Starchart*: GPU Program Power/Performance Optimization Using Regression Trees

Wenhao Jia, Princeton University
Kelly A. Shaw, University of Richmond
Margaret Martonosi, Princeton University

*Statistical Tuning via Automatically- and Recursively-Constructed, Hierarchically-Applied Regression Trees

The Problem

[Q1] What if you need to select between multiple GPU platforms?

- Decide which platform offers the best power/performance
- For each platform, find the best parameter settings
- Are there parameter settings that work well across several platforms?

[Q2] What if you need to choose program operating points that optimize power while hitting certain performance targets?

- Performance targets change dynamically
- Understand how paramater settings affect performance/power

Complex design spaces → Hard to answer

How to automate choices about HW and SW design options?

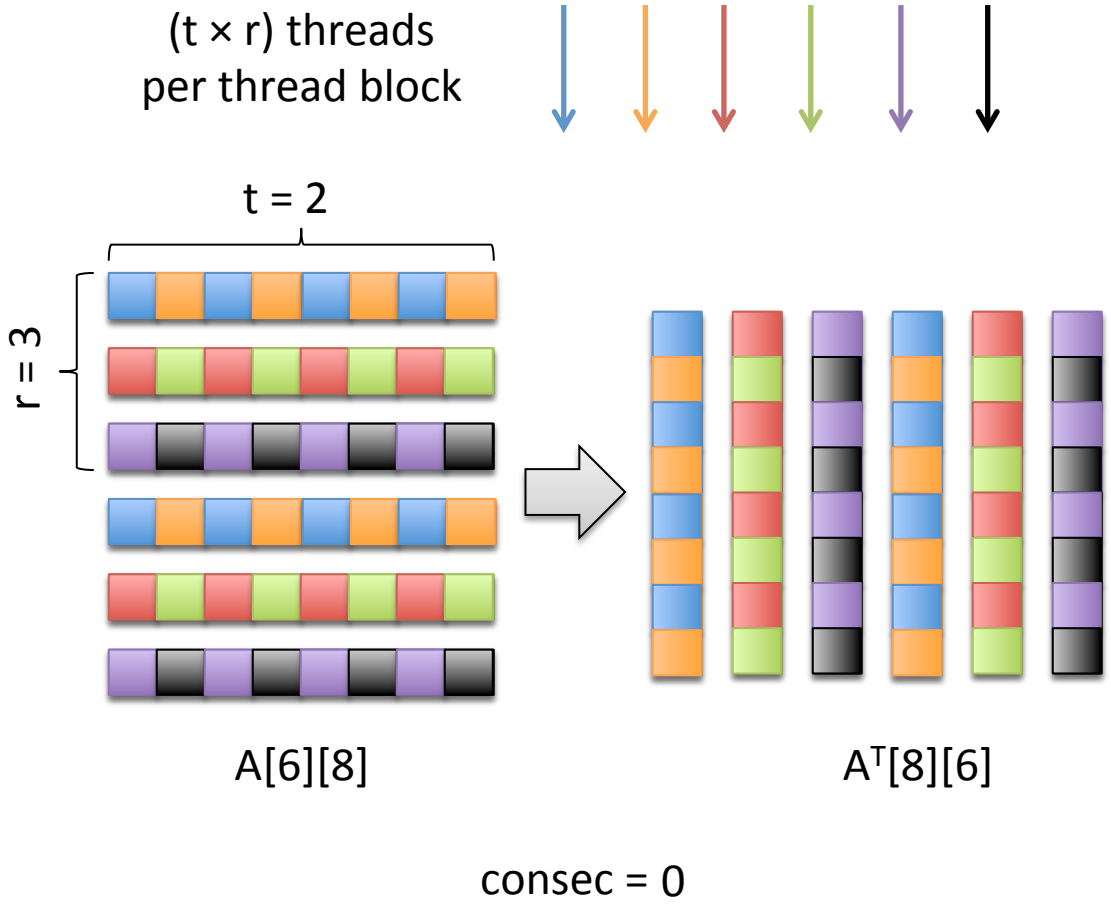
Design Space Exploration: Existing Approaches

- Exhaustive experiments
  - Time-consuming
  - Even if possible, hard to analyze high-dimensional space
- Center-point-based exploration
  - Based on human experience and intuition
  - Can miss important trends
- Performance/power models
  - [Joseph06HPCA][Lee06ASPLOS][Jia12ISPASS]
  - Linear regression doesn't work well across "performance cliffs", sacrificing accuracy when distinct subspaces exist
  - Can only find global optimal

