Inferred Models for Dynamic and Sparse Hardware-Software Spaces

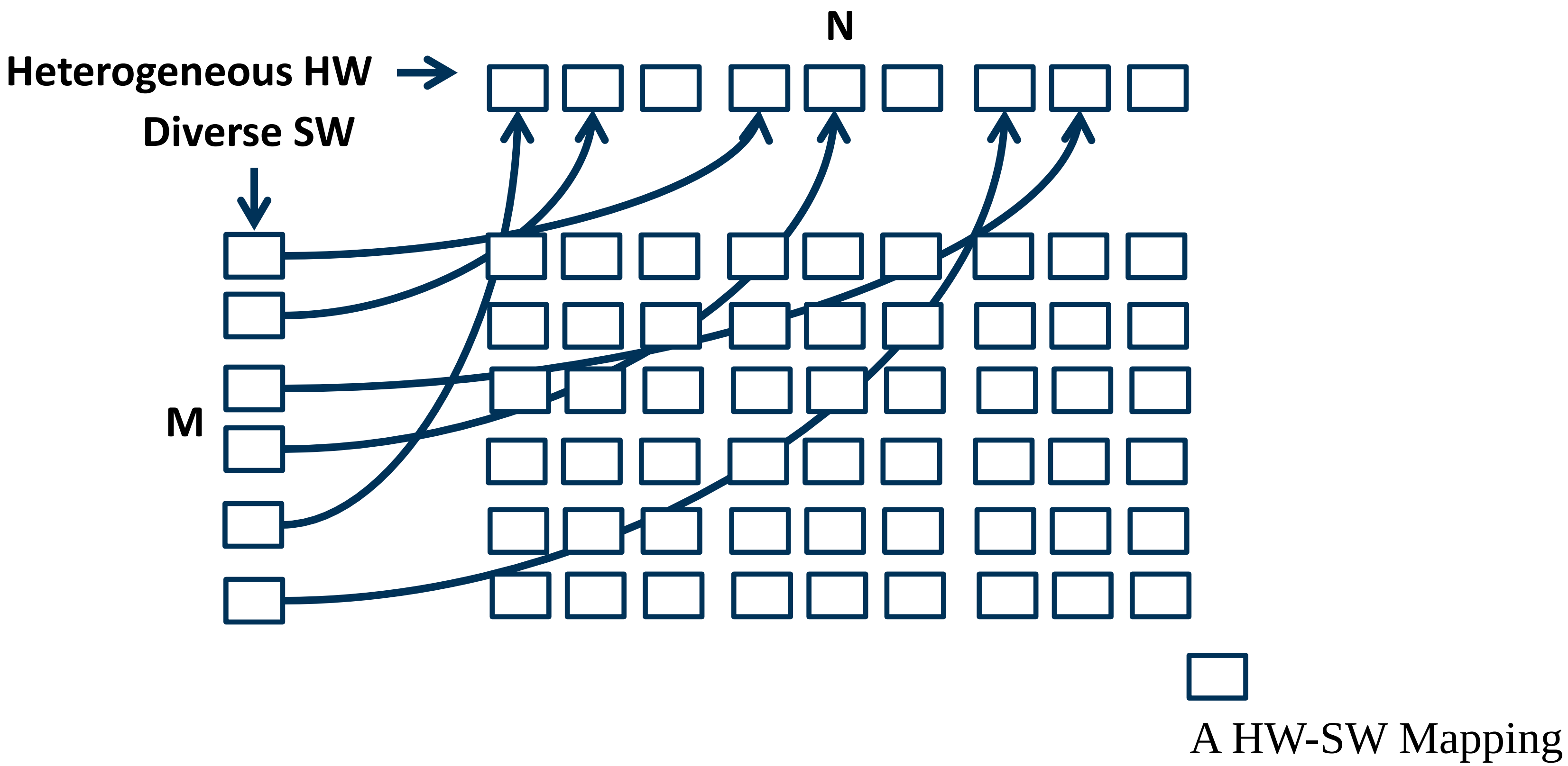
Weidan Wu, Benjamin C. Lee

Duke University

Trends in Management & Diversity

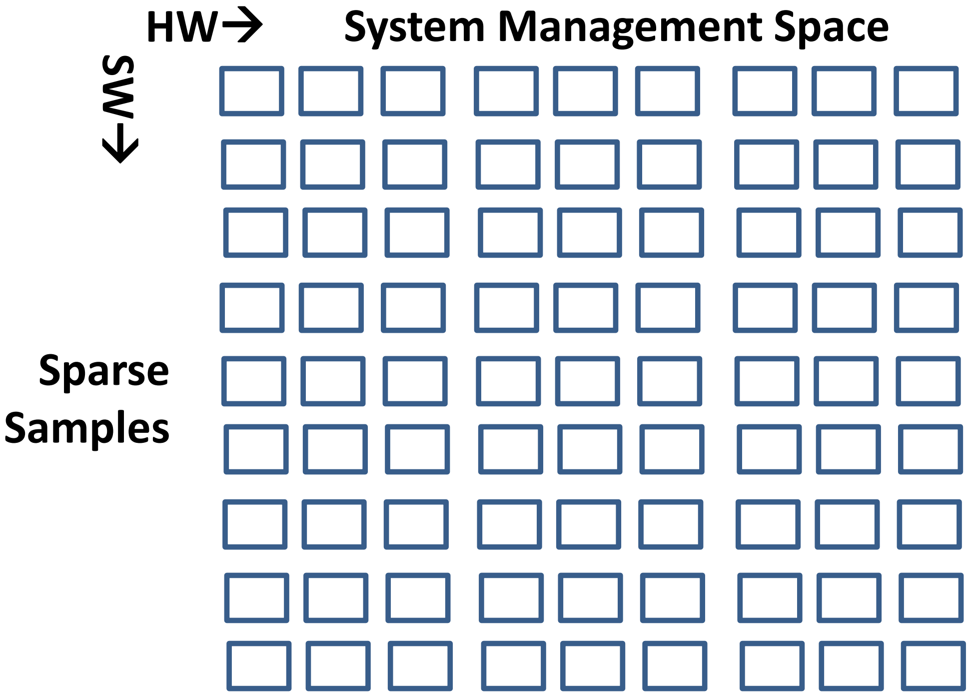
- Increasingly Sophisticated Management
  - Allocate resources, schedule applications, …
  - Understand HW-SW interactions
- Increasingly Diverse HW & SW
  - Heterogeneous cores, VMs, contention, …
  - Diverse clients, jobs, tasks, …

