Stargazer: STAtistical Regression-Based GPU Architecture AnalyZER

Wenhao Jia, Kelly A. Shaw, Margaret Martonosi

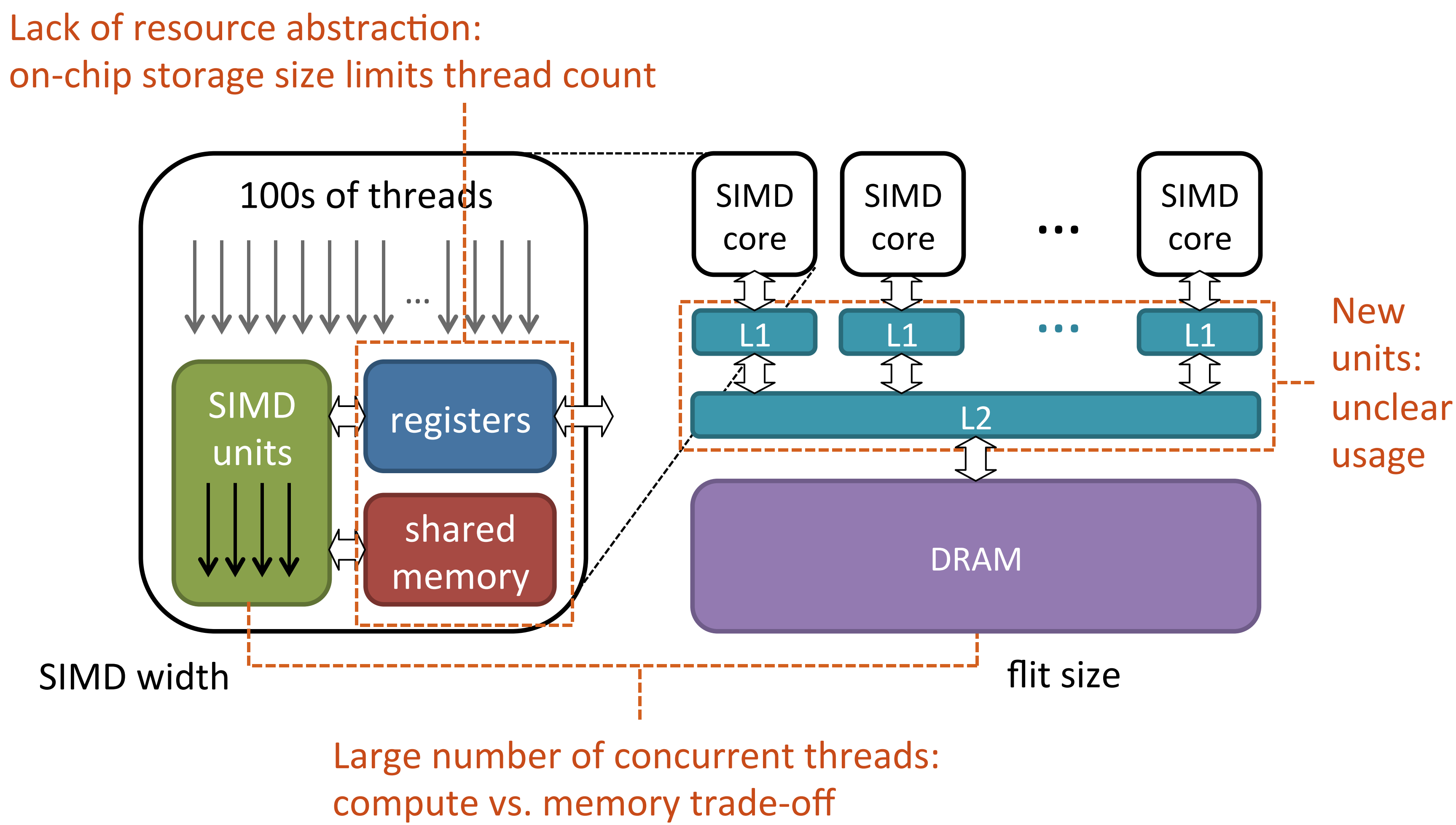
ISPASS 2012

PRINCETON UNIVERSITY

GPU Design Space Exploration

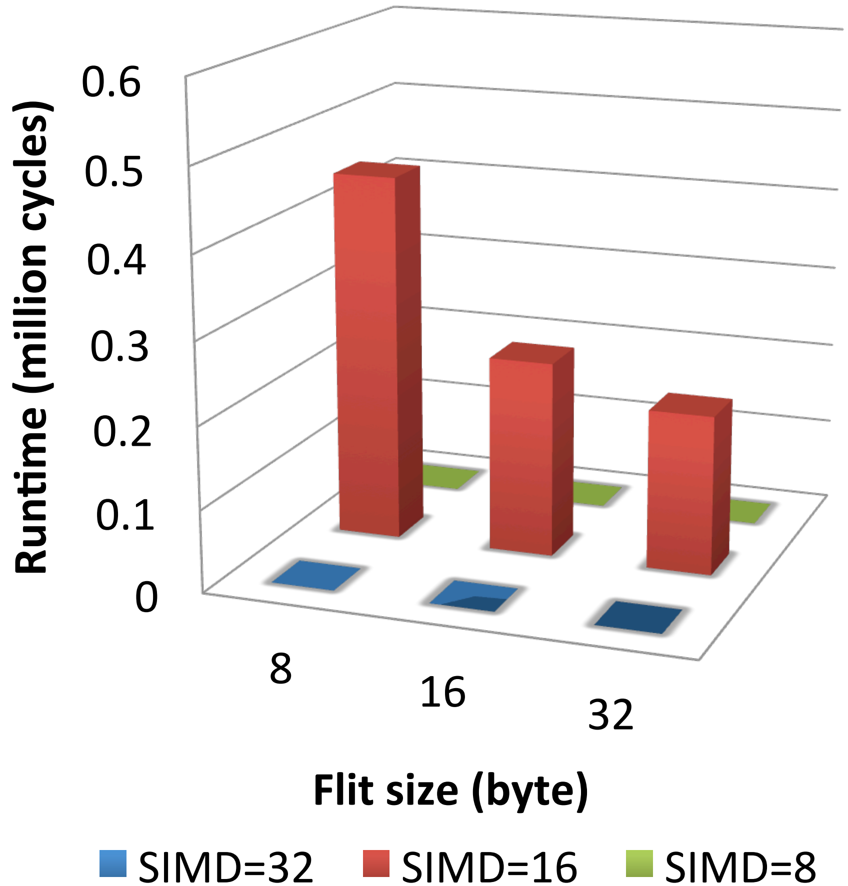
- Graphics Processing Units (GPUs) are becoming increasingly popular as parallel computing platforms
- The goal of GPU design space exploration:
  performance = function(design parameters)
  - H/W perspective: guide future GPU designs
  - S/W perspective: performance portability across designs
- Both hardware and software developers will benefit from a fast and accurate GPU design space explorer