This Work: Starchart Design Explorer

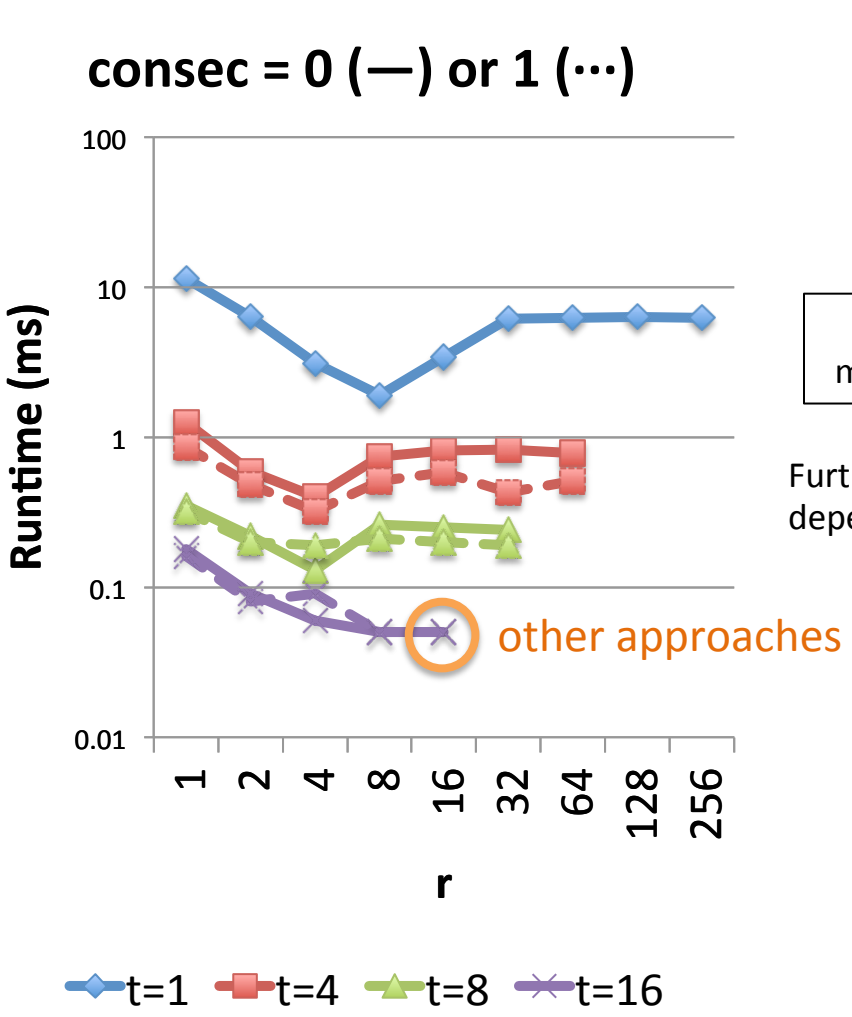
- Partition-based approach is powerful and robust
  - Handles real-system measurement variance
  - Handles “performance cliffs” and “subspaces” common for GPU systems
  - Applicable to multiple metrics and CPUs
  - Tree visualizations are intuitive
- For GPU users, tool builders & HW designers
  - Optimize designs within or across different platforms
  - Reveal power/performance trade-offs
  - Measure a program’s input sensitivity
  - > 300X speed-up in design space exploration

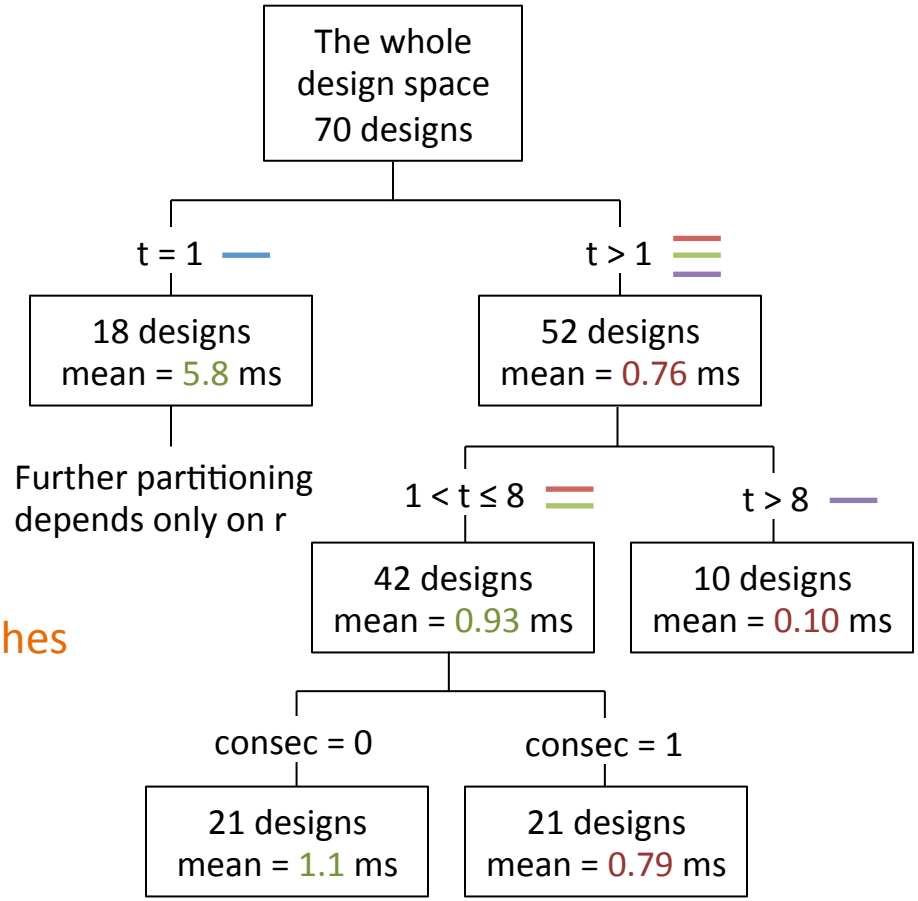
Motivation: Matrix Transpose



param	meaning	value
r	# rows / thread block	1–256
t	# threads / row	1–16
consec	threads work on consecutive elements?	0 / 1
	# total designs	70