Mapping Software to Hardware



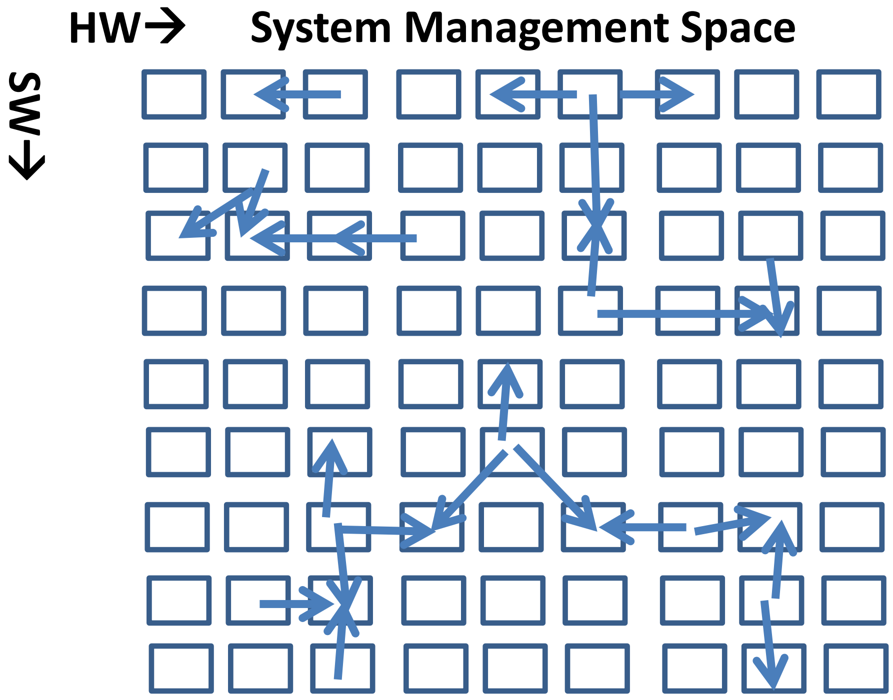
– Management space explosion (M x N)

Profilers Support Management



– But profile sparsity increases with diversity

Inference with Sparse Profiles

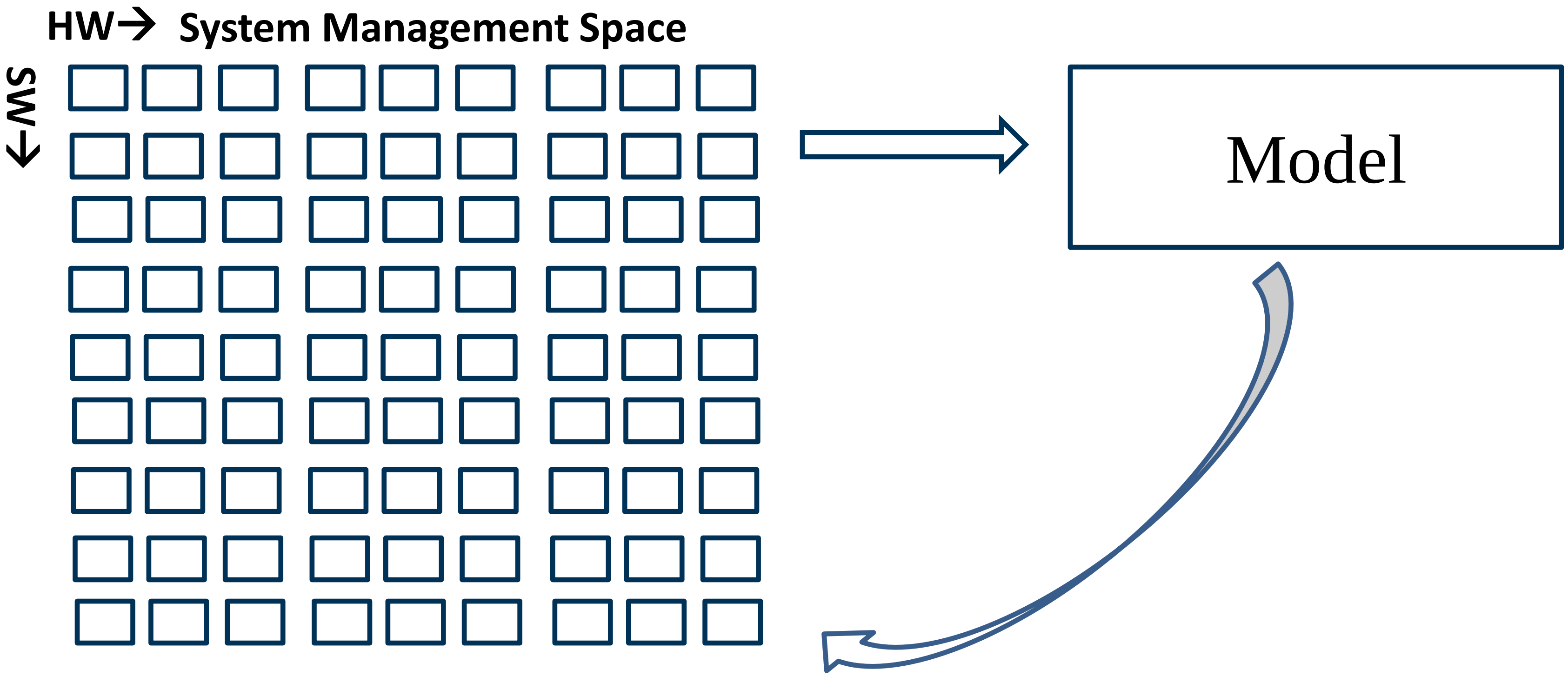


Outline

- Introduction
- **Inferred Performance Models**
- Generalized Models
- Specialized Models
- Conclusions

