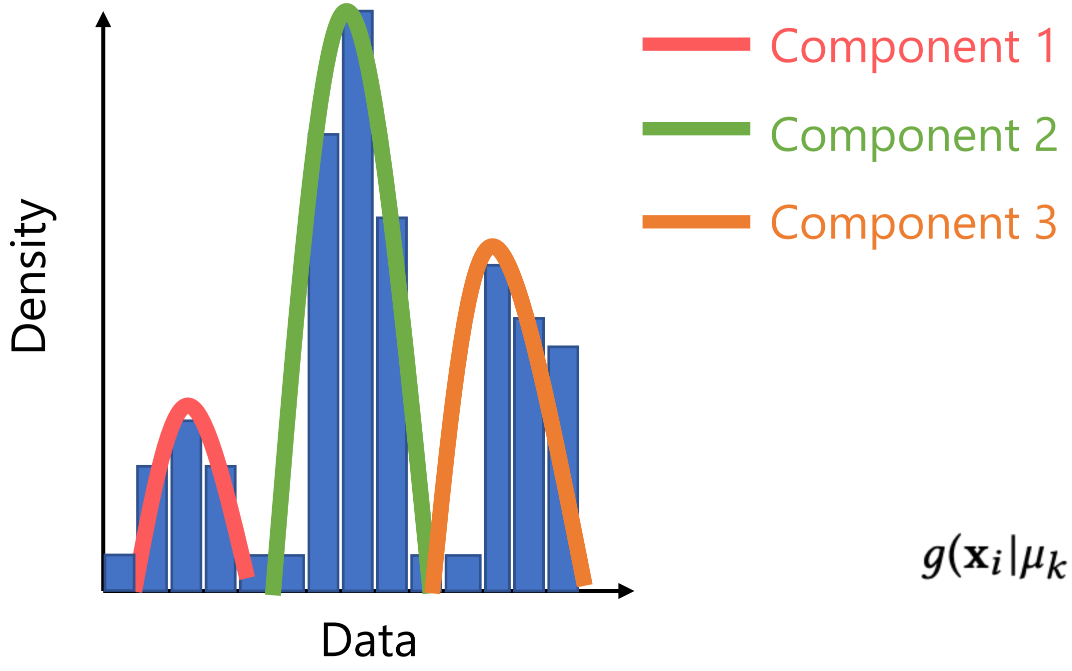
- Goal: different enough but still realistic to be plausible

Generating Data for Accuracy

- Goal: different enough but still realistic to be plausible
- How:
  - Random number generator →different but not plausible

Generating Data for Accuracy

- Goal: different enough but still realistic to be plausible
- How:
  - Random number generator →different but not plausible
  - Gaussian Mixture Model (GMM) →plausible but not different



K: number of components
x_i : data points, i=1,…,N
w_k: weight of k-th component

Probability that x_i belongs to k-th comp:

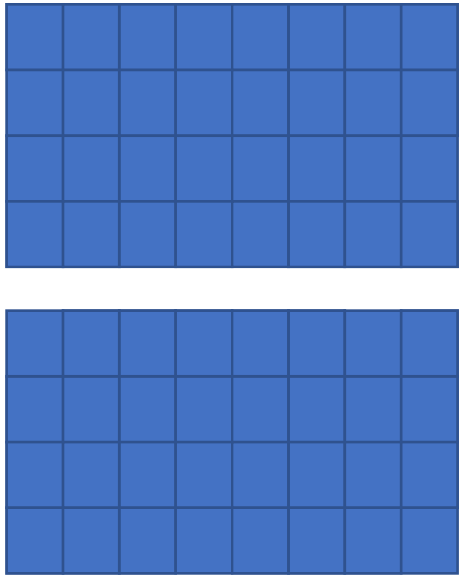
$$p(\mathbf{x}_i) = \sum_{k=1}^{K} w_k g(\mathbf{x}_i | \mu_k, \Sigma_k)$$

$$g(\mathbf{x}_i | \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \mu_k)^\top \Sigma_i^{-1} (\mathbf{x}_i - \mu_k) \right)$$