GPU Design Complexity



Matrix Multiply—2 Factors

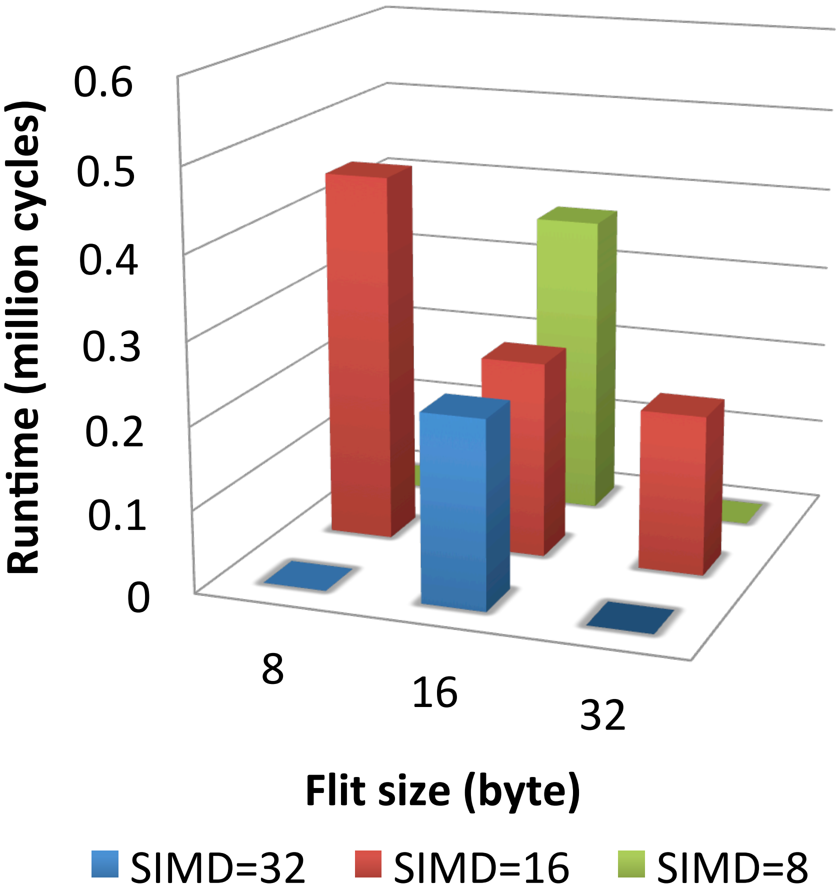
runtime ~ f(flit)



Matrix Multiply—2 Factors

runtime ~ f(flit)

runtime ~ g(simd)