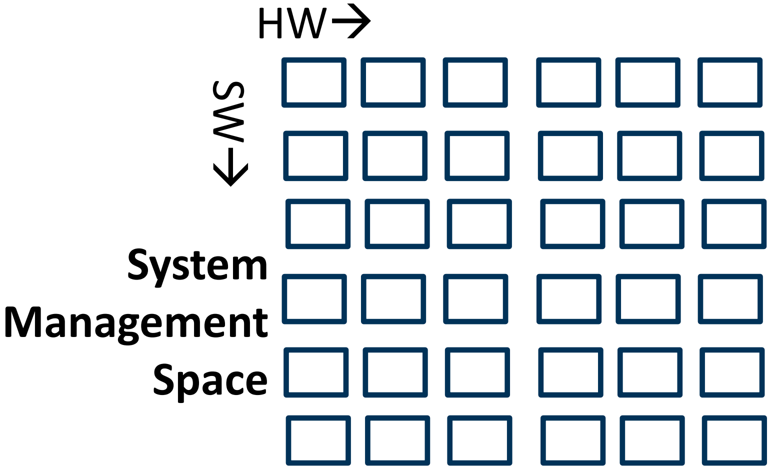
Inferred Performance Models

– Models, predictions support management



Integrated HW & SW Analysis

- Lays a foundation for run-time management
- Increases diversity among sparse samples
- Prior work separates HW & SW



New Challenges

- Larger space, greater sparsity
  - Data re-usability is critical
  - 30 parameters → 5E+15 points
- Less structured training data
  - SW profiles from arbitrary, real shards
  - HW profiles from defined, simulated design space

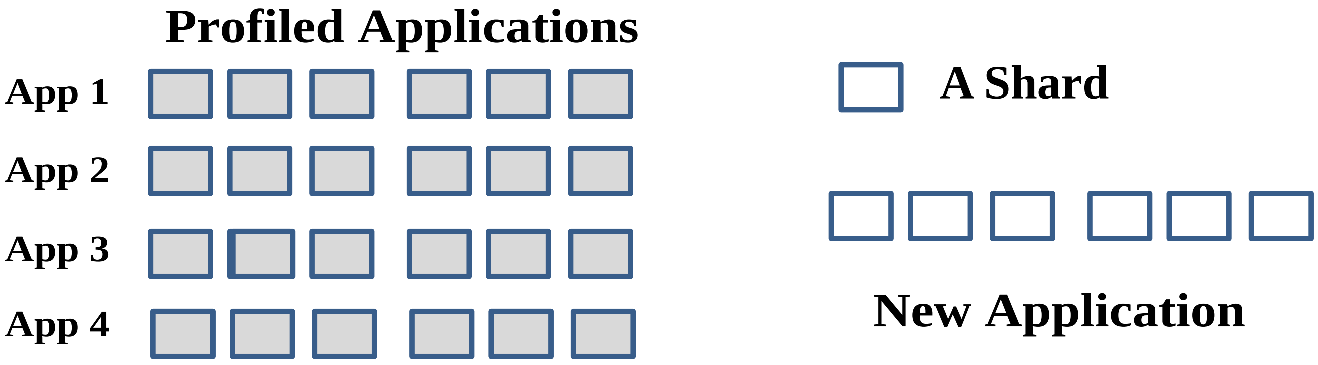
Principles and Strategies

- Enhance data re-usability
  - Shard-level profiles
  - Portable characteristics (μ-arch independent)

- Automate modeling
  - Genetic algorithm
  - Mitigate space explosion

Shard-level Profiles

- Shards: short dynamic instruction segments
- Re-use data among applications
  - New shards resemble existing ones
  - Monolithic profiles only useful when entire application resembles existing one

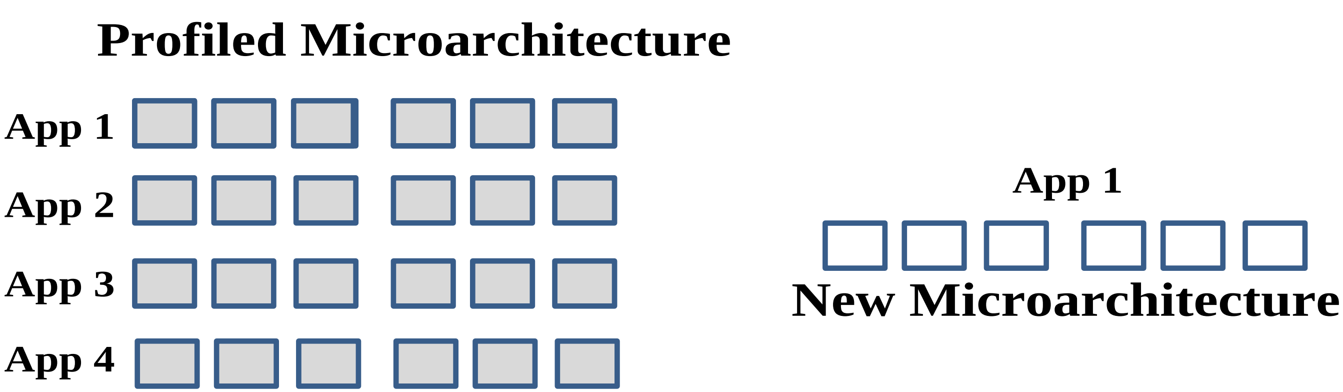


Shard-level Profiles

- Shards are sparse, randomly sampled segments of 10M instructions
- Shards from diverse applications complement each other, reducing profiling costs
- Shards expose intra-application diversity