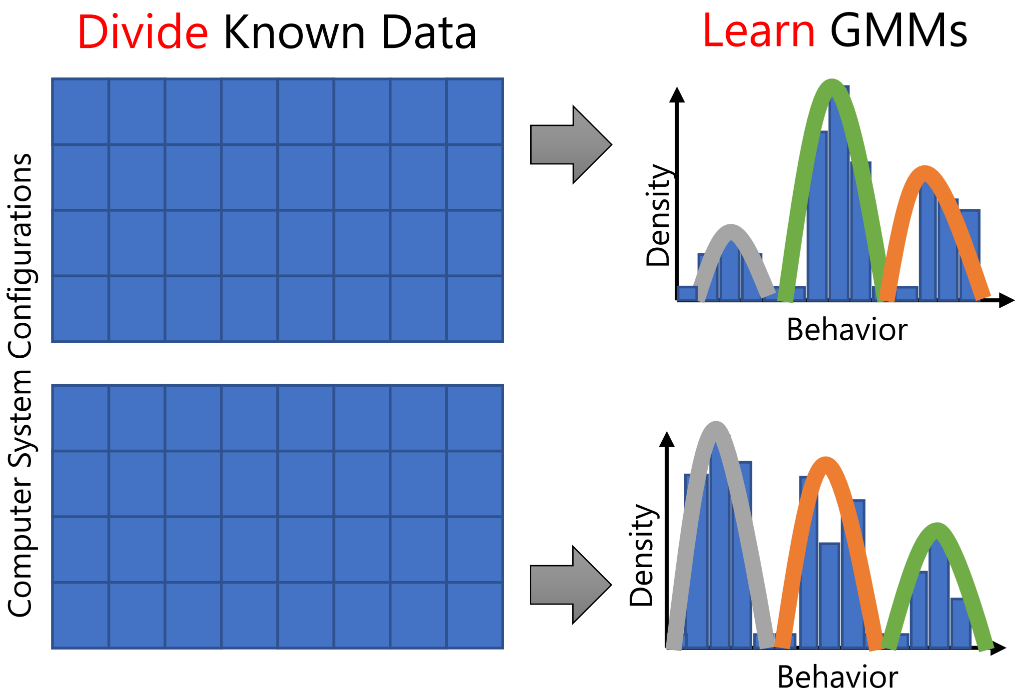
Generating Data with a GMM

Divide Known Data

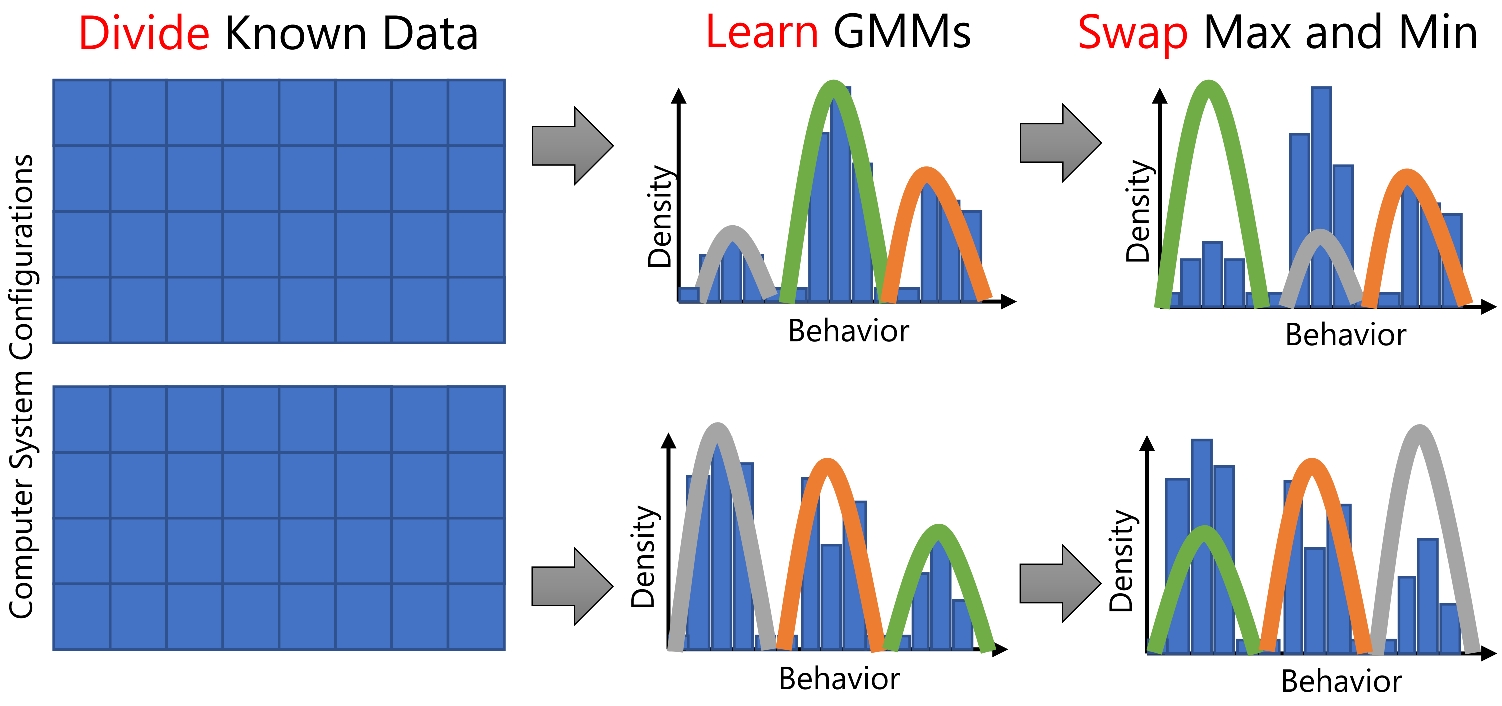
Computer System Configurations



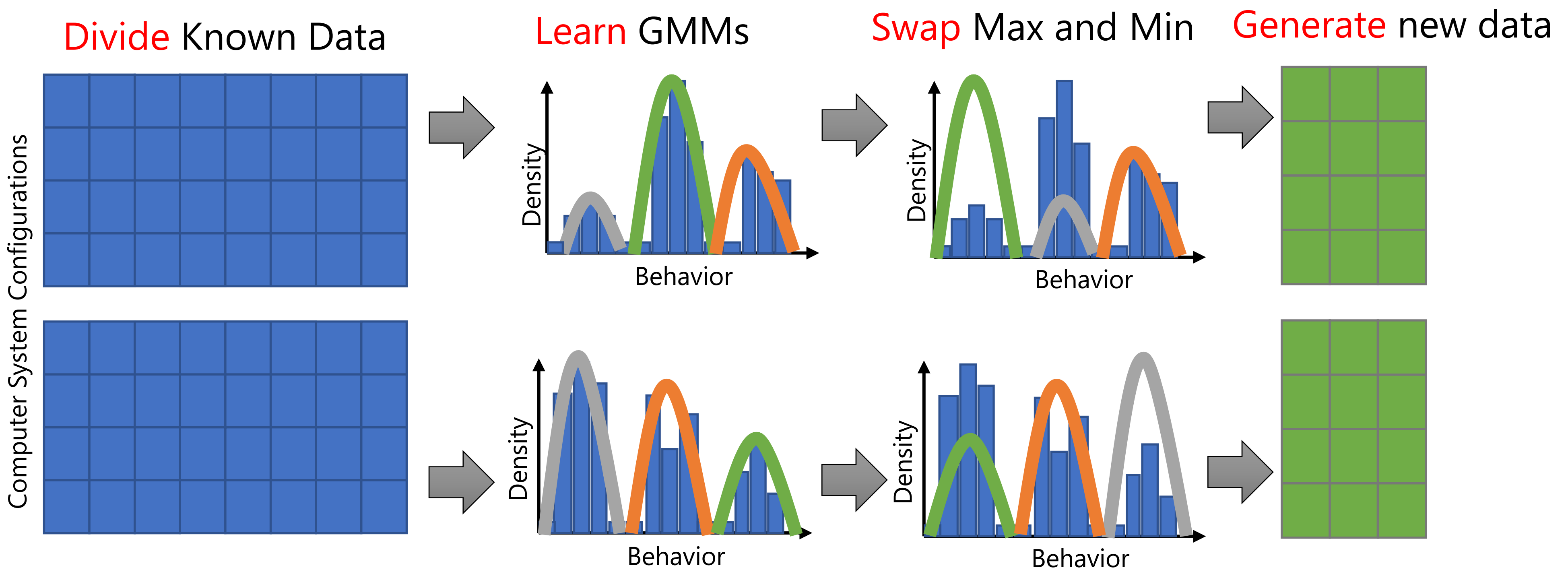
Generating Data with a GMM



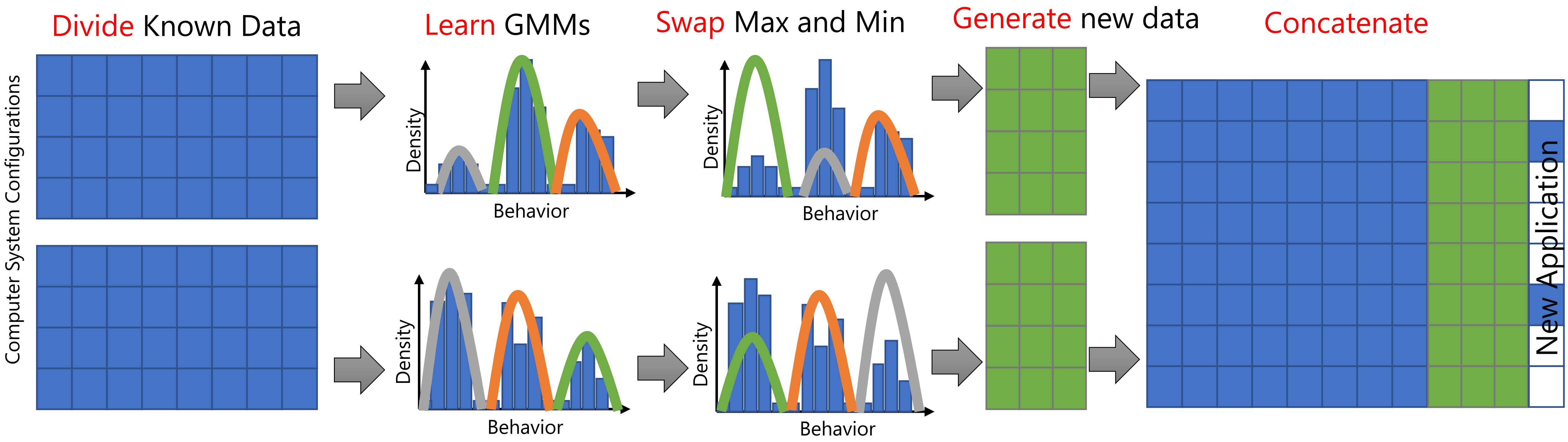