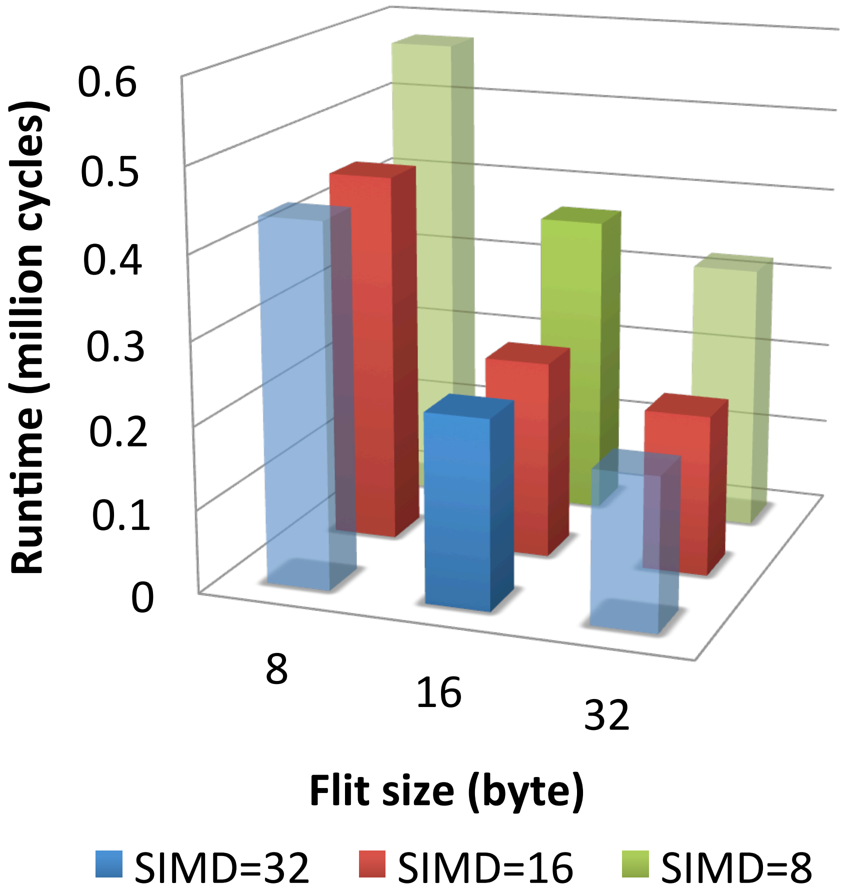


Matrix Multiply—2 Factors

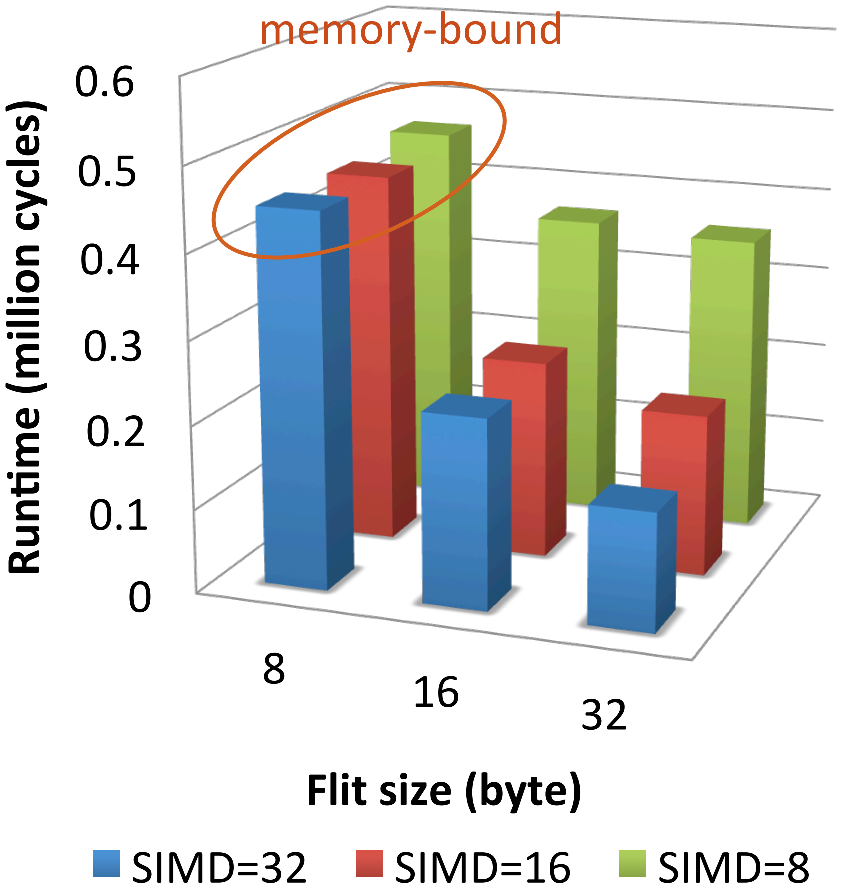
runtime ~ f(flit)

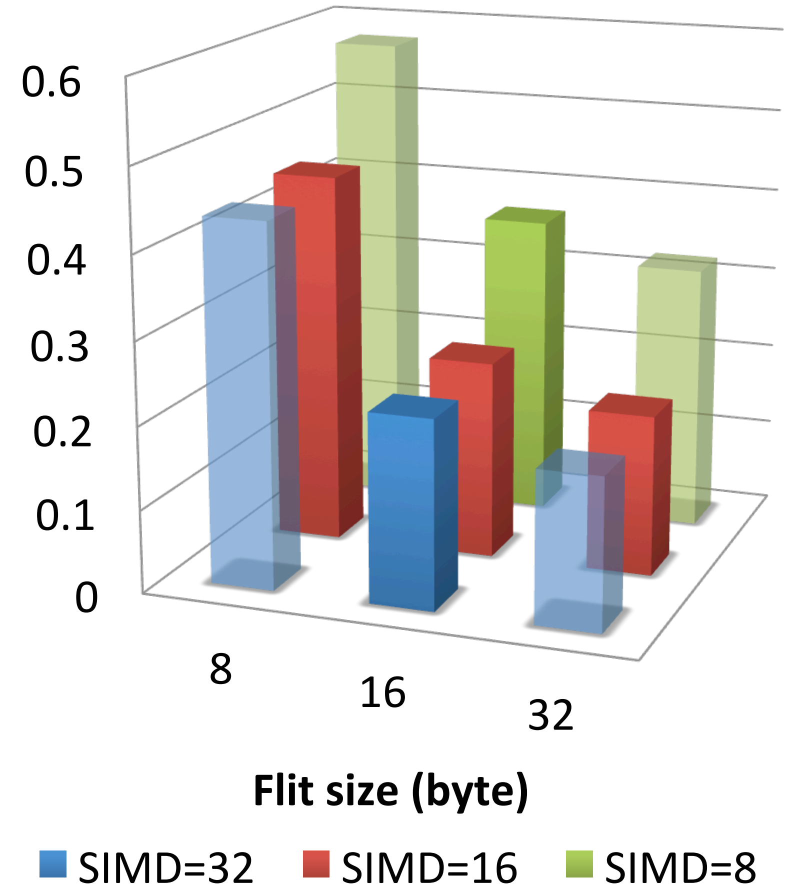
runtime ~ g(simd)

runtime ~ f(flit) + g(simd)

