MACHINE LEARNING FOR PERFORMANCE AND POWER MODELING OF HETEROGENEOUS SYSTEMS

JOSEPH L. GREATHOUSE, GABRIEL H. LOH

ADVANCED MICRO DEVICES, INC.

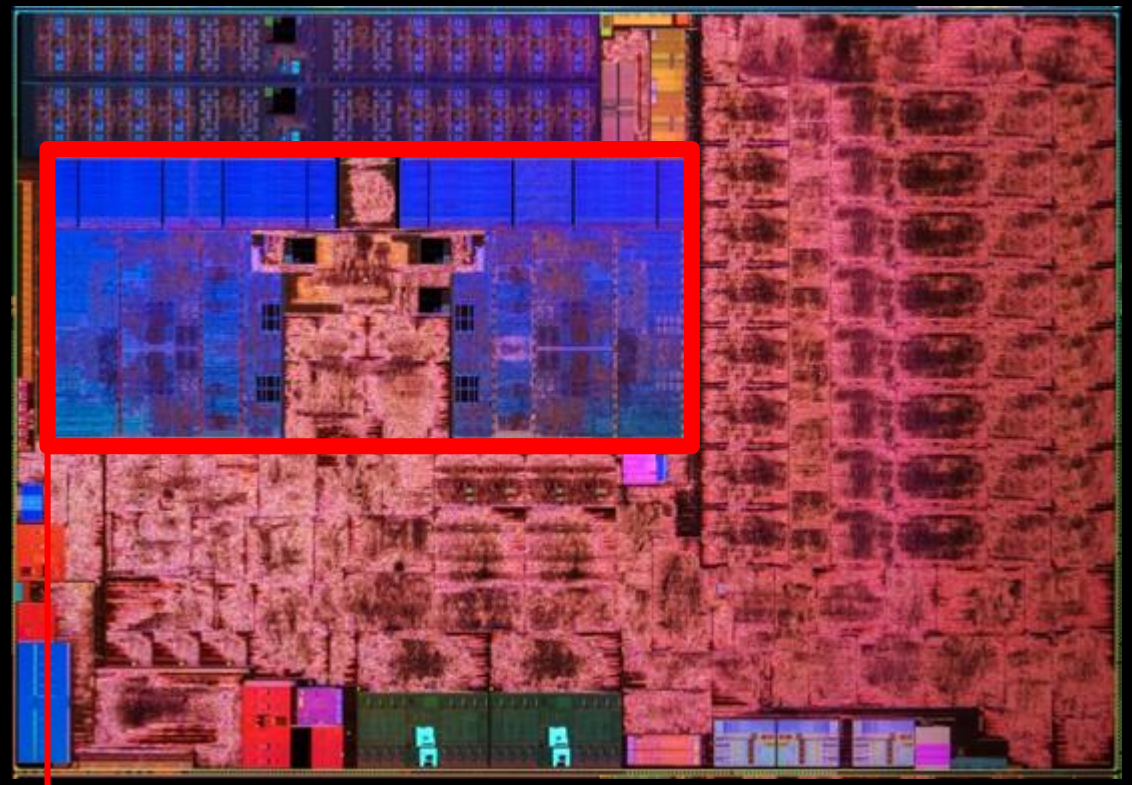
LARGE DESIGN SPACE FOR HETEROGENEOUS SYSTEMS

6th Gen. AMD A-Series Processor “Carrizo”

High-Level System Design Points

LARGE DESIGN SPACE FOR HETEROGENEOUS SYSTEMS

6th Gen. AMD A-Series Processor “Carrizo”