Generating Data with a GMM



Generating Data with a GMM



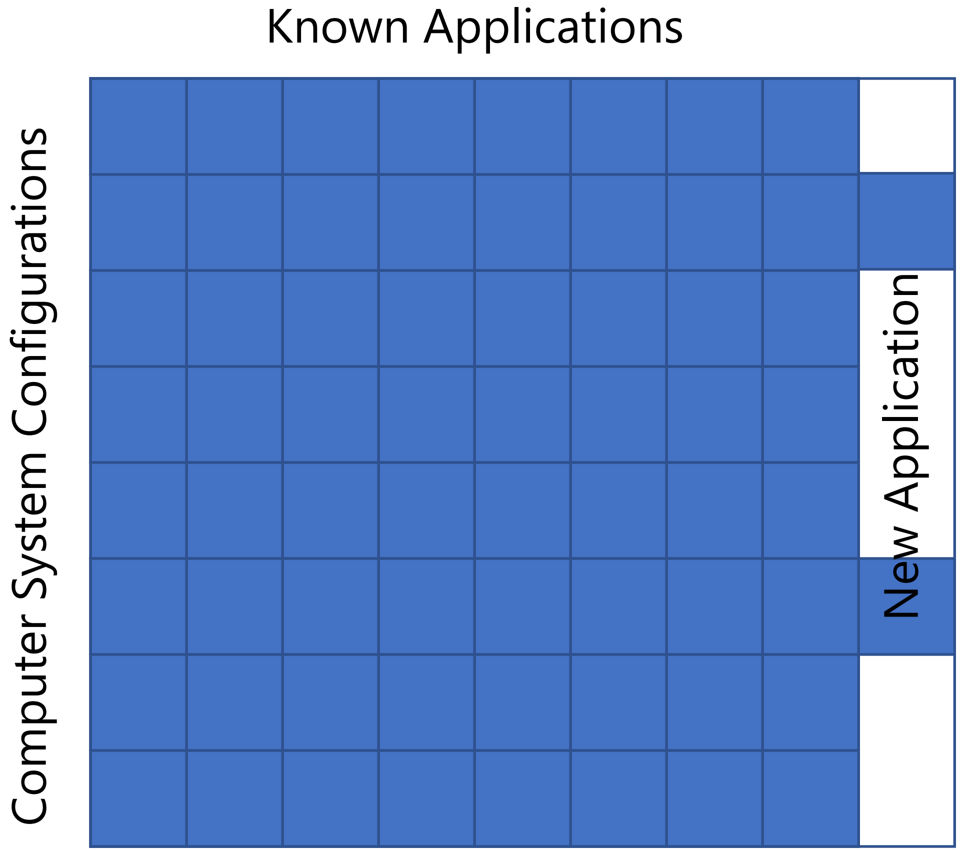
Generating Data with a GMM



Multi-phase Sampling

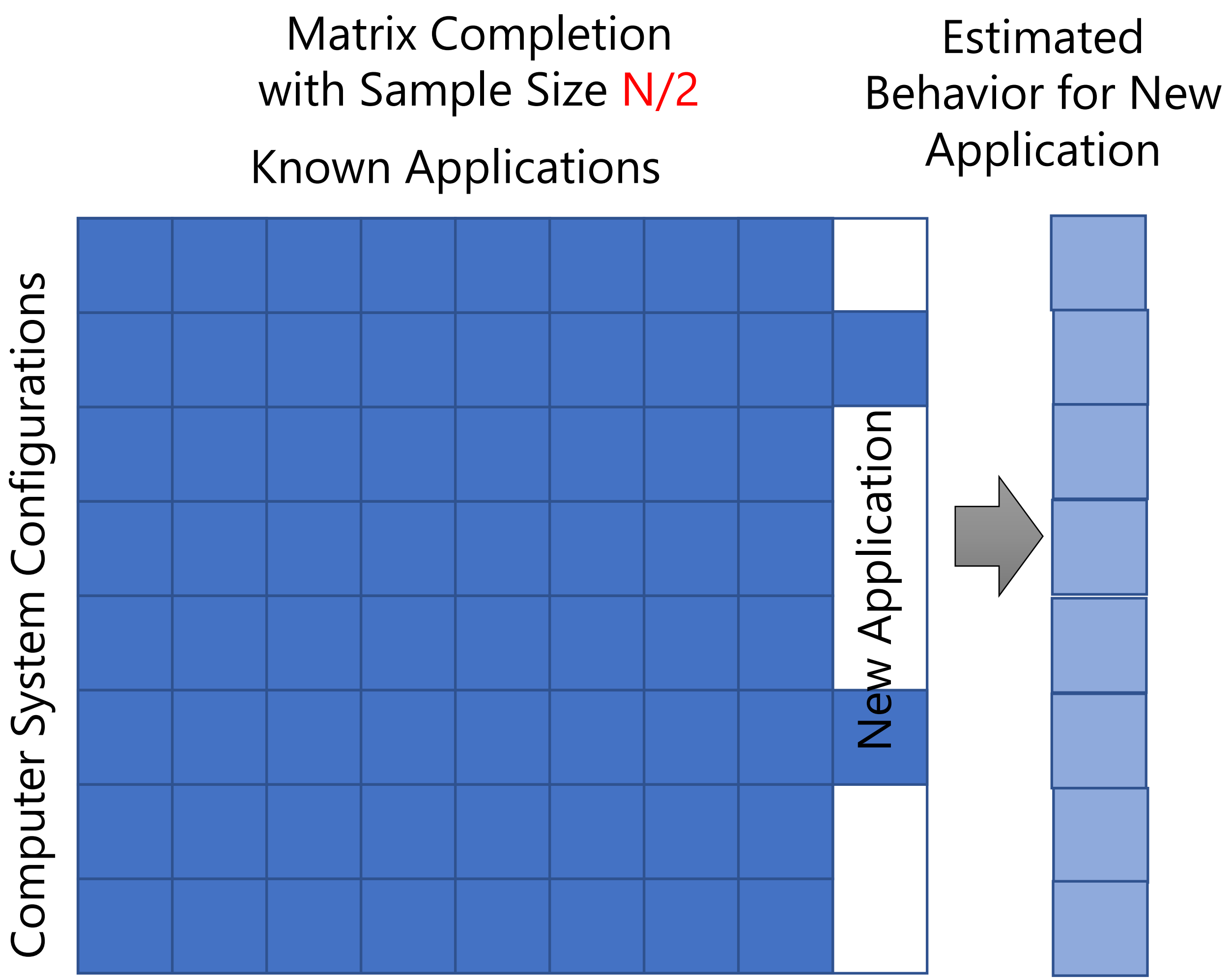
Input: Configuration-Application data matrix, Sampling budget N