Portable Characteristics

- Re-use data among microarchitectures
  - Microarchitecture-independent measures
  - Ex: instruction mix versus cache miss rate
  - Existing SW profiles relevant for new HW



Sharing Supports Inference

- Shards enhances data re-use across SW
- Portability enhances data re-use across HW
- Inferred models require less training data due to enhanced re-use

Statistical Inference

$$Y \quad = \quad X^T \quad \times \quad \beta \quad + \quad \epsilon$$

CPI   ALUs, cache size, … mem instr freq   regression coefficients

$$\begin{bmatrix} 1.21 \\ 0.89 \\ \vdots \\ 2.36 \\ 0.71 \end{bmatrix} \qquad \begin{bmatrix} 2 & 128\ldots & 0.39 \\ 4 & 64 \;\cdots & 0.27 \\ & \vdots \quad \vdots & \\ 6 & 256\ldots & 0.36 \end{bmatrix} \qquad \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

- X includes non-linear kernel transformations
  - Ex: log(cache size)
- X includes pair-wise interactions
  - Ex: ALU instructions, units

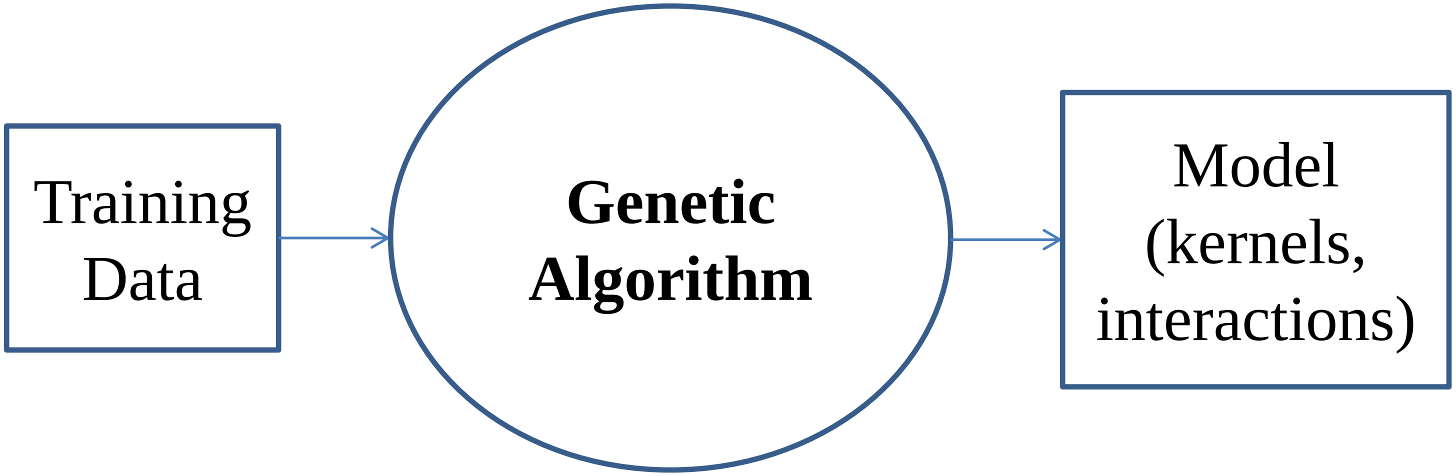
Space of Model Specifications

- Many kernel transformations
  - log, power, cubic spline, exponential, sqrt…
  - 30 parameters, 5 kernels → 5^{30} model specs

- Many parameter interactions
  - Hardware and software interact
  - $\binom{30}{2} = 435$ pairwise interactions → 2^{435} specs