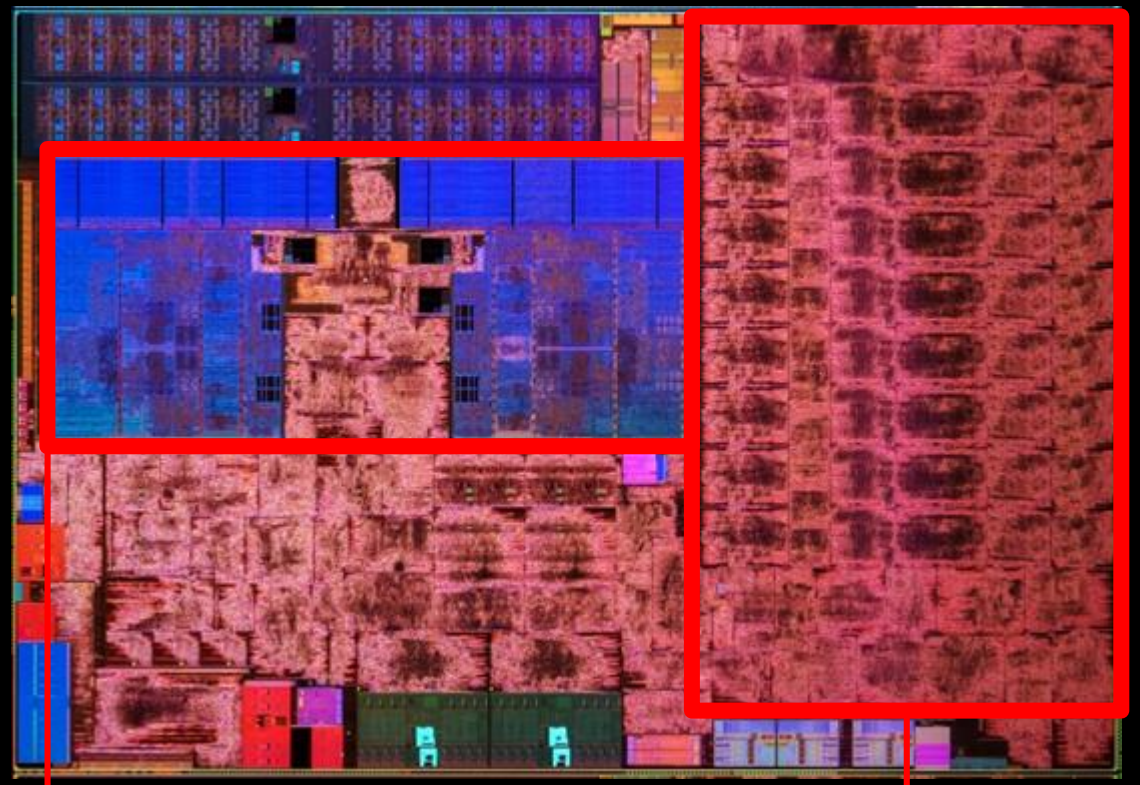
CPUs

High-Level System Design Points

- Some CPU design spaces:
  - How large?
  - How many CPUs?
  - How fast should the CPUs run?
  - How much power should it use?

LARGE DESIGN SPACE FOR HETEROGENEOUS SYSTEMS

6th Gen. AMD A-Series Processor “Carrizo”