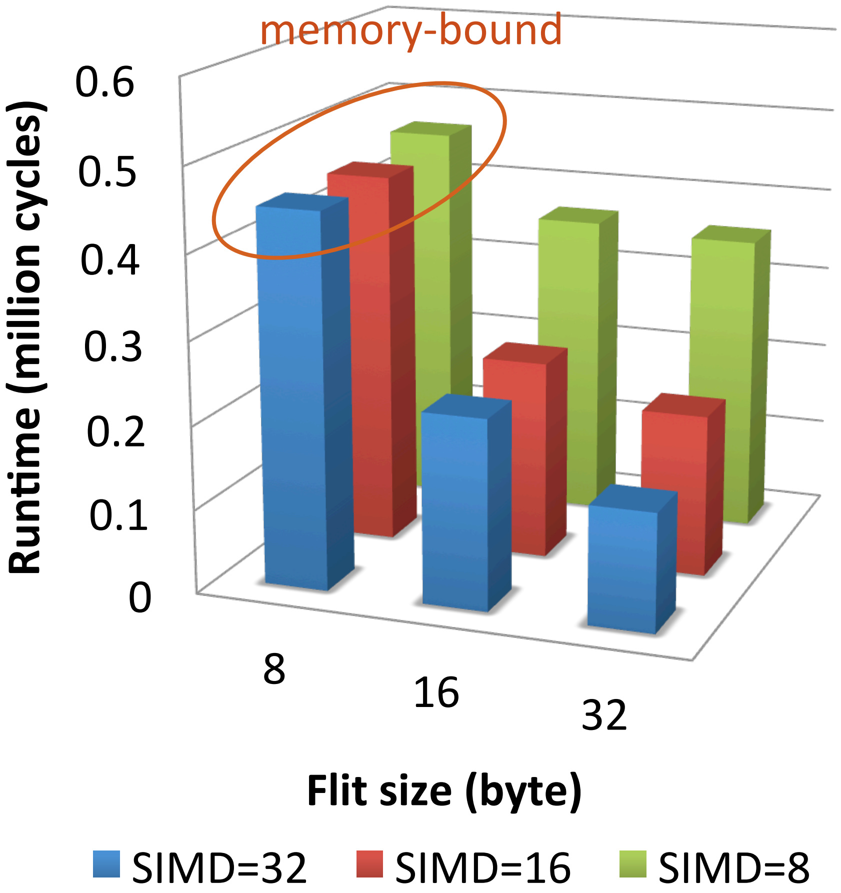


Matrix Multiply—2 Factors





Matrix Multiply—2 Factors



runtime ~ f(flit)

runtime ~ g(simd)

runtime ~ f(flit) + g(simd) + h(flit : simd)