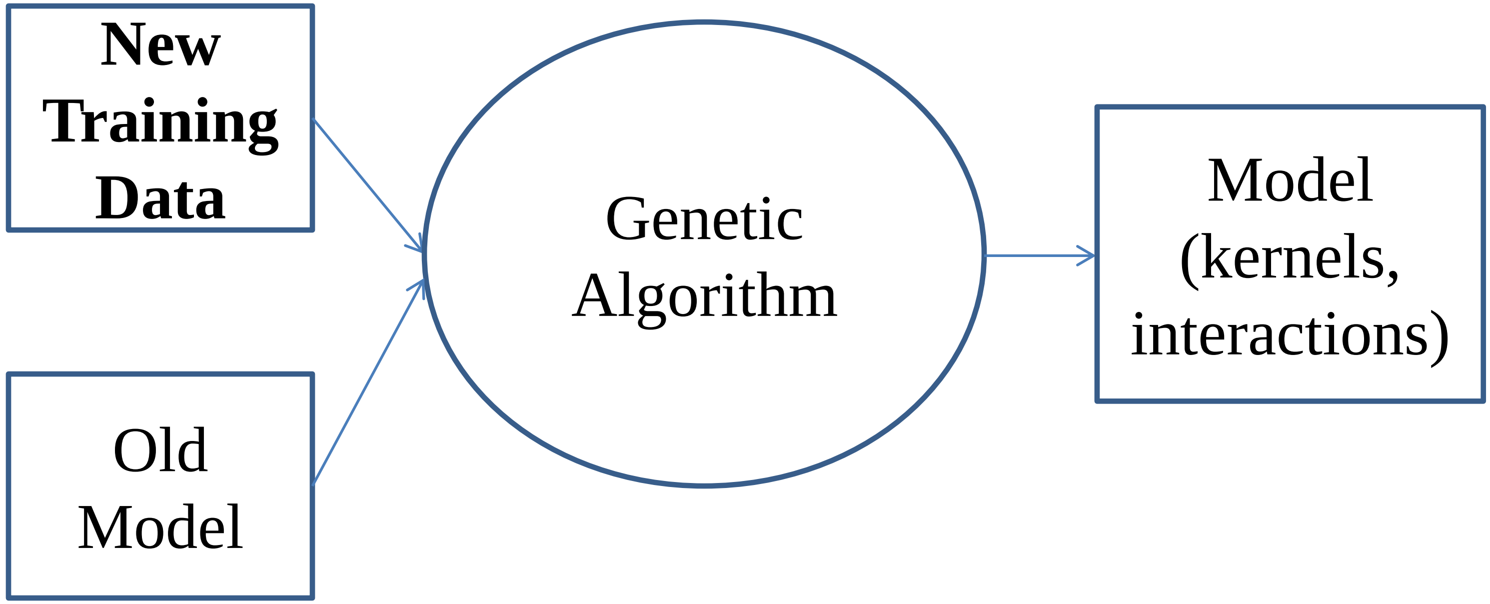
Duke Architecture

Automatic Model Construction



– Model specification encoded as genes
– Mutation, crossover search models
– Selection evolves model toward higher accuracy

Automatic Model Updates



– New data updates model specification
– Algorithm changes kernels, interactions, fit

Outline

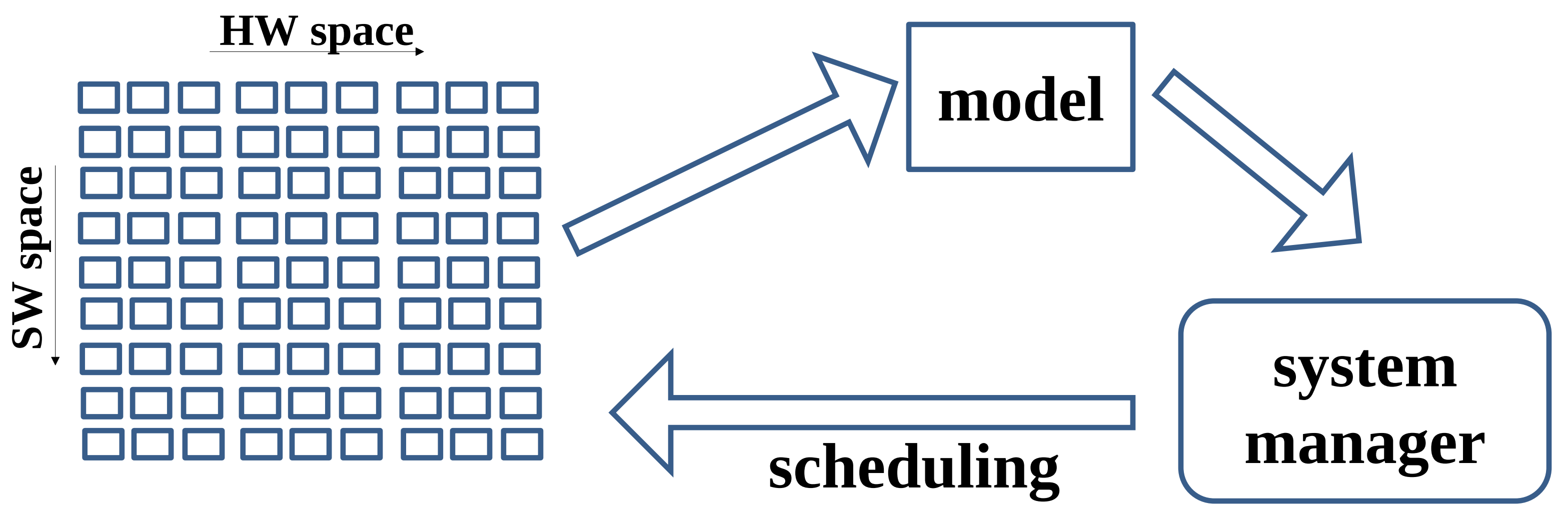
- Introduction
- Inferred Performance Models
- **Generalized Models**
- Specialized Models
- Conclusions

Generalized Models

- Diverse SW as applications enter/leave system
  – Ex: democratized datacenter computing
- Heterogeneous HW as architectures tuned
  – Ex: big/small cores, VMs, contention, …
- Profiled data collected as SW runs on HW
- Models update to accommodate dynamics