Matrix Completion
with Sample Size N/2



Multi-phase Sampling

Input: Configuration-Application data matrix, Sampling budget N



Multi-phase Sampling

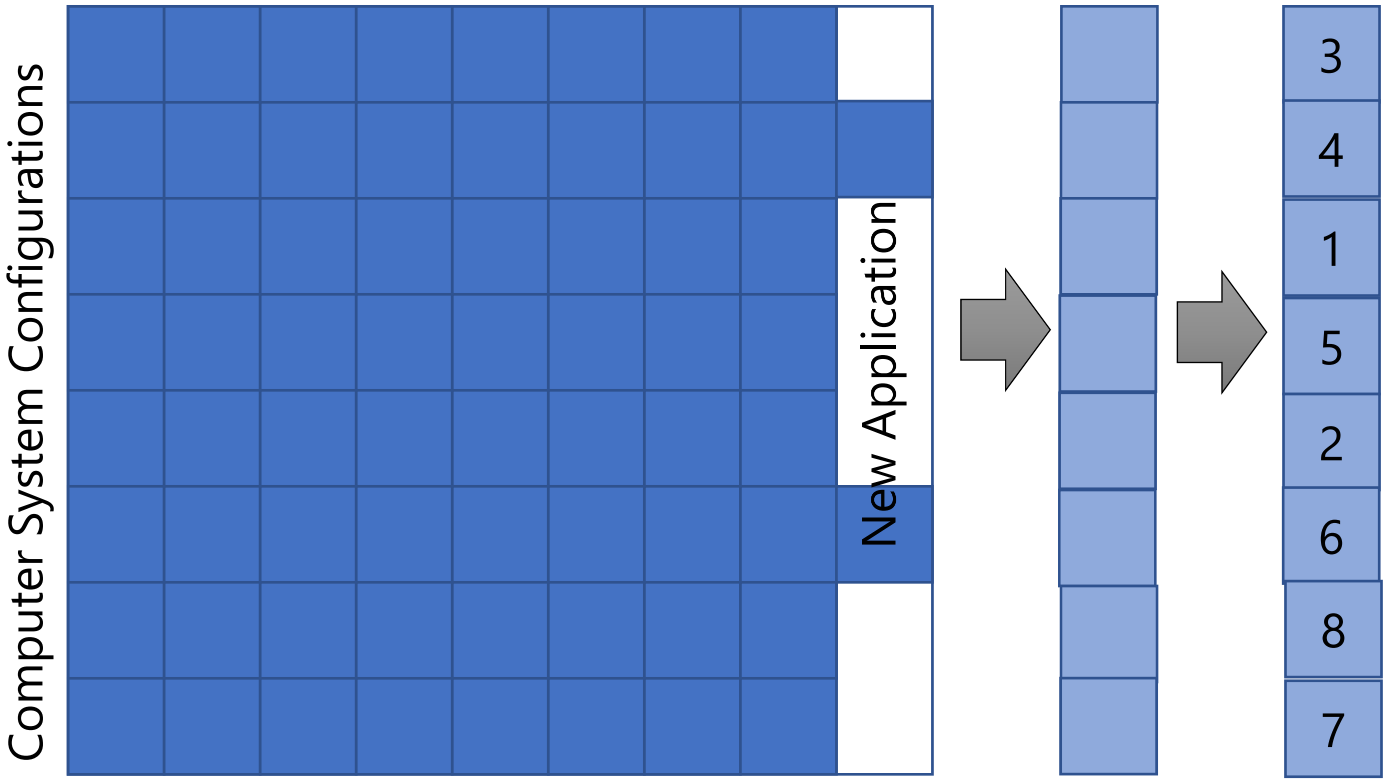
Input: Configuration-Application data matrix, Sampling budget N