Motivation: Matrix Transpose





Handles high-degree interaction
Handles distinct subspaces

r: #rows / thread block, t: #threads / row, consec: consecutive?

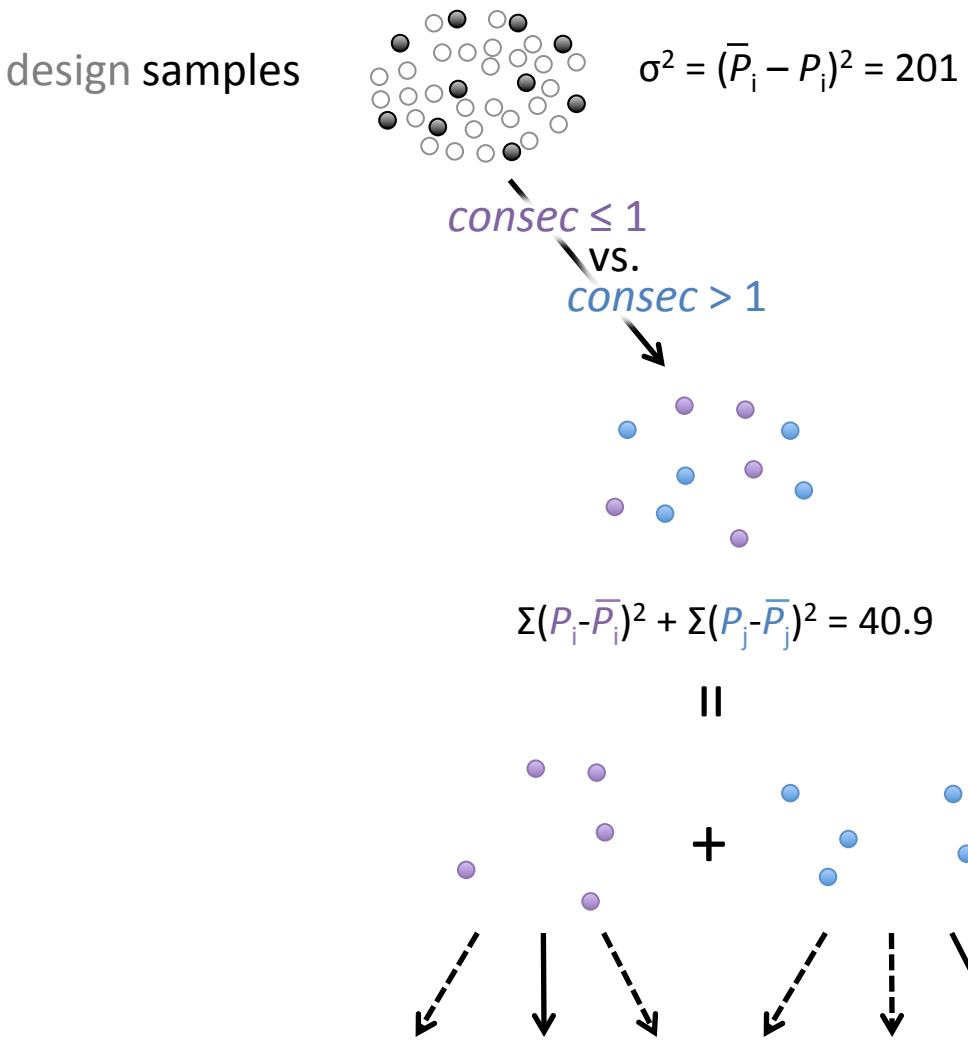
Starchart Workflow

- Step 1: Sampling
  - Uniformly & randomly sample and measure designs
- Step 2: Modeling
  - Recursively partition a space using samples
  - Based on regression tree theory, statistically sound
  - Robust enough to handle real-system experiments
  - Automated, comprehensive, and easy-to-visualize
- Step 3: Application
  - Solve subspace-based design problems

Example: Breadth-First Search

param	meaning	category	value
t	# threads / thread block	threads organization	1–1024
n	# nodes / thread	threads organization	1–64
consec	threads process consecutive nodes?	coalescing vs. caching	0 / 1
a	store attributes[] in parallel arrays?	data layout	0 / 1
s	store status[] in parallel arrays?	data layout	0 / 1
		# total designs # needed samples	~500K 200 (0.04%)

Recursive Partitioning



n: #nodes/thread, t: #threads/block, consec: consecutive? a & s: parallel arrays?