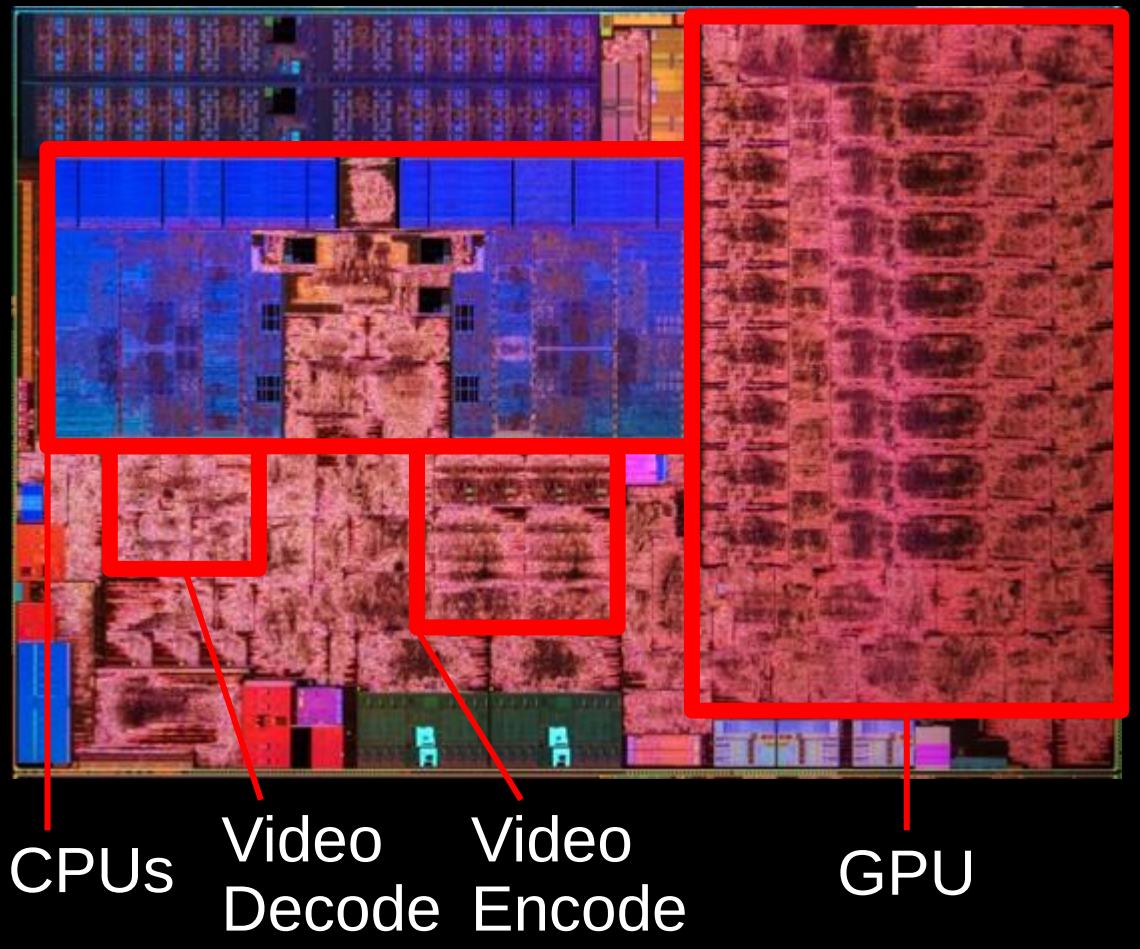


High-Level System Design Points

- Some CPU design spaces:
  - How large?
  - How many CPUs?
  - How fast should the CPUs run?
  - How much power should it use?
- Some GPU design spaces:
  - How much parallelism?
  - How fast should it run?
  - How much power should it use?

LARGE DESIGN SPACE FOR HETEROGENEOUS SYSTEMS

6th Gen. AMD A-Series Processor “Carrizo”