↕

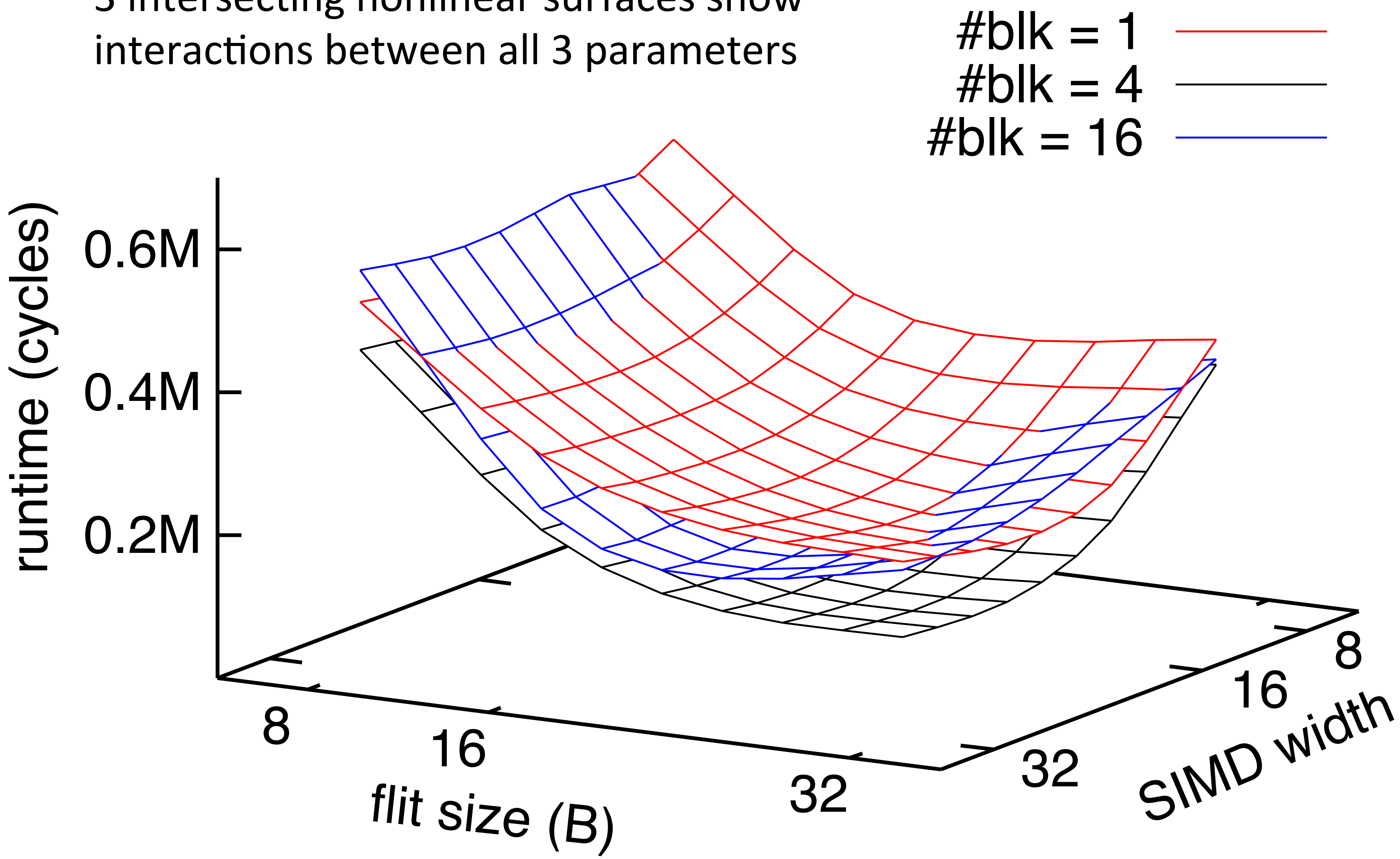
runtime ~ flit + simd + flit : simd

Is modeled by additive interaction

Parameter interactions require us to explore the GPU design space thoroughly

Matrix Multiply—3 Factors

3 intersecting nonlinear surfaces show interactions between all 3 parameters



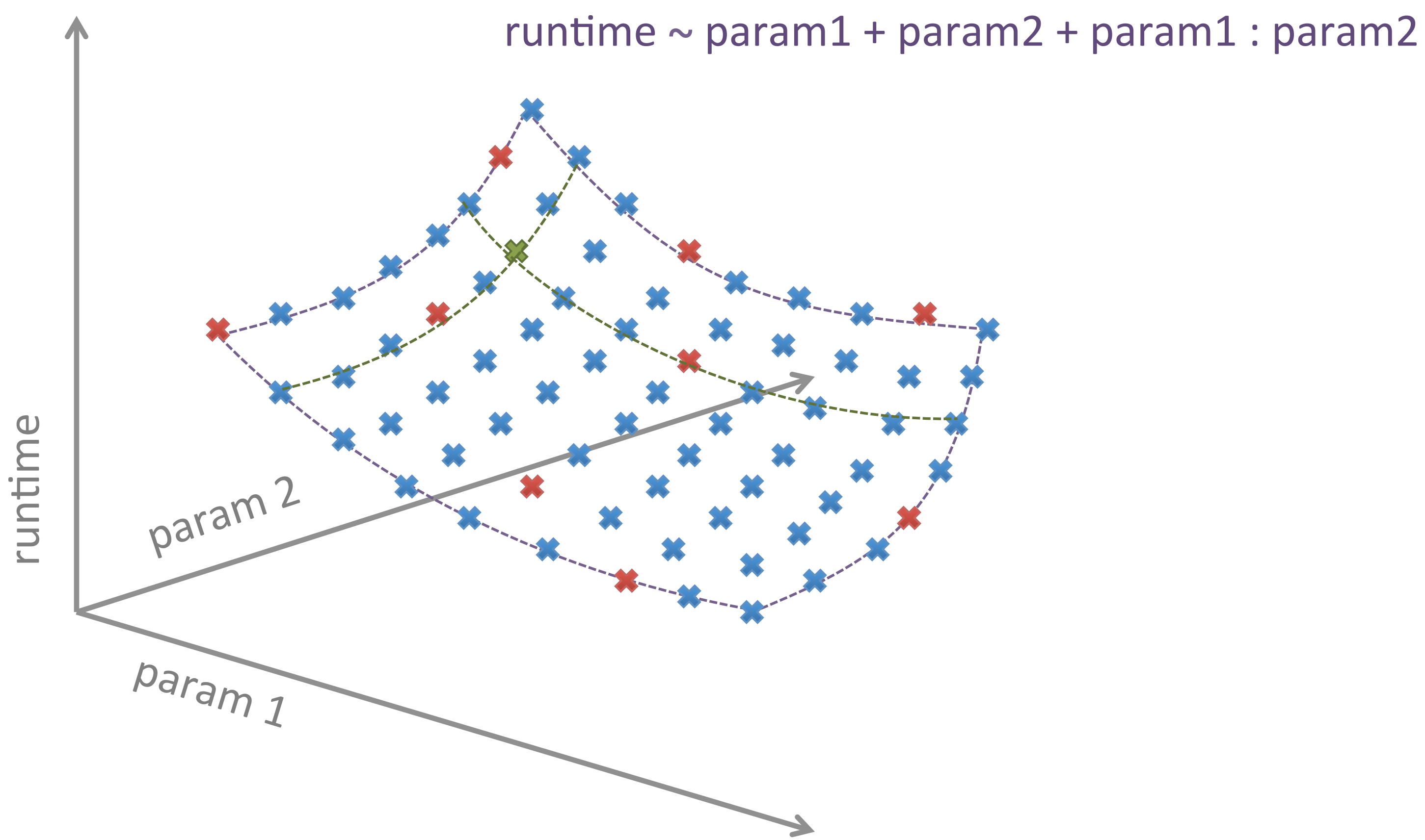
Our Work

- Provide an effective statistical regression-based GPU design space exploration framework
  - Automated: Automatically discover significant factors and their interactions
  - Efficient: Up to 15000× speed-up vs. exhaustive exploration
  - Accurate: 1.1% average prediction error when only 0.03% of the space is sampled
- Example uses of Stargazer
  - Design space pruning
  - Application characterization

Related Work

- GPU Performance Modeling
  - Simulators: Accurate but simulating all points is time-consuming
  - Analytical models: Descriptive but relies on a set of program simplification rules
- CPU/CMP Design Space Exploration
  - Techniques based on artificial neural network and linear regression
  - Not yet ported to GPUs
  - Also not fully automated and parallelized