Algorithm

recursive partitioning

do

  for each current partition C

    try all tentative splits

    for each parameter d_s

      for each possible value v_t of d_s

      split C into C_1 and C_2 based on (d_s, v_t)

        $SSE_{st} = \Sigma_i(P_{1i} - P_1) + \Sigma_j(P_{2j} - P_2)$

    find the smallest SSE_{mn} among all SSE_{st}

    stopping criteria

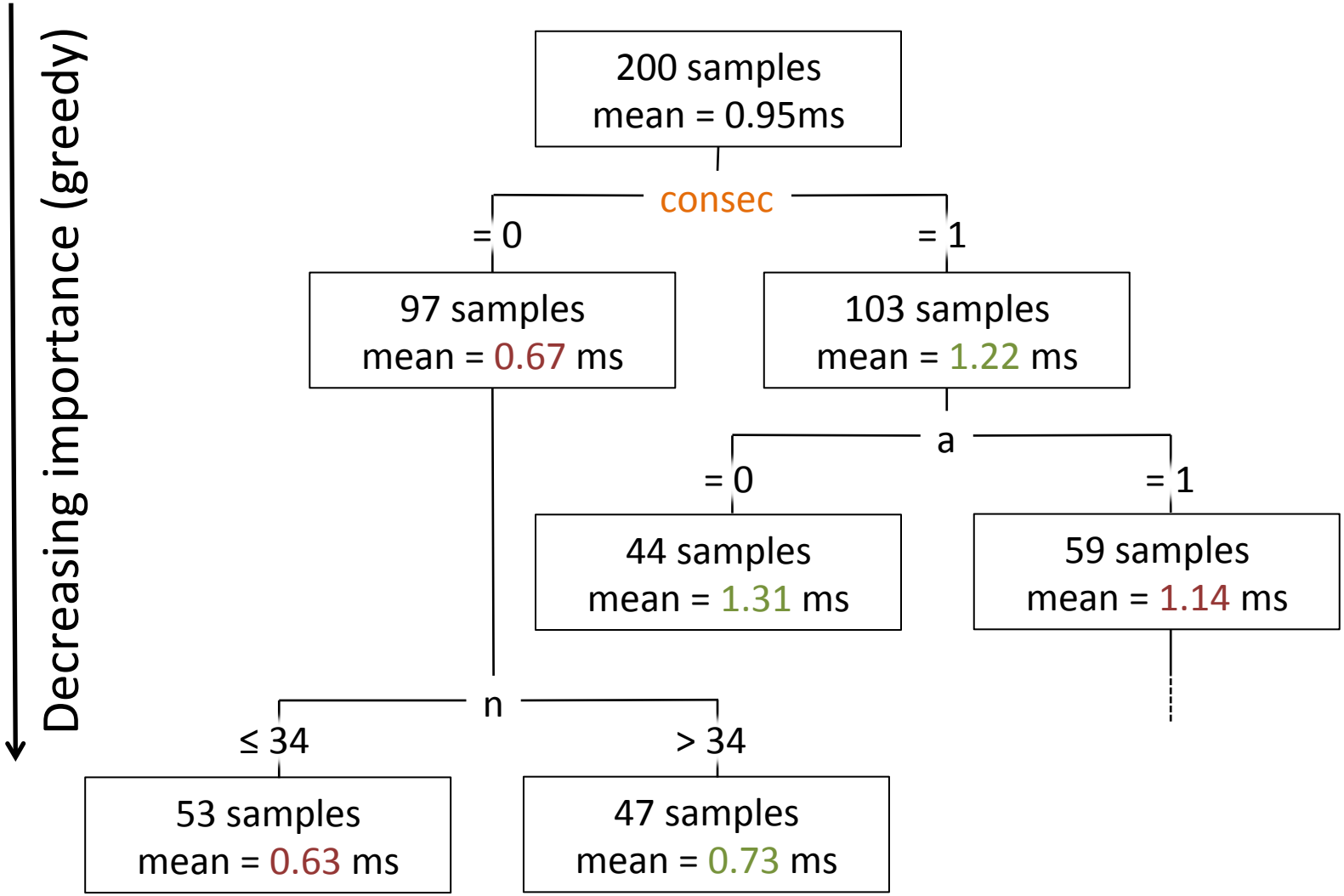
    if $SSE_{mn} > 0$

      split C using (d_m, v_n) and add to tree

while at least one partition was split

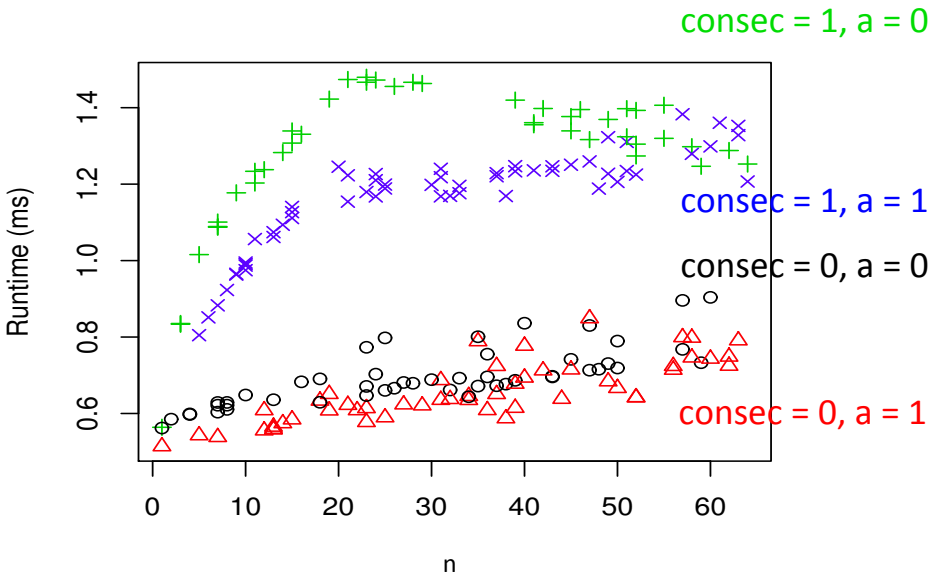
output the generated partition tree

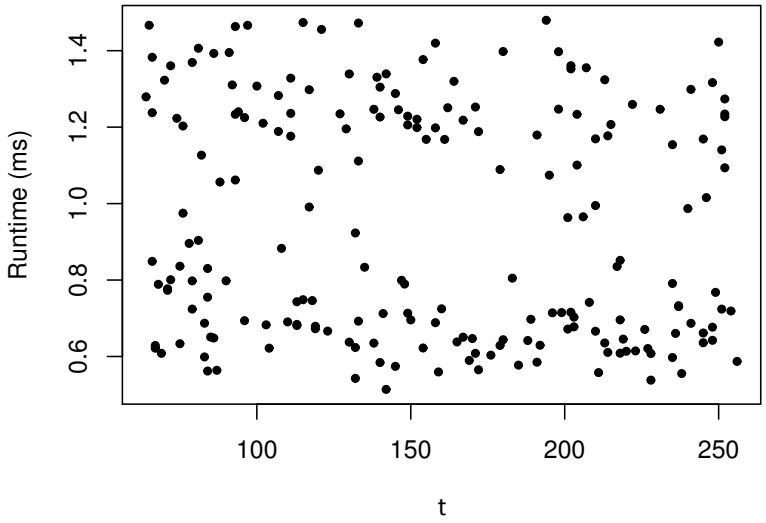
Regression Tree for BFS



n: #nodes/thread, t: #threads/block, consec: consecutive? a & s: parallel arrays?