Matrix Completion with Sample Size N/2

Known Applications

Estimated Behavior for New Application

Select N/2 Best Configs



$$\text{efficiency} = \frac{\text{estimated performance}}{\text{estimated power}}$$

Multi-phase Sampling

Input: Configuration-Application data matrix, Sampling budget N

Matrix Completion with Sample Size N/2

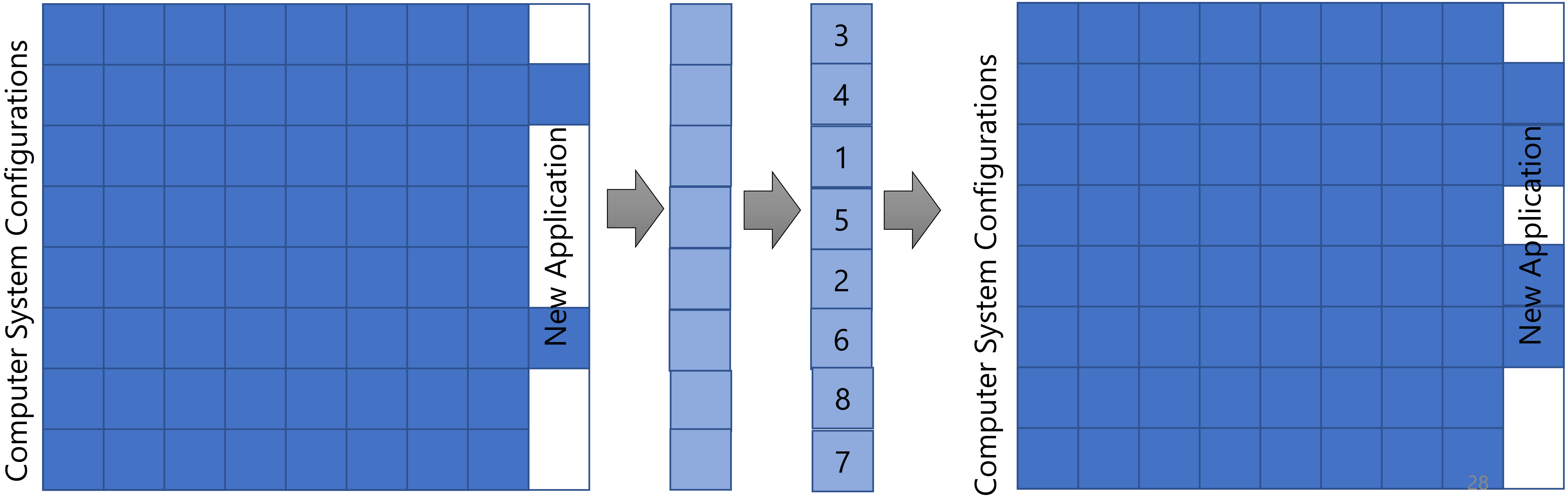
Known Applications

Estimated Behavior for New Application

Select N/2 Best Configs

Matrix Completion with N/2 original samples and N/2 estimated best configs

Known Applications

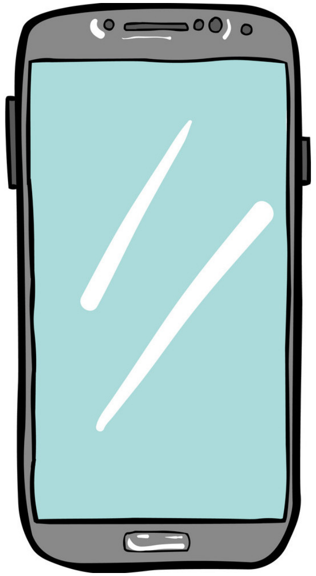


Outline

- Motivation
- Methods
- Experimental Results
- Conclusion

Experimental Setup

	Mobile	Server
System	Ubuntu 14.04	Linux 3.2.0 system
Architecture	ARM big.LITTLE	Intel Xeon E5-2690
# Applications	21	22
# Configurations	128	1024



Learning Models and Frameworks

Learning Models	Category
MCGD	MC
MCMF	MC
Nuclear	MC
WNNM	MC
HBM	Bayesian

First comprehensive study of matrix completion (MC) algorithms for systems optimization task