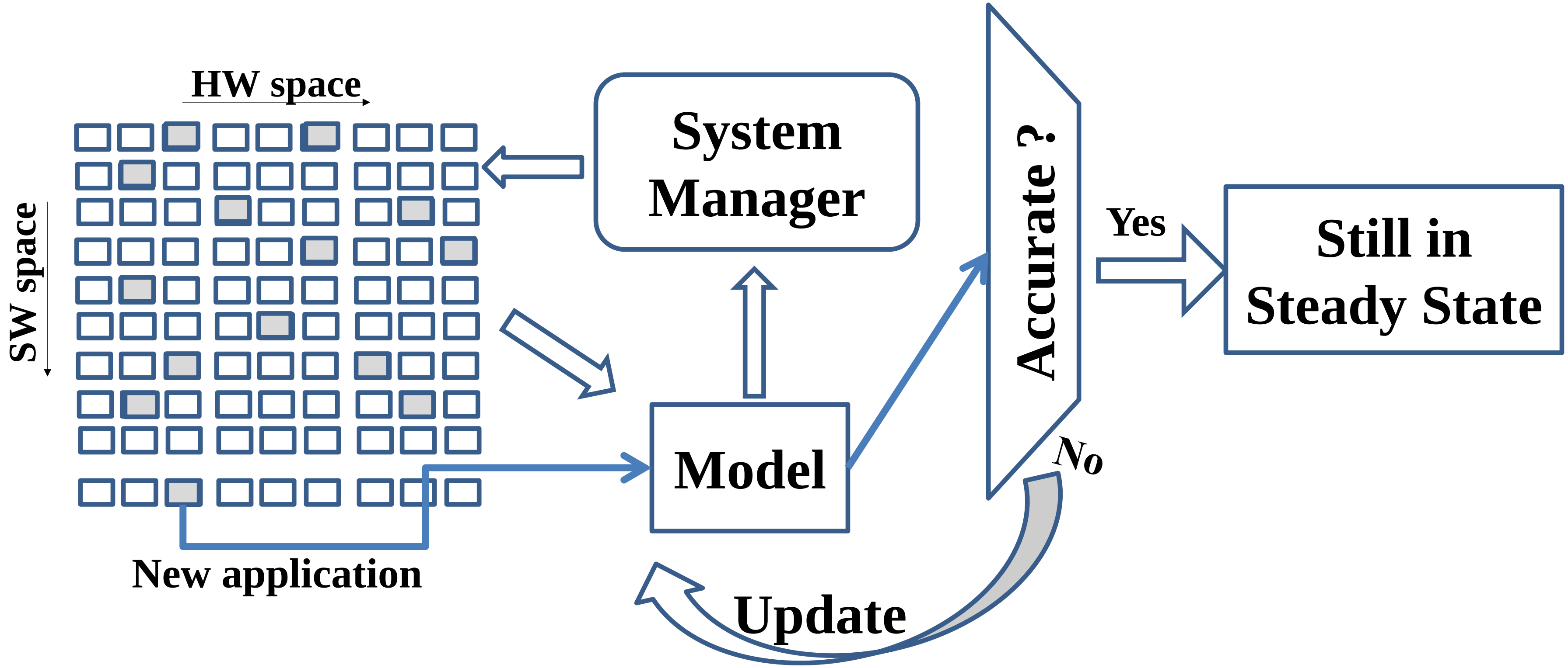
Inductive Hypothesis

- System in steady state
- Accurate model is trained M(H,S)
- Manager uses model predictions



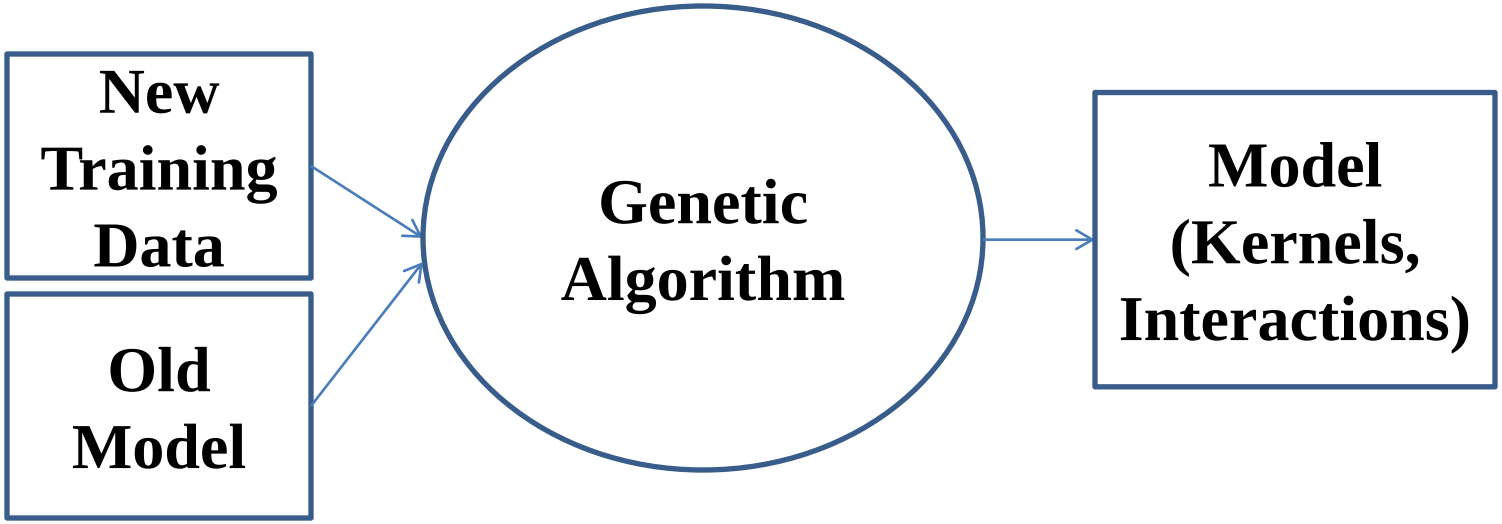
Inductive Step

– System is perturbed with new SW or HW
– Profile new SW-HW, check prediction



Model Updates

- Poor prediction triggers model update
  - Collect a few profiles for new SW (e.g., 10-20)
  - Update kernels, interactions, fit