Results Validation



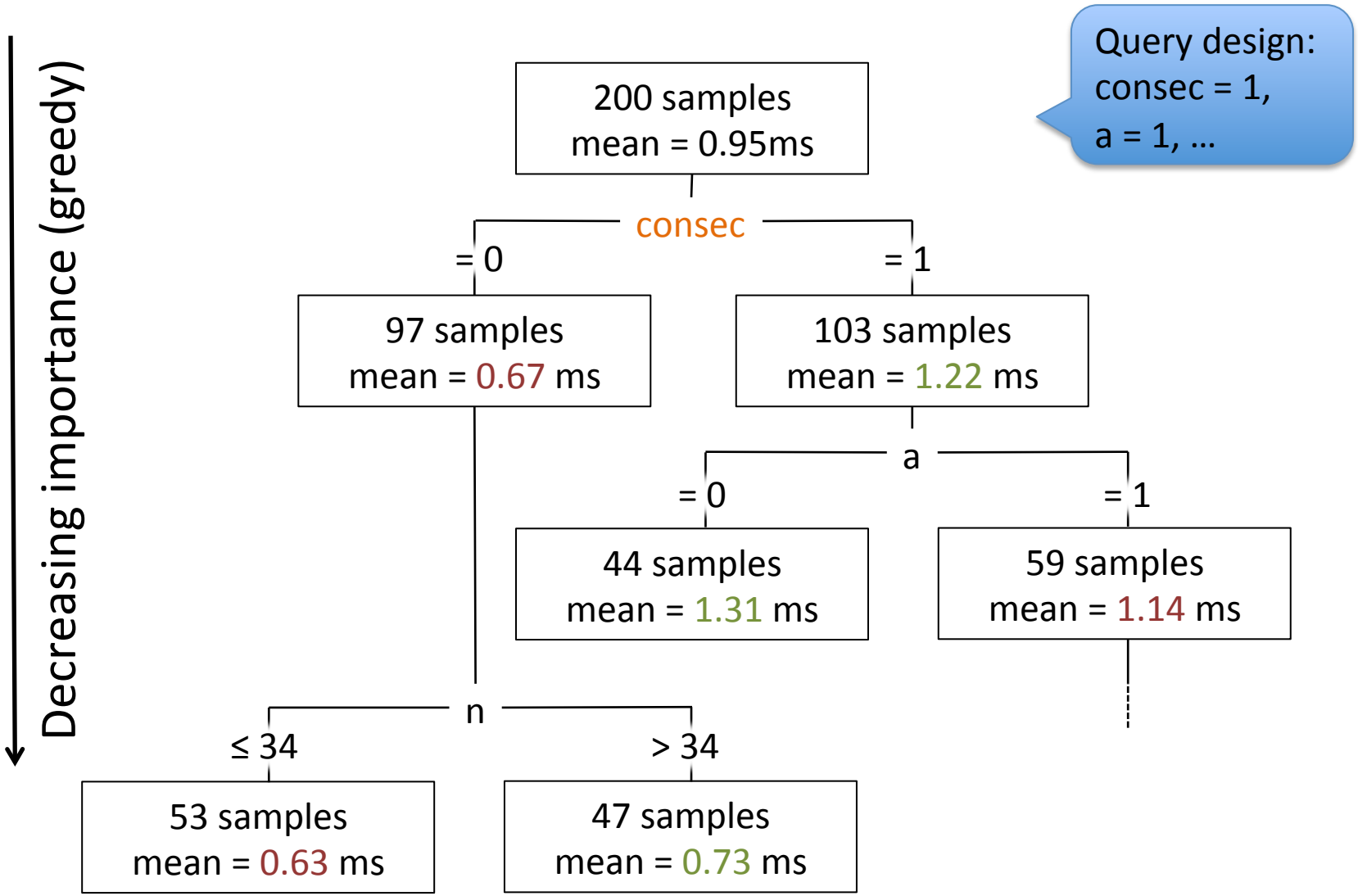


consec and *a* together divide the designs into several fairly disjoint sets

t does not appear in the tree, and hence runtime does not have a strong dependence on *t*

n: #nodes/thread, t: #threads/block, consec: consecutive? a & s: parallel arrays?

Regression Tree for BFS



n: #nodes/thread, t: #threads/block, consec: consecutive? a & s: parallel arrays?

How to Use Regression Trees

- With enough leaf nodes (~50 for our design spaces), a regression tree approaches a fairly accurate description of the design space
  - Can be used to predict the performance/power of any design by tracing a tree top-down
  - Can be used to look for the best design/subspace
  - Can be used to do subspace-based design exploration (4 case studies in the paper)