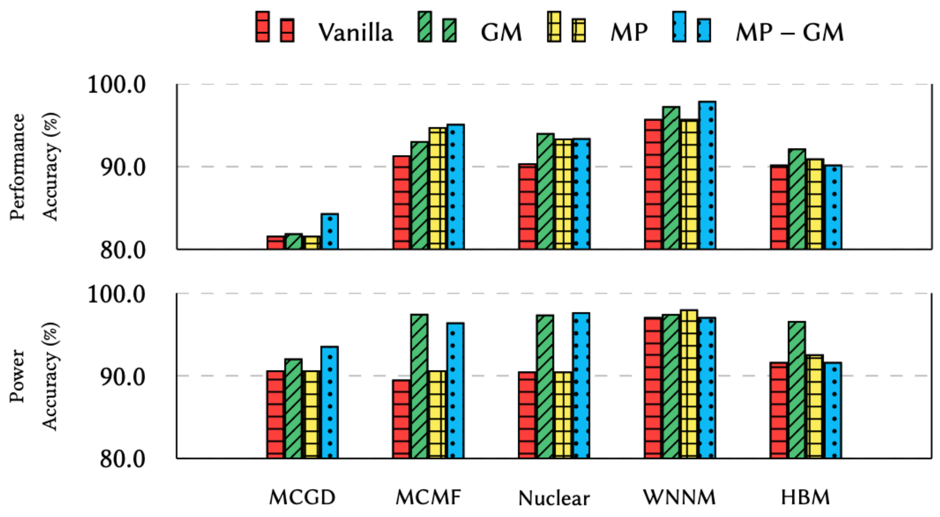
Learning Models and Frameworks

Learning Models	Category
MCGD	MC
MCMF	MC
Nuclear	MC
WNNM	MC
HBM	Bayesian

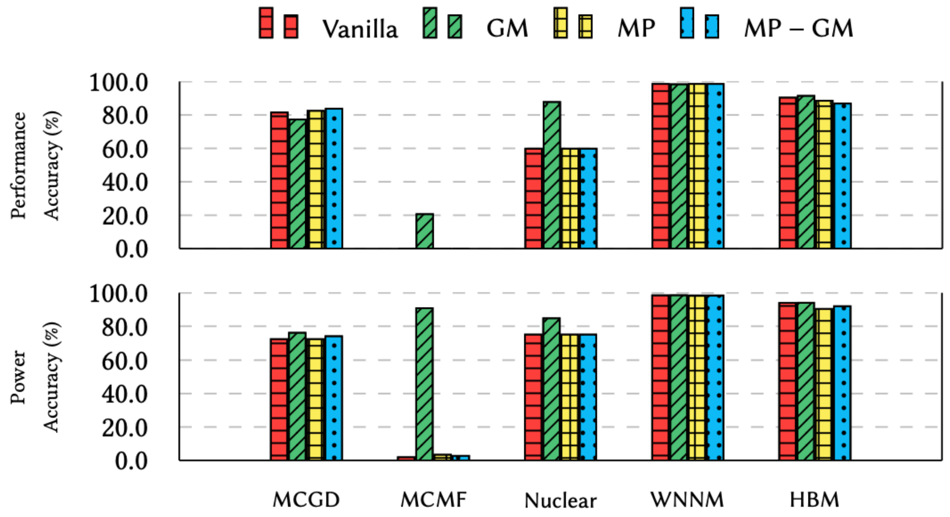
First comprehensive study of matrix completion (MC) algorithms for systems optimization task

Frameworks	Definitions
Vanilla	Basic learners
GM	Generative model
MP	Multi-phase sampling
MP-GM	Combine GM and MP

Improve Prediction Accuracy w/ GM



Mobile



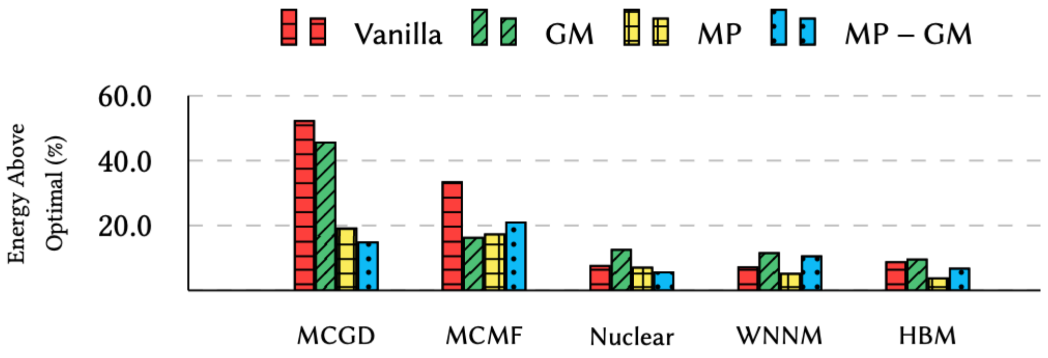
Server

High is Better

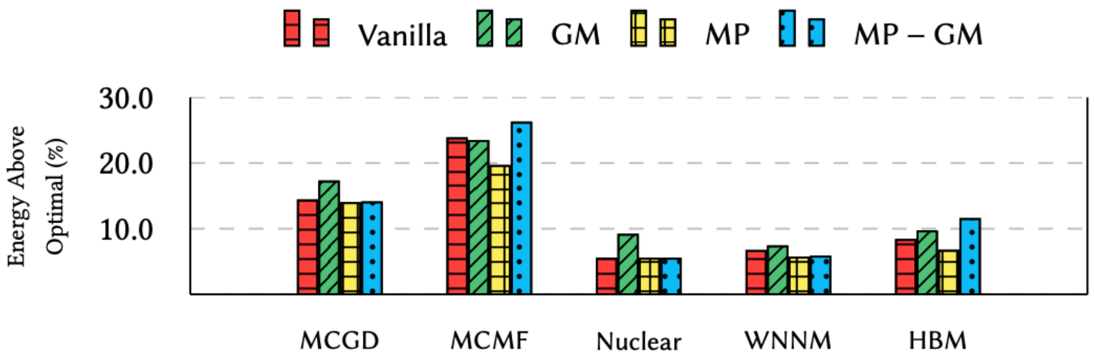
Average percentage points of accuracy improvement

		GM	MP	MP – GM
Mobile	Performance	1.8	1.4	2.3
	Power	4.3	0.6	3.4
Server	Performance	9.0	−0.2	−0.3
	Power	20.5	−0.4	0.1
Average		8.9	0.4	1.4

Improve Energy Savings w/ MP



Mobile



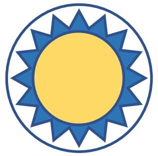
Server

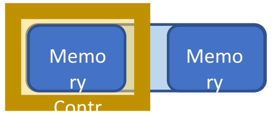
Lower is Better

Average energy improvement

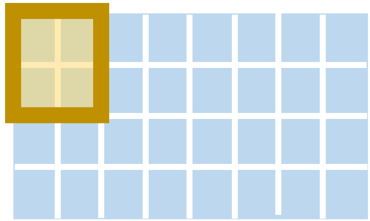
	GM	MP	MP – GM
Mobile	−14%	41%	22%
Server	−22%	11%	−6.5%