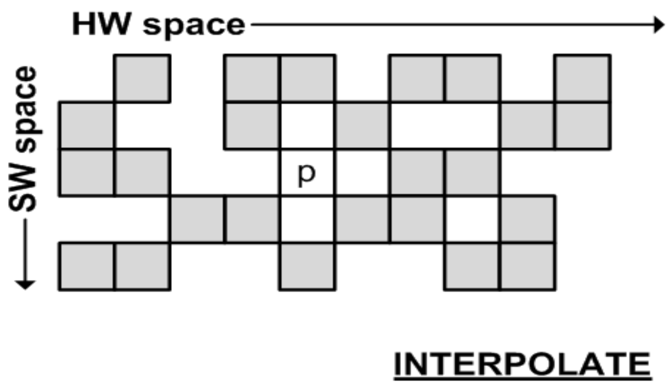


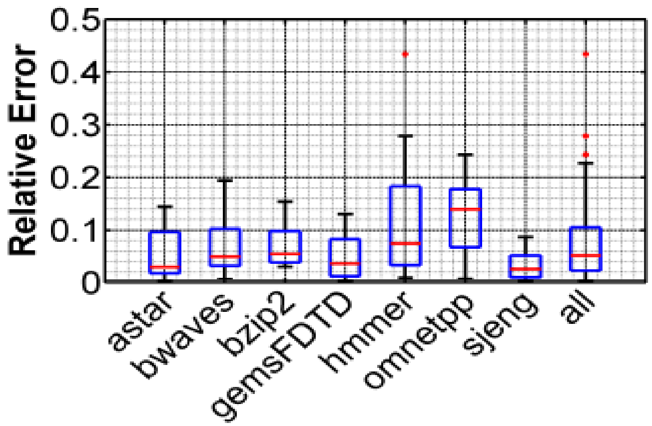
Integrated HW & SW Space

- Hardware Space (17 parameters)
  - Pipeline parameters → e.g. width, rob size
  - Cache parameters → e.g., cache size, associativity
  - Functional unit → e.g., ALU count
- Software Space (13 parameters)
  - Instruction mix
  - Locality → e.g., re-use distance
  - ILP → e.g., producer-consumer distance

Steady State Interpolation

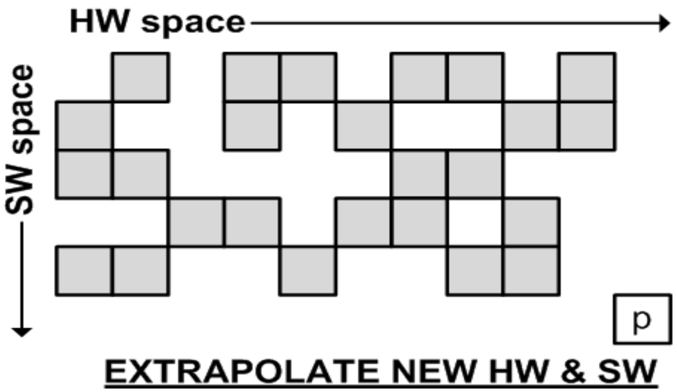
- Train model with sparse HW-SW profiles
- Interpolate for HW-SW pairs not profiles

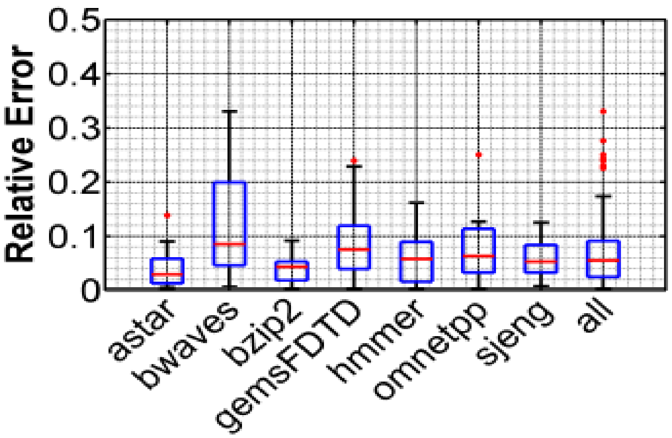




Perturbed Extrapolation

- – Train model with sparse HW-SW profiles
- – Extrapolate for new SW and new HW

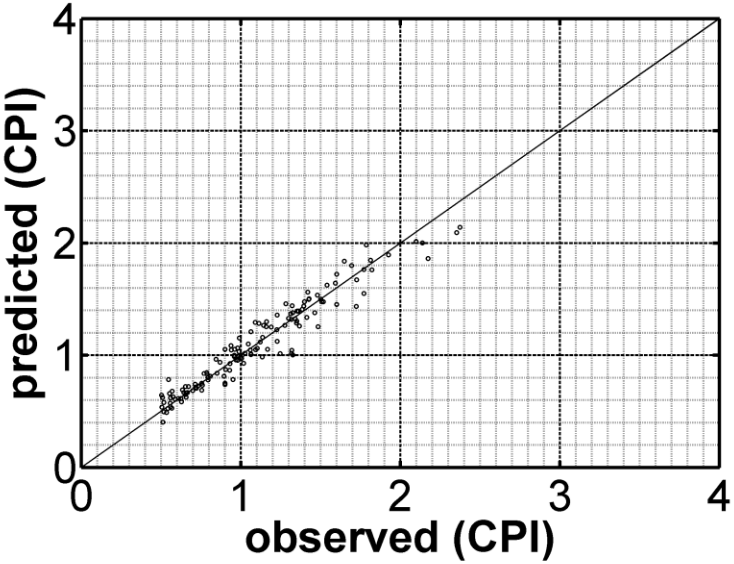


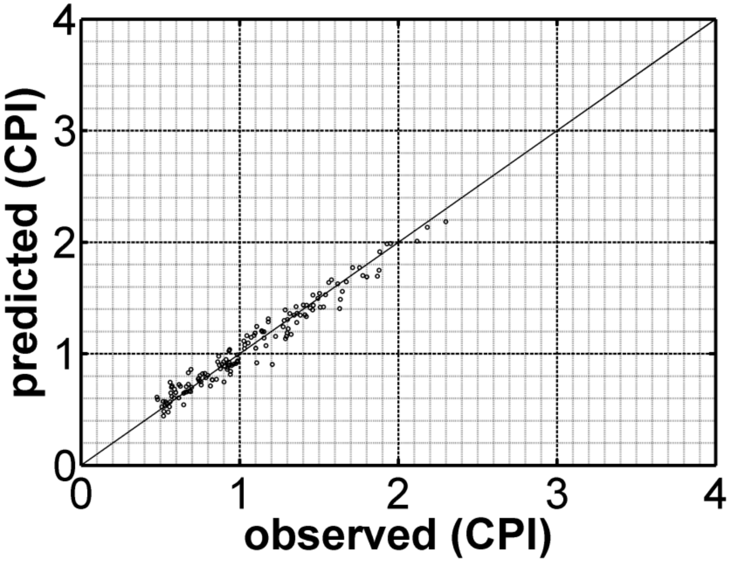


- – Predict app ***n*** from ***n-1*** apps
- – Also supports SW variants (compiler opt, data inputs)