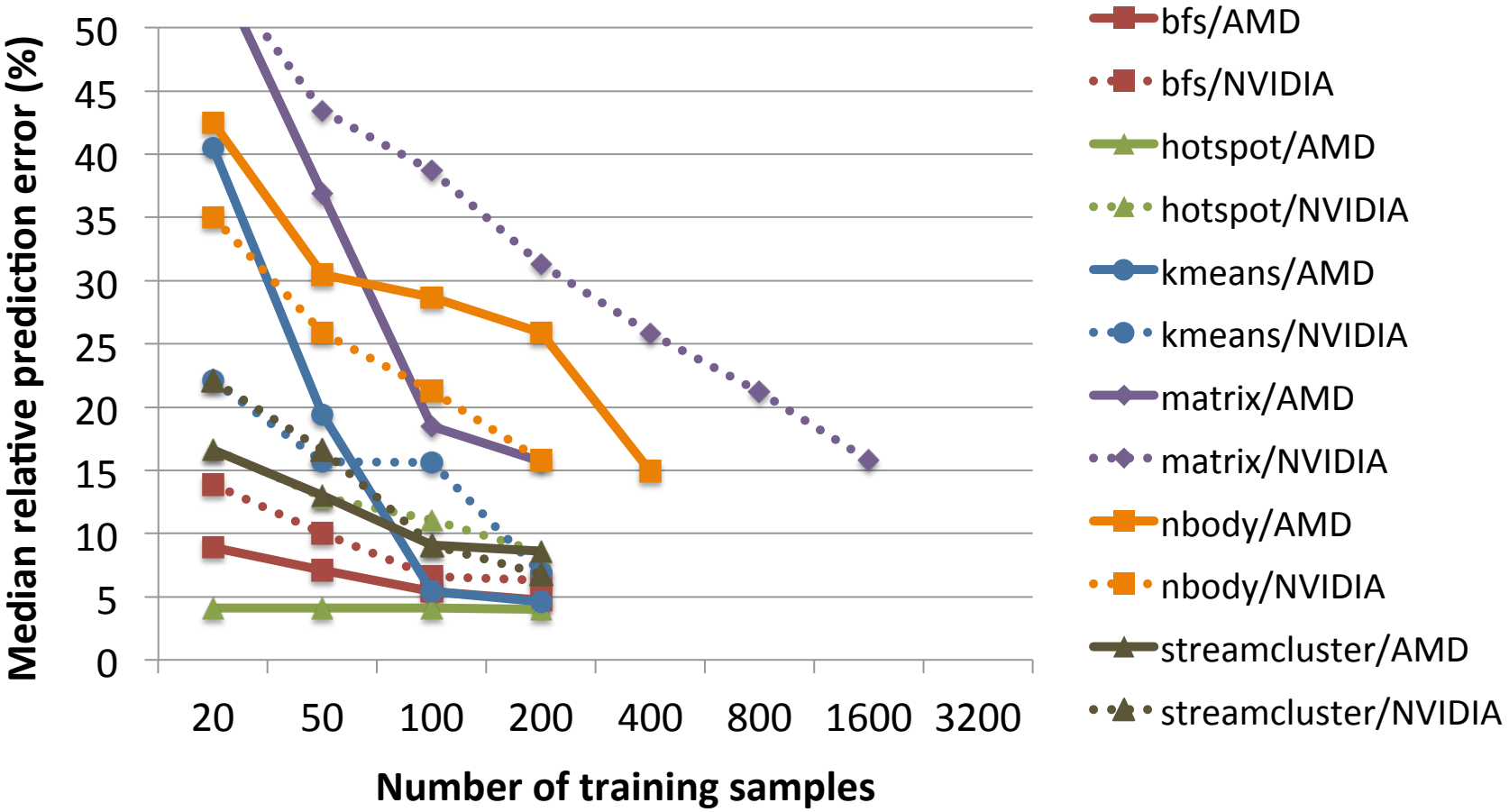
Starchart Design Questions

- When to stop splitting a partition
  - Stop when $\Delta SSE < 0$: clear meaning, more levels
  - Stop when $\Delta SSE < \delta$: fewer leaves, faster training
- What statistical model to use in each node
  - Arithmetic mean: simple, robust, more levels
  - Linear regression models: fewer levels

Real-System Measurement

Parameter	Value	
GPUs	NVIDIA Tesla C2070	AMD Radeon HD 7970
Software environment	CUDA	OpenCL
Benchmarks	bfs, hotspot, kmeans, streamcluster (Rodinia) matrix, nbody (NVIDIA SDK)	
Evaluated metrics	performance (i.e. runtime) power	