Regression: Sample–Model–Predict



Regression builds an application-specific performance model which can predict GPU performance through interpolation

Step 1: Sampling

- Design space specification
  - Independent variables P_i ($i = 1, 2, \ldots, n$): SIMD width, memory bandwidth, concurrent block count, …
  - Dependent variable: runtime, power, …
  - Space: $|P_1| \times |P_2| \times \ldots \times |P_n|$ points — large!
- We sample the space randomly and uniformly
- Measure the performance of each sample: simulation, real system evaluation, …
- All samples can be measured in parallel

Step 2: Regression Modeling

- runtime ~ $f_1(P_1) + f_2(P_2) + \ldots + f_n(P_n) + f_{1,2}(P_1 \times P_2) + f_{1,3}(P_1 \times P_3) + \ldots + f_{n-1,n}(P_{n-1} \times P_n)$
  - f(): Natural cubic splines (piecewise polynomial functions)
  - The complete model has many [$O(n^2)$] terms
- A stepwise method automatically includes only relevant factors and pair interactions in the model
- Greedy, based on adjusted R^2 of tentative models

Select Factors Based on Adjusted R^2

- R^2 (coefficient of determination) measures how well a model fits observed data ($0 < R^2 < 1$)
  - However, R^2 always grows as more terms are included (may lead to overfitting)
- Adjusted R^2 quantifies the real marginal benefit of each additional term
  - A new model's adjusted R^2 is larger than the old model's R^2 only if the effect of a new term is bigger than what a random term would have brought

Stargazer: The Stepwise Algorithm

```
current model M = {}
unused parameter set T = {P_1, P_2, …, P_n}
while T is not empty
  for each P_i in T
    generate a tentative model M_i = M + P_i
  select the M_imax with the highest adjusted R^2
  if M_imax's adjusted R^2 > M's R^2
    M = M_imax, T = T - P_i
    for each P_j (j != i) already in M
      if interaction P_i:P_j is significant
        M = M + P_i:P_j
    else
      return M
```

Initialization

If the next most significant factor indeed affects runtime, include it in the model

Also test its interactions with included factors

Else exit the routine

Methodology

- Simulator: GPGPU-Sim 2.1.1b
- Benchmark programs
  - GPGPU-Sim: AES, BFS, CP, LPS, RAY, STO
  - Rodinia: `backprop`, `bfs`, `hotspot`, `nw`
  - CUDA SDK: `matMul`
- Modeling target: runtime
- Training: 300 samples / Testing: 200 points
- Environment: R

The Studied Design Space

Parameters	Values	Unit	#Points	Meaning
#blk	1, 2, 4, 8, 16	blocks/core	5	# concurrent blocks
c$	1, 2, 4, 8, 16, 32	KB	6	Constant cache size
t$	1, 2, 4, 8, 16, 32	KB	6	Texture cache size
smp	1, 2, 4, 8	count	4	# shared memory ports
ccp	1, 2, 4, 8	count	4	# constant cache ports
simd	8, 16, 32	count	3	SIMD width
mshr	1, 2, 3	count/thread	3	# Miss Status Holding Regs
dramq	16, 32, 64	count	3	DRAM scheduler queue size
intra	1, 2, 4, 8	count	4	Intra-warp coalesce
inter	2, 4, 6	count	3	Inter-warp coalesce
TOTAL			933,120	

Stepwise Example: Matrix Multiply

runtime ~	#blk	c$	t$	smp	ccp	simd	mshr	dramq	intra	inter
R^2	0.719	0.023	0.004	0.004	0.004	0.154	0.001	0.009	0.014	0.002

#blk +	c$	t$	smp	ccp	simd	mshr	dramq	intra	inter
Adjusted R^2	0.717	0.718	0.719	0.716	0.851	0.716	0.719	0.718	0.718