Conclusion

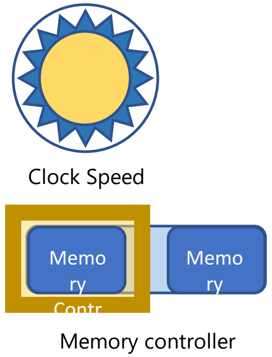

Clock Speed

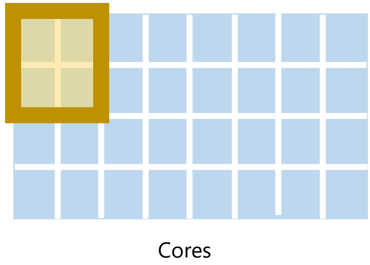


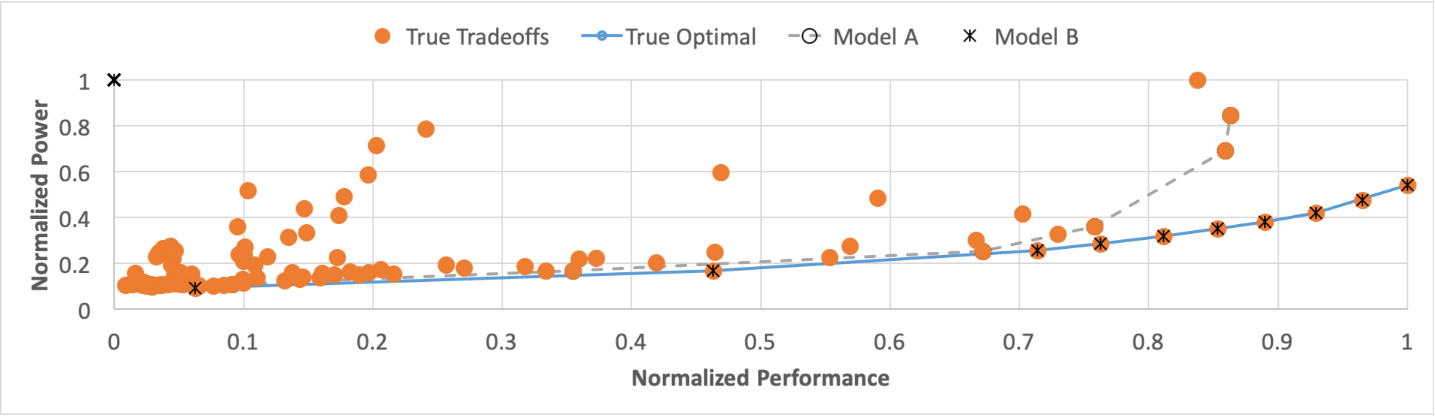
Memory controller


Cores

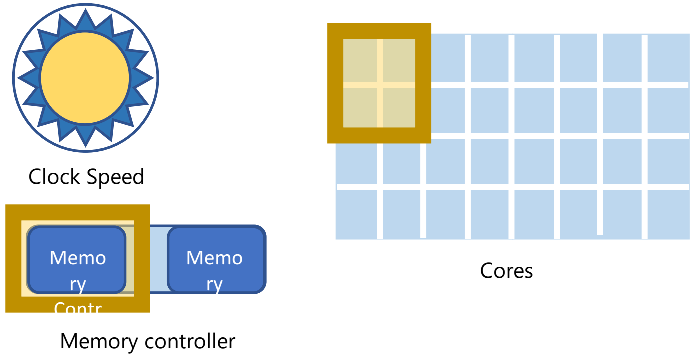
Conclusion


Clock Speed
Memo
ry
Memo
ry
Memory controller

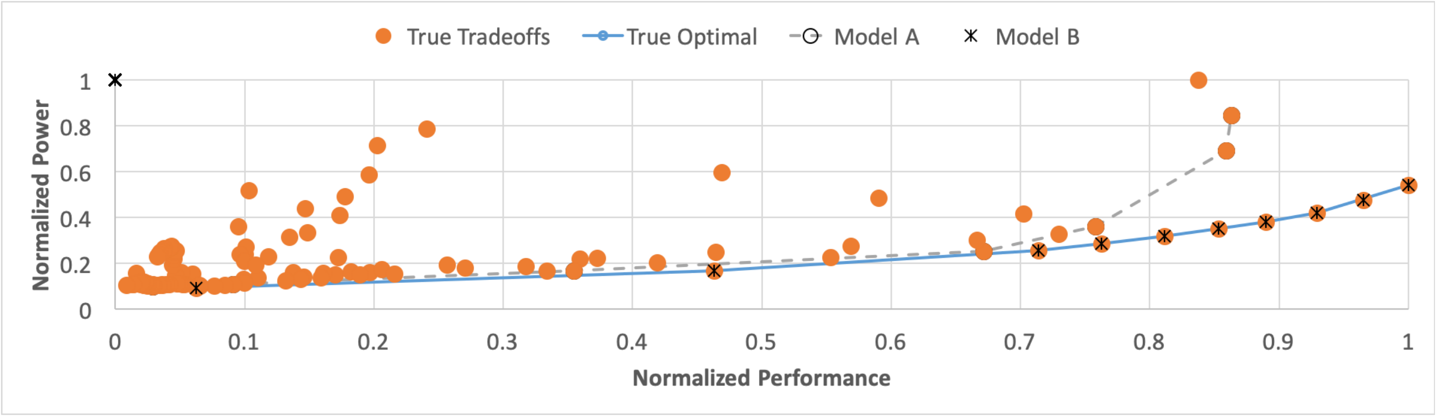

Cores


True Tradeoffs
True Optimal
Model A
Model B
Normalized Power
Normalized Performance
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9