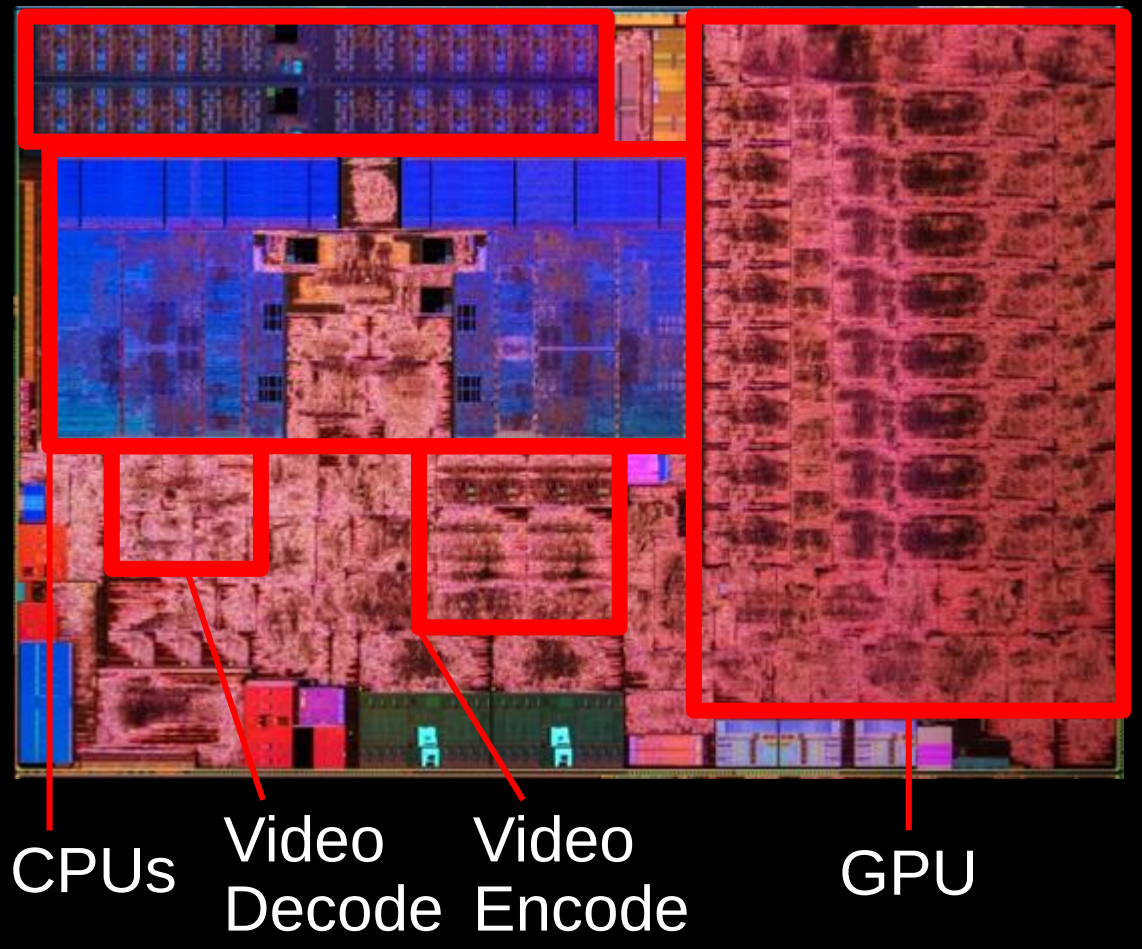


High-Level System Design Points

- Some CPU design spaces:
  - How large?
  - How many CPUs?
  - How fast should the CPUs run?
  - How much power should it use?
- Some GPU design spaces:
  - How much parallelism?
  - How fast should it run?
  - How much power should it use?
- What accelerators are needed?

LARGE DESIGN SPACE FOR HETEROGENEOUS SYSTEMS

6th Gen. AMD A-Series Processor “Carrizo”



High-Level System Design Points

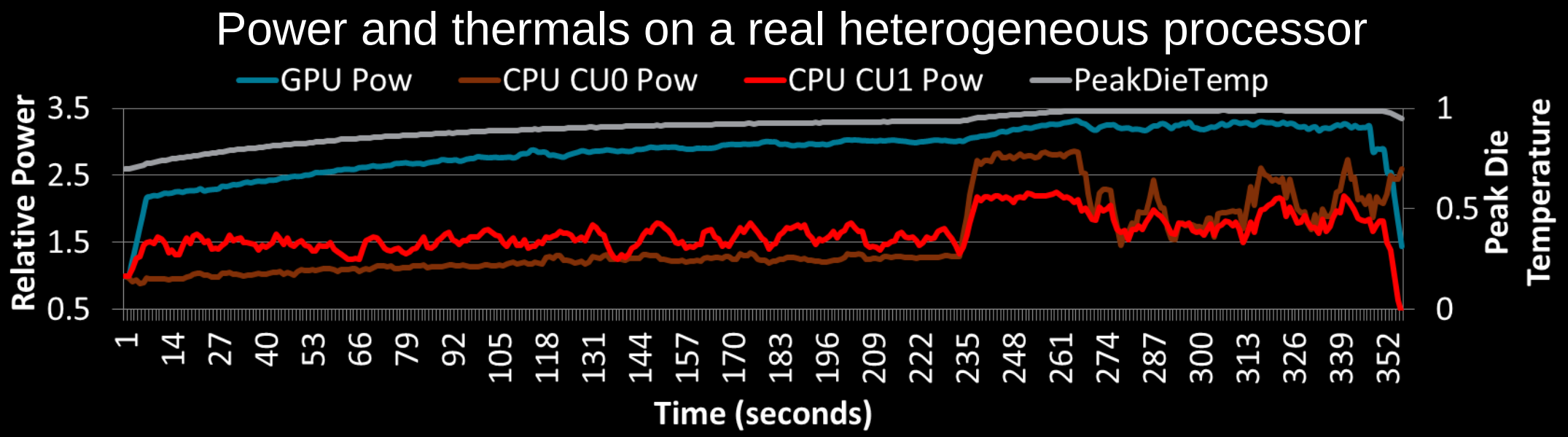
- Some CPU design spaces:
  - How large?
  - How many CPUs?
  - How fast should the CPUs run?
  - How much power should it use?
- Some GPU design spaces:
  - How much parallelism?
  - How fast should it run?
  - How much power should it use?
- What accelerators are needed?
- How much bandwidth needed?