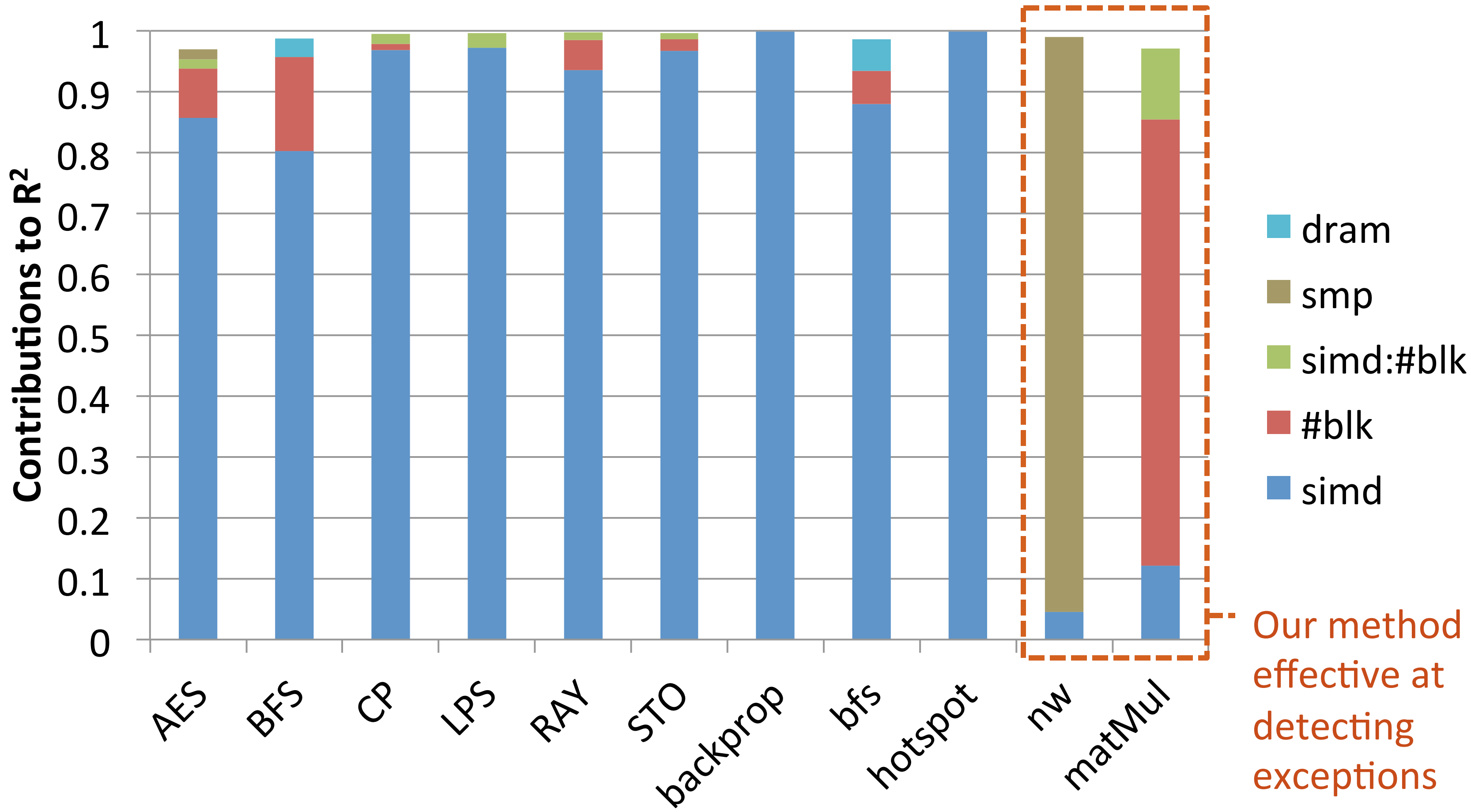
simd +	simd:#blk
Adjusted R^2	0.967

⋮ ⋮

Current model: `runtime ~ #blk + simd + simd:#blk + intra + intra:#blk`

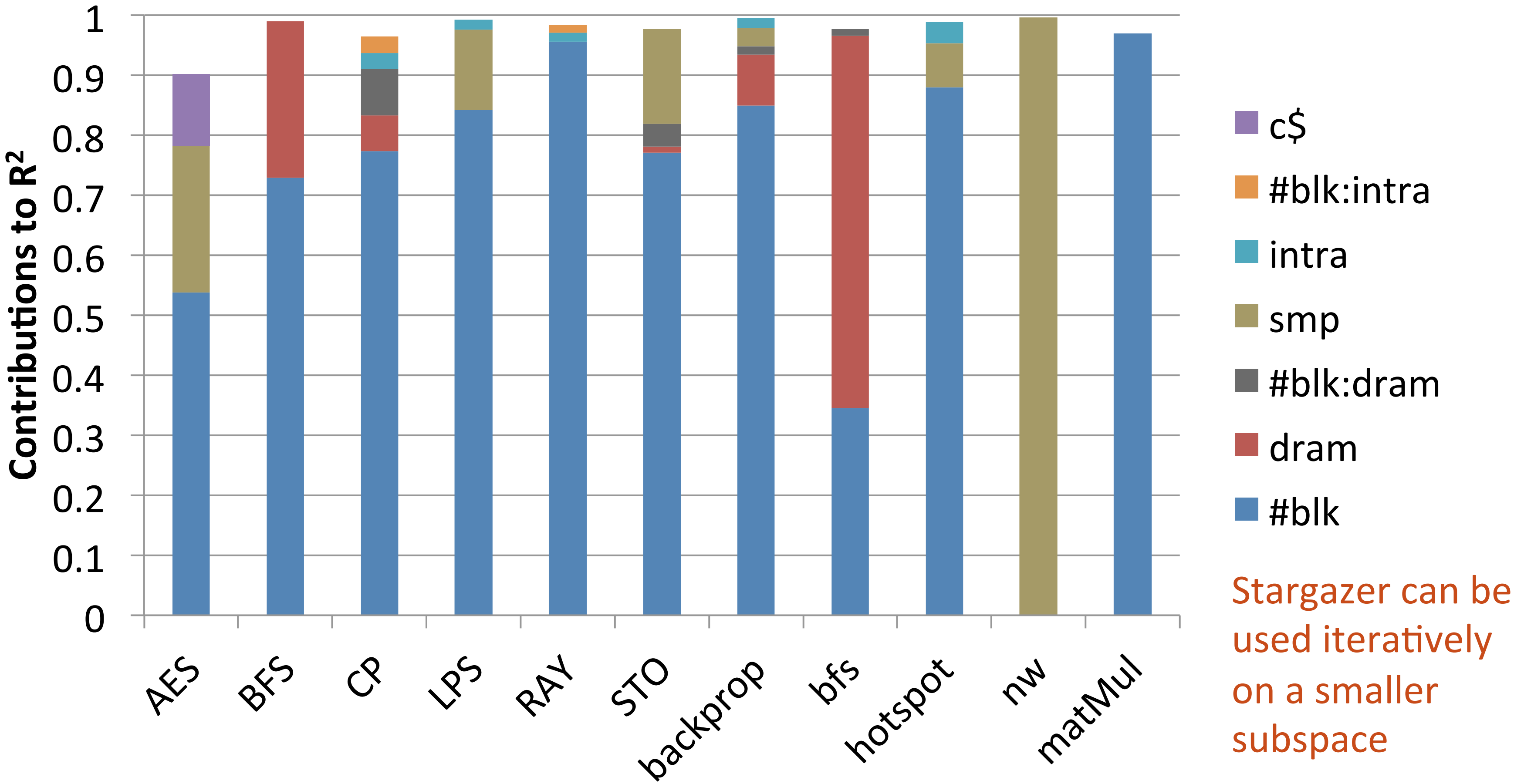
R^2 ~~0.719~~ ~~0.853~~ ~~0.973~~ 0.975

R^2 Close to 1: Good Model Fit



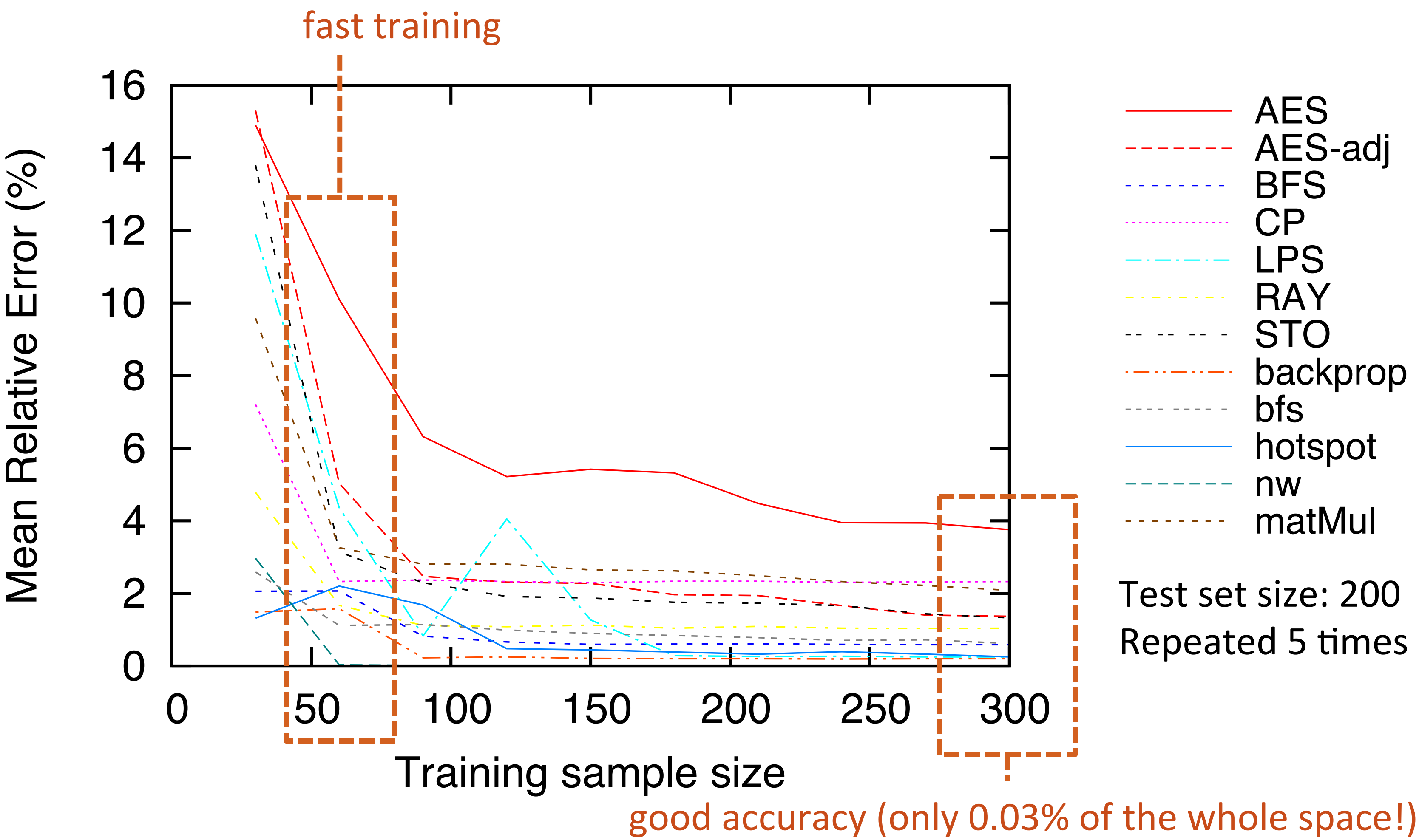
Our method is effective at detecting exceptions

At SIMD = 32: Diverse Secondary Factors



Stargazer can be used iteratively on a smaller subspace

Prediction Accuracy vs. Sample Size



Test set size: 200
Repeated 5 times

Results Summary

- Reduce simulation time
  - Automatically prune design space
  - 30–60 samples: < 5% error for most programs
  - Up to 15000× simulation time reduction (60 samples)
- Application characterization (paper has more details)
  - The number and shape of splines reflect application behavior
  - Can be used to assess benchmark suite diversity

Conclusions

- Regression modeling is a fast and effective method to model extremely large GPU design spaces
- Stargazer automatically generates compact models for a complex design space
  - < 300 samples for exploring a 933K-point design space
  - 1.1% average error on 11 benchmark programs
- Useful for GPU designers and programmers, and should also improve on state-of-the-art for CPU design space exploration tools

Thank you! Please try Stargazer at
http://www.princeton.edu/~wjia/stargazer

STAtistical Regression-Based GPU Architecture AnalyZER

PRINCETON UNIVERSITY

The background image is a screenshot of Stellarium, simulating the night sky of New York City at 11:22pm on March 16th 2012