Design space sizes	66K–millions of designs	
Training samples	200–3200 (< 0.3% of design space sizes, i.e. > 300X productivity improvement)	

Accuracy vs. Training Set Size



- Use prediction accuracy on 200 validation samples to incrementally select training set size
- 200 samples for most programs, less than 0.3% of all design spaces

Revisiting Motivation Questions

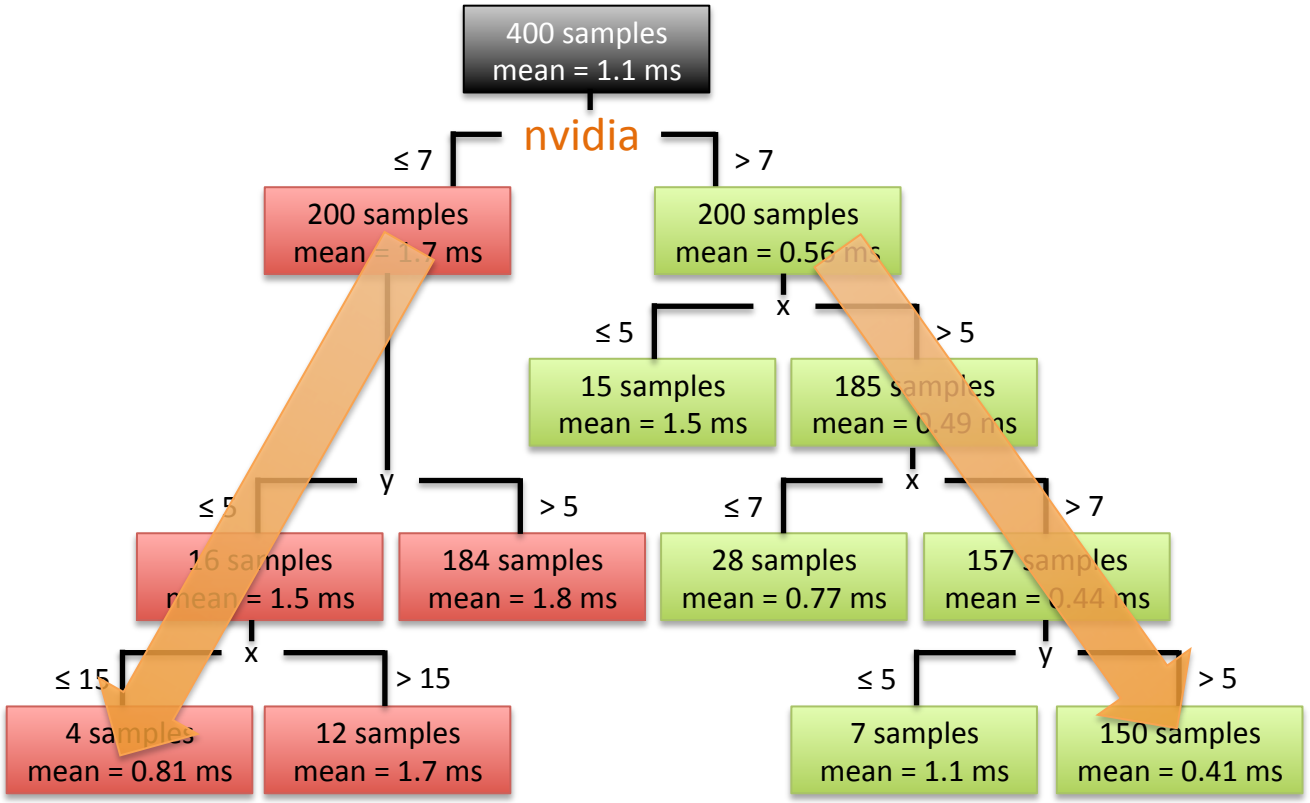
[Q1] What if you need to select between multiple GPU platforms
- – Decide which platform offers the best power/performance
- – For each platform, find the best parameter settings
- – Are there parameter settings that work well across several platforms?