EXAMPLE OF A DESIGN SPACE EXPLORATION

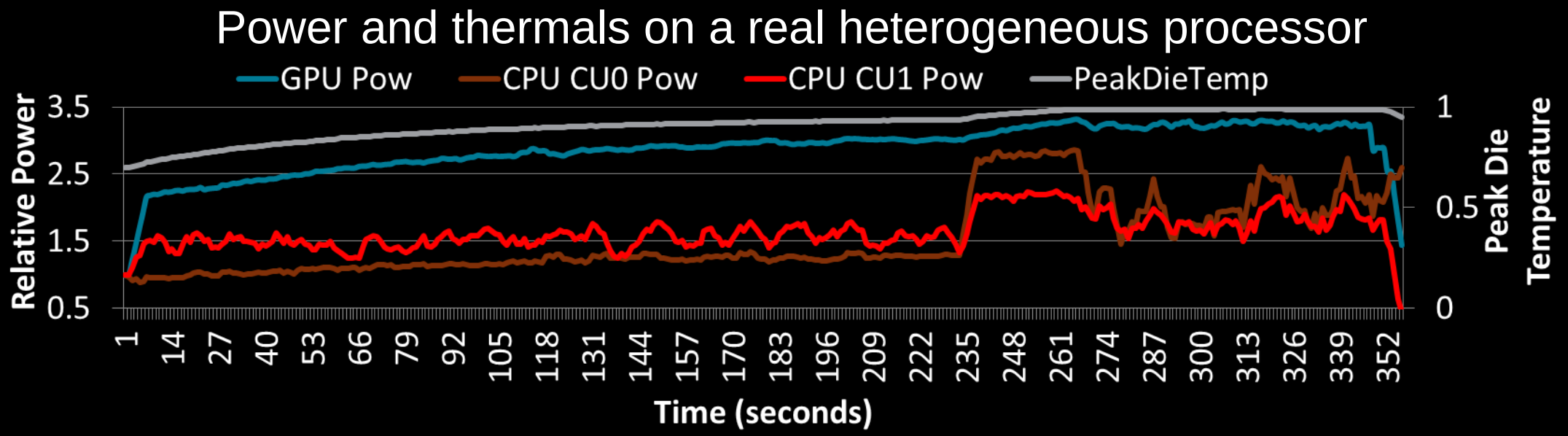
Various Designs Using AMD GPUs

Name	CUs	Max Freq. (MHz)	Max DRAM BW (GB/s)
AMD E1-6010 APU	2	350	11
AMD A10-7850K APU	8	720	35
Microsoft Xbox One™ Processor	12	853	68
Sony PlayStation® 4 Processor	18	800	176
AMD Radeon™ R9-280X	32	1000	286
AMD Radeon™ R9-290X	44	1000	352
AMD Radeon™ R9 Fury X	64	1000	512