Relative Accuracy

– Accurate interpolation, extrapolation
– Correlation coefficient > 0.9





Outline

- Introduction
- Inferred Performance Models
- Generalized Models
- **Specialized Models**
- Conclusions

Specialized Models

- Generality is expensive
  - Requires many SW characteristics (e.g,. 13)

- With domain knowledge, SW behavior expressed at higher level
  - Reduces number of SW characteristics
  - Reduces profiling cost
  - Increases model accuracy

Sparse Matrix-Vector Multiply

$$\mathbf{A} = \begin{pmatrix} a_{00} & a_{01} & 0 & 0 & 0 & 0 \\ a_{10} & a_{11} & 0 & 0 & a_{14} & a_{15} \\ 0 & 0 & a_{22} & 0 & a_{24} & a_{25} \\ 0 & 0 & 0 & a_{33} & a_{34} & a_{35} \end{pmatrix}$$

$$\texttt{b\_value} = \left(a_{00} \; a_{01} \; a_{10} \; a_{11} \;\middle|\; 0 \; 0 \; a_{14} \; a_{15} \;\middle|\; a_{22} \; 0 \; 0 \; a_{33} \;\middle|\; a_{24} \; a_{25} \; a_{34} \; a_{35} \right)$$

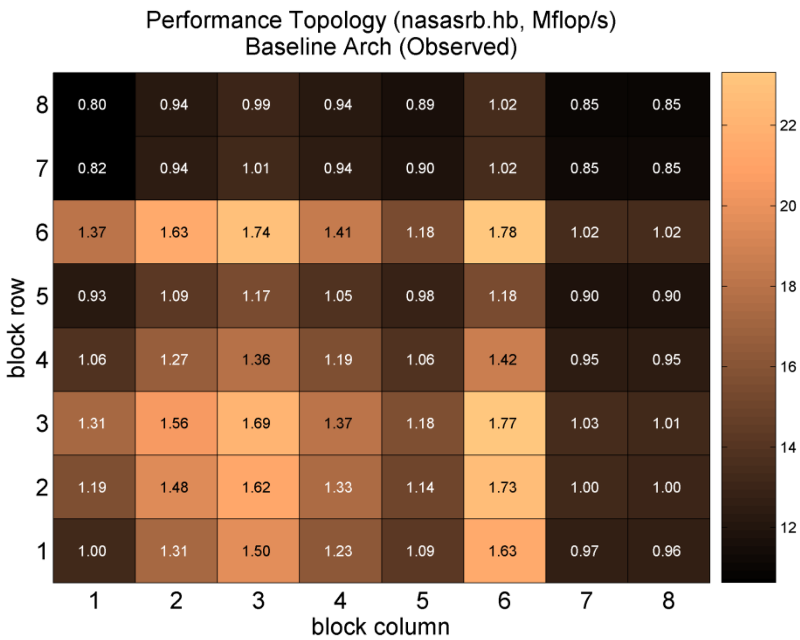
– Compute y=Ax+b when A is sparse, blocked

– SW space → block row, block column, fill ratio

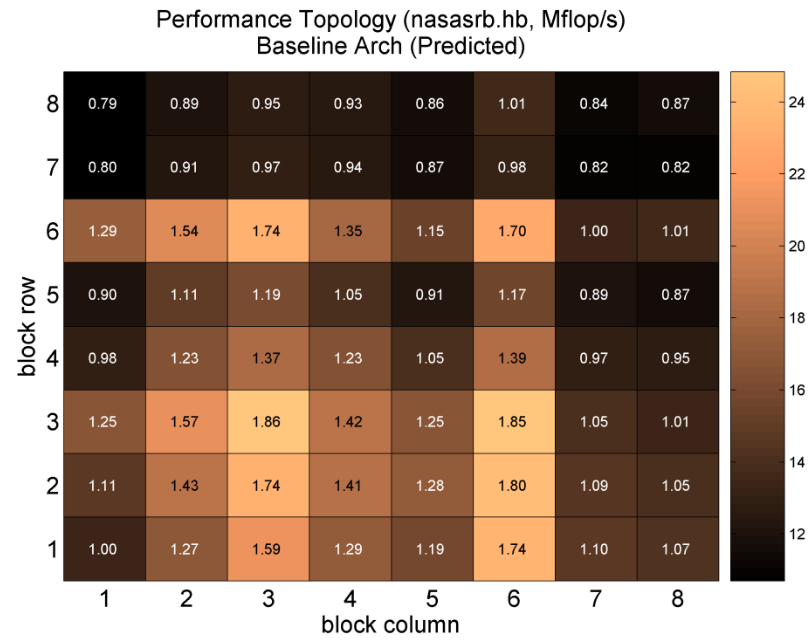
– HW space → cache

SpMV Model Accuracy

– Models irregular performance caused by fill ratios



True performance



Predictive performance

Also in the paper…

- Shard-level prediction
  - Basis of application prediction
- Genetic algorithm evaluation
  - Convergence versus model accuracy
- Coordinated optimization for SpMV
  - Optimize HW and software
  - Optimize performance and power

Conclusions

- Present framework to close data-to-decision gap
- Infer performance from huge, sparse data
- Automate modeling in dynamic managers
- Apply domain knowledge for concise models

Inferred Models for Dynamic and Sparse Hardware-Software Spaces

Weidan Wu, Benjamin C. Lee
Duke University