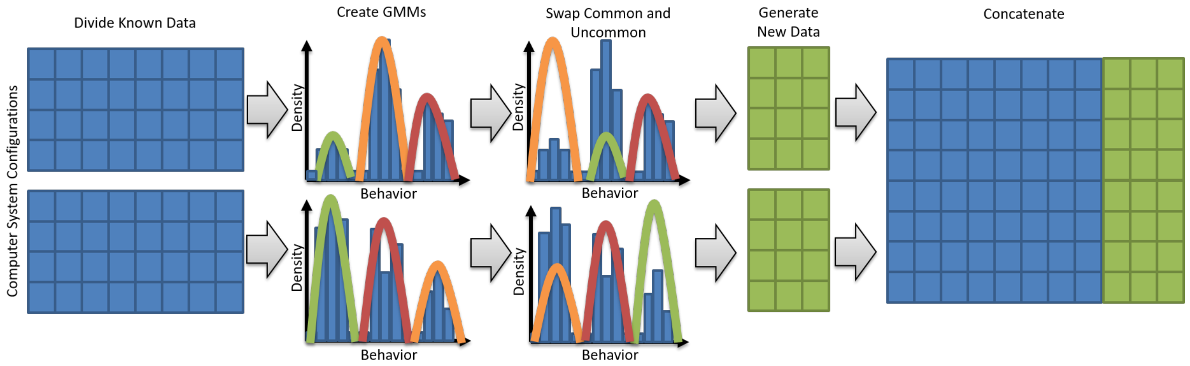
Conclusion



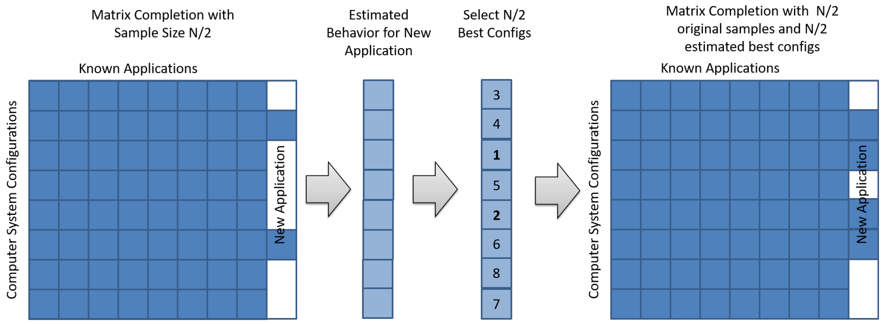
Clock Speed
Memo ry Contr
Memo ry
Memory controller
Cores



True Tradeoffs
True Optimal
Model A
Model B
Normalized Power
1
0.8
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Performance

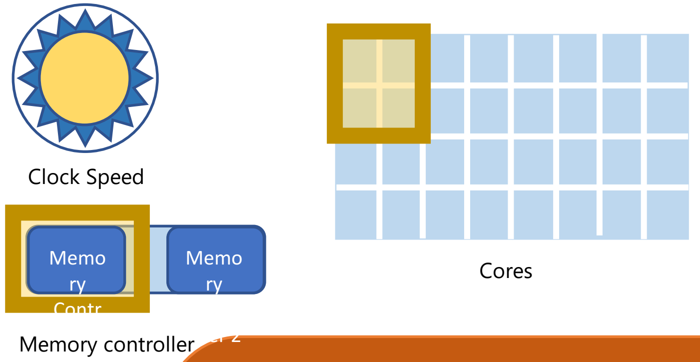


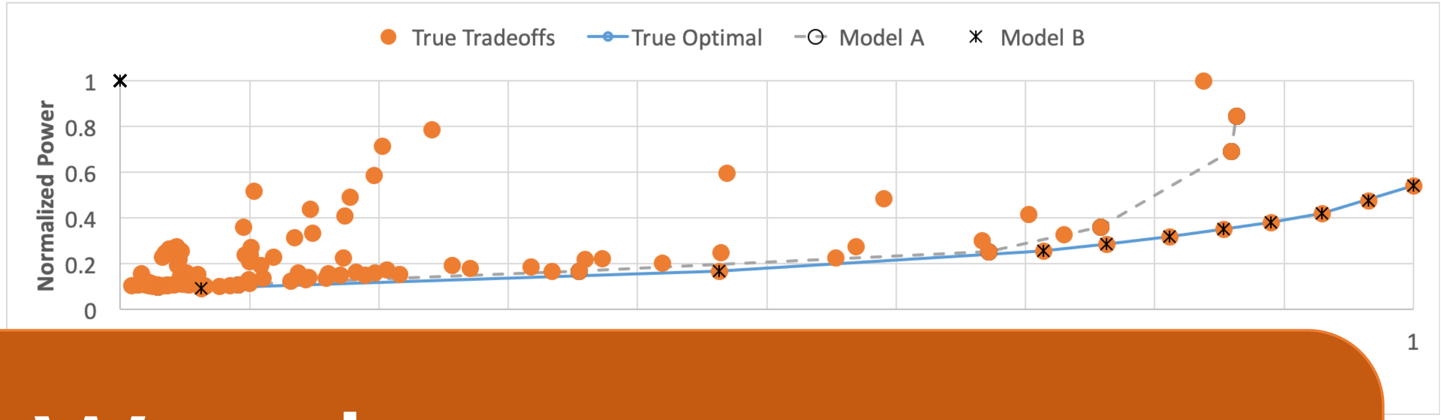
Divide Known Data
Computer System Configurations
Create GMMs
Density
Behavior
Swap Common and Uncommon
Generate New Data
Concatenate

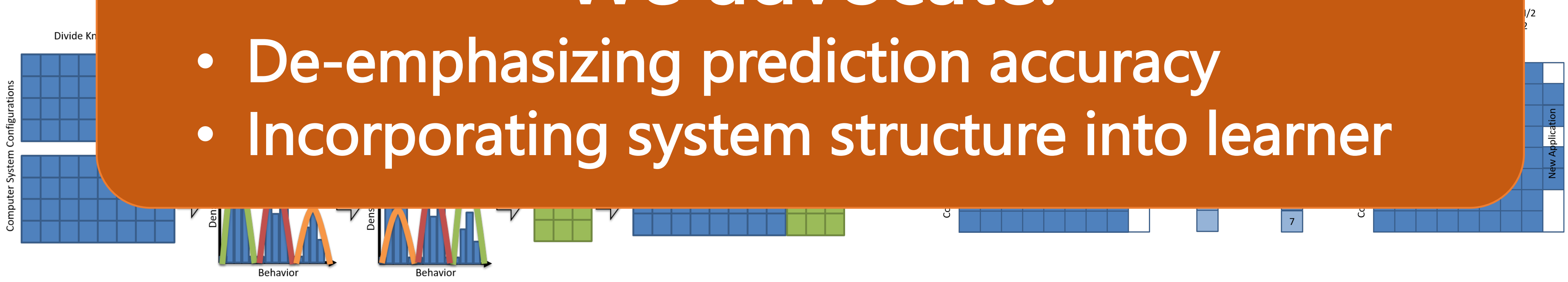


Matrix Completion with Sample Size N/2
Known Applications
Computer System Configurations
New Application
Estimated Behavior for New Application
Select N/2 Best Configs
3
4
1
5
2
6
8
7
Matrix Completion with N/2 original samples and N/2 estimated best configs
Known Applications

Conclusion







We advocate:

- De-emphasizing prediction accuracy
- Incorporating system structure into learner

Yi Ding, Nikita Mishra, and Henry Hoffmann. 2019. Generative and Multiphase Learning for Computer Systems Optimization. In The 46th Annual International Symposium on Computer Architecture (ISCA '19)