DESIGN SPACE EXPLORATIONS REQUIRE LONG RUNTIMES

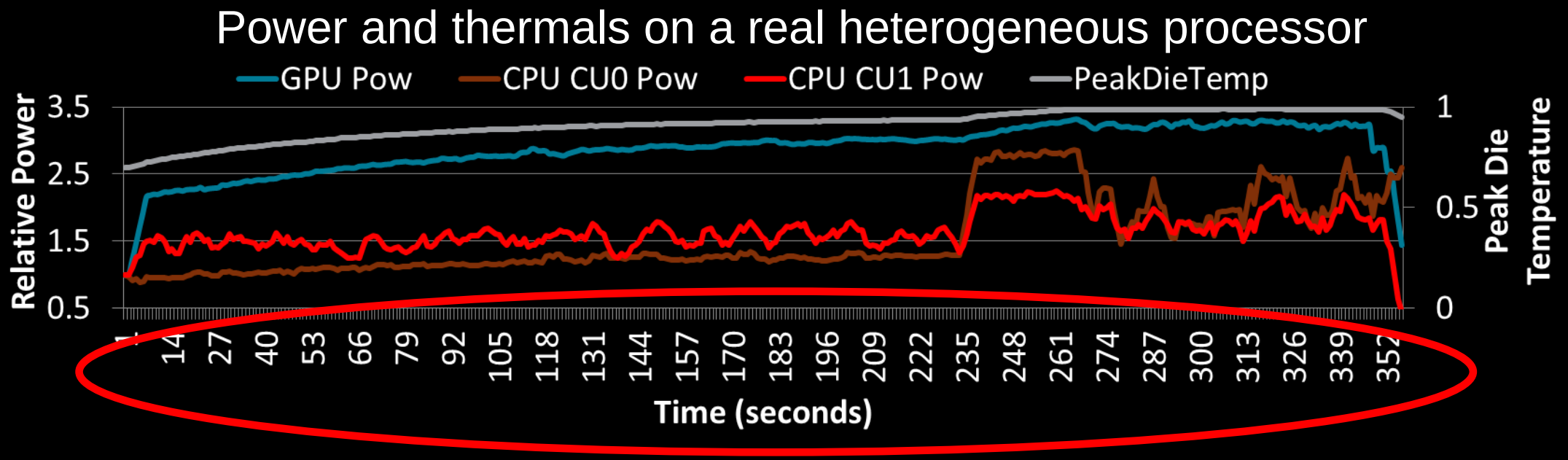


DESIGN SPACE EXPLORATIONS REQUIRE LONG RUNTIMES



- Real applications of interest are large and complex
  - Not microbenchmarks, can run for minutes or hours
  - Performance and power can rely on what happened in the past
  - Complex interactions between various heterogeneous devices

DESIGN SPACE EXPLORATIONS REQUIRE LONG RUNTIMES



~2.5 trillion CPU instructions, ~60 trillion GPU operations

- Real applications of interest are large and complex
  - Not microbenchmarks, can run for minutes or hours
  - Performance and power can rely on what happened in the past
  - Complex interactions between various heterogeneous devices

DESIGN-SPACE EXPLORATION HARD ON EXISTING PLATFORMS

- Microarchitecture simulators (e.g., gem5, Multi2Sim, SESC, GPGPU-Sim)
  - Excellent for low-level details.
  - Too slow for broad design space explorations of full applications:
    - 6 minutes * 60s/min * 4 CPU cores * ~1.75GHz (AMD A8-4555M) +
      6 minutes * 60s/min * 6 CUs * 64 FPU/CU * 425MHz (AMD Radeon™ HD 7600G) =
      ~60 trillion operations ≈ 1 year of simulation time @ 2 MIPS

DESIGN-SPACE EXPLORATION HARD ON EXISTING PLATFORMS

- Microarchitecture simulators (e.g., gem5, Multi2Sim, SESC, GPGPU-Sim)
  - Excellent for low-level details.
  - Too slow for broad design space explorations of full applications:
    - 6 minutes * 60s/min * 4 CPU cores * ~1.75GHz (AMD A8-4555M) +
      6 minutes * 60s/min * 6 CUs * 64 FPU/CU * 425MHz (AMD Radeon™ HD 7600G) =
      ~60 trillion operations ≈ 1 year of simulation time @ 2 MIPS
- Emulators (e.g., Cadence Palladium, Synopsys ZeBu, Mentor Veloce, IBM AWAN)
  - Much faster than SW simulation, just as detailed, but require a nearly-complete design

DESIGN-SPACE EXPLORATION HARD ON EXISTING PLATFORMS

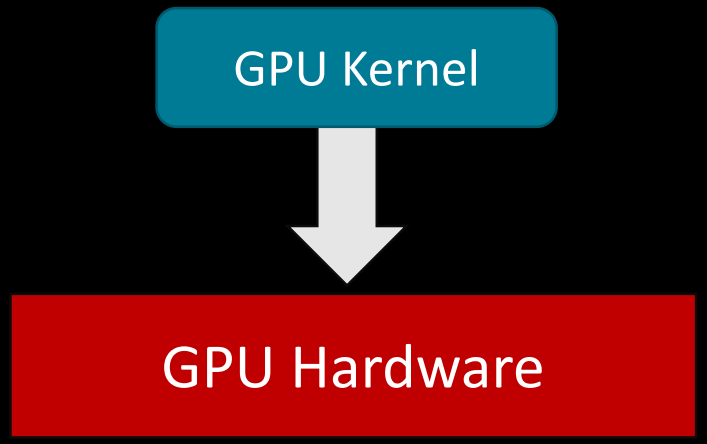
- Microarchitecture simulators (e.g., gem5, Multi2Sim, SESC, GPGPU-Sim)
  - Excellent for low-level details.
  - Too slow for broad design space explorations of full applications:
    - 6 minutes * 60s/min * 4 CPU cores * ~1.75GHz (AMD A8-4555M) +
      6 minutes * 60s/min * 6 CUs * 64 FPU/CU * 425MHz (AMD Radeon™ HD 7600G) =
      ~60 trillion operations ≈ 1 year of simulation time @ 2 MIPS
- Emulators (e.g., Cadence Palladium, Synopsys ZeBu, Mentor Veloce, IBM AWAN)
  - Much faster than SW simulation, just as detailed, but require a nearly-complete design
- Functional simulators (e.g., AMD SimNow, Simics, QEMU, etc.)
  - Faster than microarchitectural simulators, good for software bring-up
  - No relation to hardware performance