[Q2] What if you need to choose program operating points that optimize power while hitting certain performance targets?
- – Understand how parameter settings affect performance/power
- – Performance targets change dynamically

Complex design spaces → Hard to answer

How to automate choices about H/W and S/W design options?

Case Study: Cross-Platform Optimization

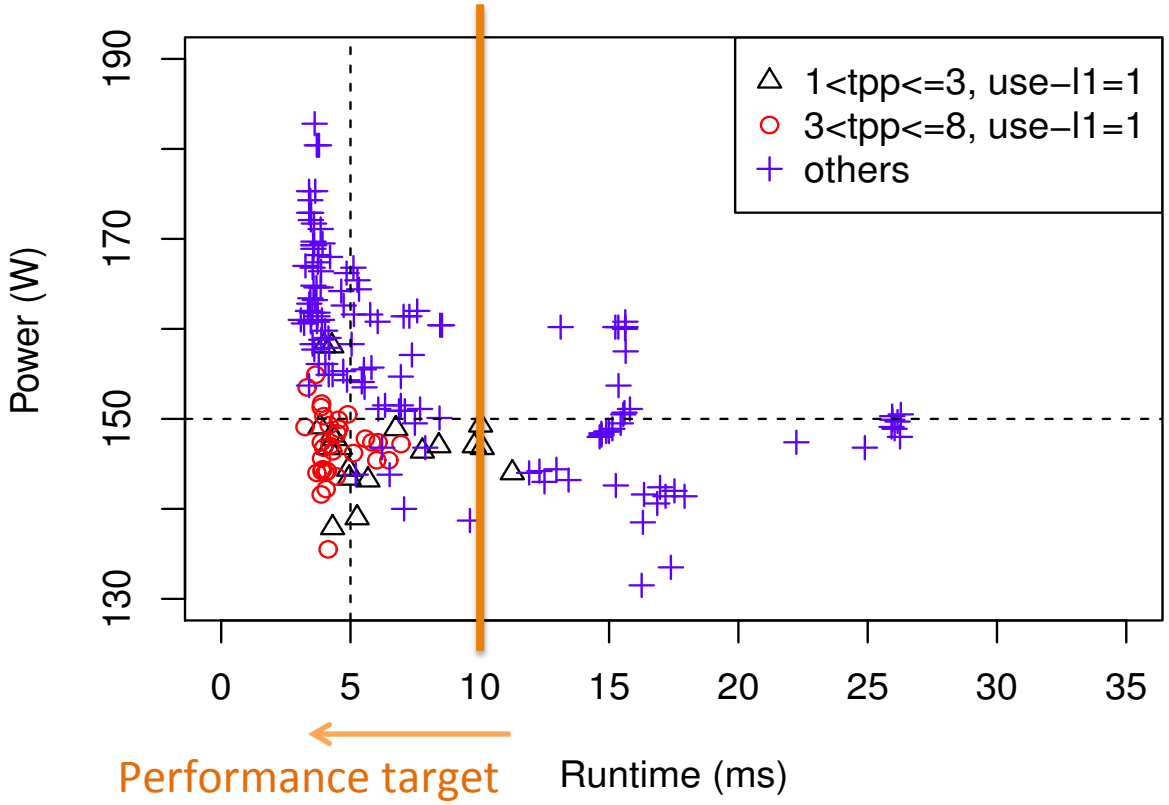


x ≤ 15, y ≤ 5 for AMD, 6.3× faster

x > 7, y > 5 for NVIDIA, 1.3× faster

- Define “use NVIDIA GPU” as a binary design parameter
- Support many different cross-platform optimization scenarios (see paper)

Case Study: Power/Performance Co-design



- Sliding performance targets disable or enable different parts of the design space
- Can look for Pareto optimal designs easily

Conclusion

- Starchart: an automated partitioning-based design space explorer
  - Handles real-system variance and high nonlinearity
  - > 300X exploration time speedup
  - Can be applied to CPU programs as well
- Subspace-based approaches useful for many real-world power/performance tuning problems
  - Cross-platform optimization: 6.3X faster than default
  - Power/performance trade-off: save 47W out of 180W with < 10% performance loss

Tool release: http://www.princeton.edu/~wjia/starchart

THANK YOU!