DESIGN-SPACE EXPLORATION HARD ON EXISTING PLATFORMS

- Microarchitecture simulators (e.g., gem5, Multi2Sim, SESC, GPGPU-Sim)
  - Excellent for low-level details.
  - Too slow for broad design space explorations of full applications:
    - 6 minutes * 60s/min * 4 CPU cores * ~1.75GHz (AMD A8-4555M) +
      6 minutes * 60s/min * 6 CUs * 64 FPU/CU * 425MHz (AMD Radeon™ HD 7600G) =
      ~60 trillion operations ≈ 1 year of simulation time @ 2 MIPS
- Emulators (e.g., Cadence Palladium, Synopsys ZeBu, Mentor Veloce, IBM AWAN)
  - Much faster than SW simulation, just as detailed, but require a nearly-complete design
- Functional simulators (e.g., AMD SimNow, Simics, QEMU, etc.)
  - Faster than microarchitectural simulators, good for software bring-up
  - No relation to hardware performance
- Spreadsheet analyses
  - Great for first-order analyses. Much faster and easier than lower-level simulators
  - Difficult to analyze application differences.

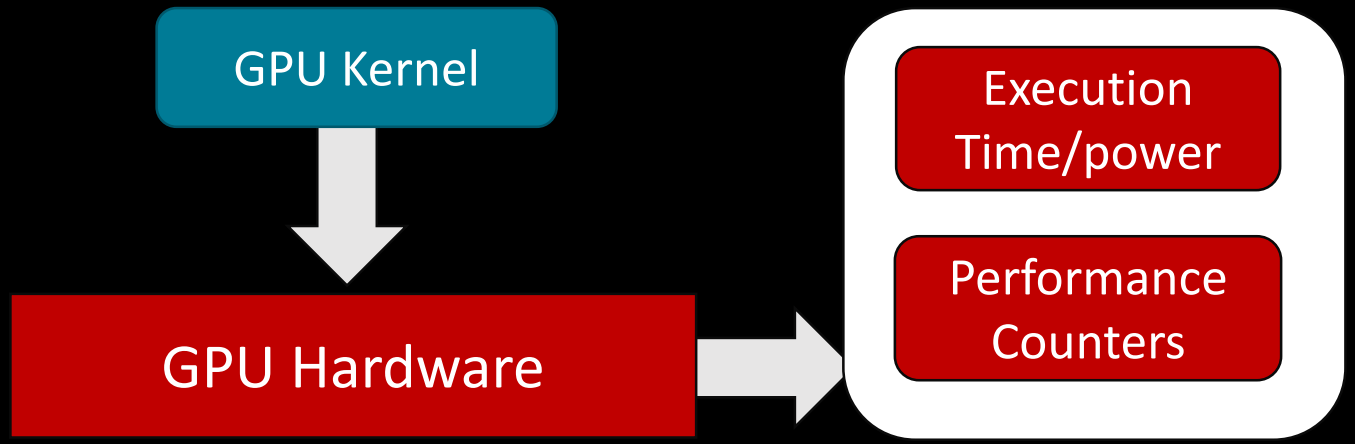
DRIVE DESIGN SPACE EXPLORATION FROM REAL HARDWARE

- Gather from running application of interest on real hardware:



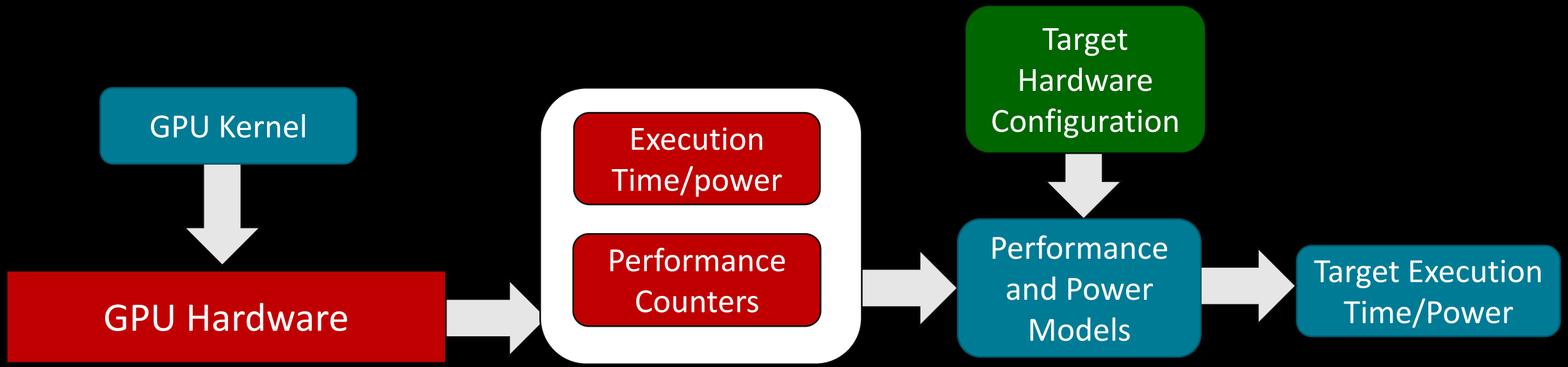
DRIVE DESIGN SPACE EXPLORATION FROM REAL HARDWARE

- Gather from running application of interest on real hardware:
  - Measured performance and power
  - Information about how the application used the hardware (e.g., performance counters)



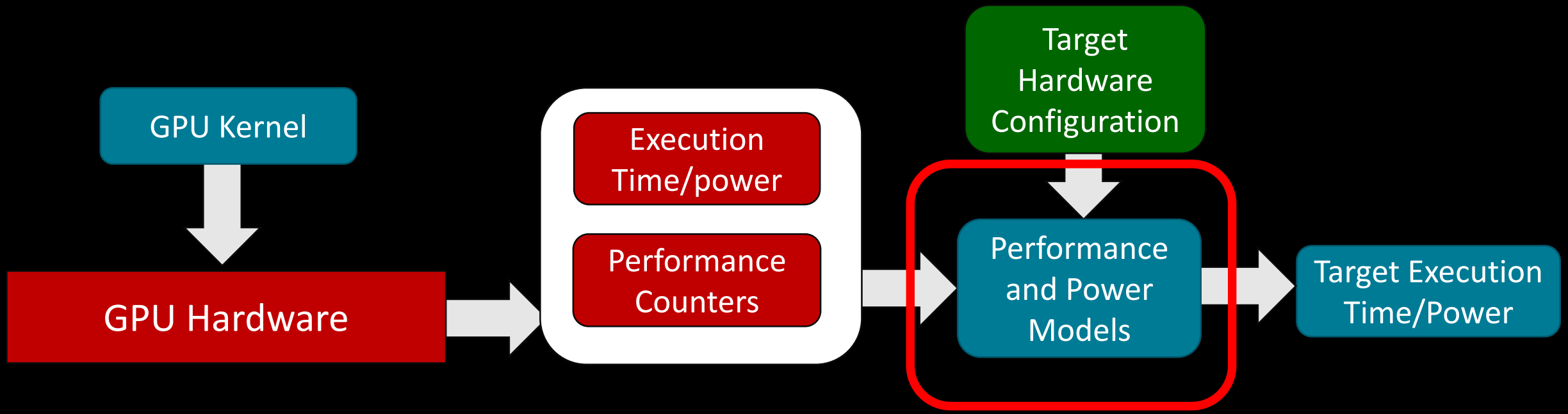
DRIVE DESIGN SPACE EXPLORATION FROM REAL HARDWARE

- Gather from running application of interest on real hardware:
  - Measured performance and power
  - Information about how the application used the hardware (e.g., performance counters)
- Estimate performance and power for different hardware design points



DRIVE DESIGN SPACE EXPLORATION FROM REAL HARDWARE

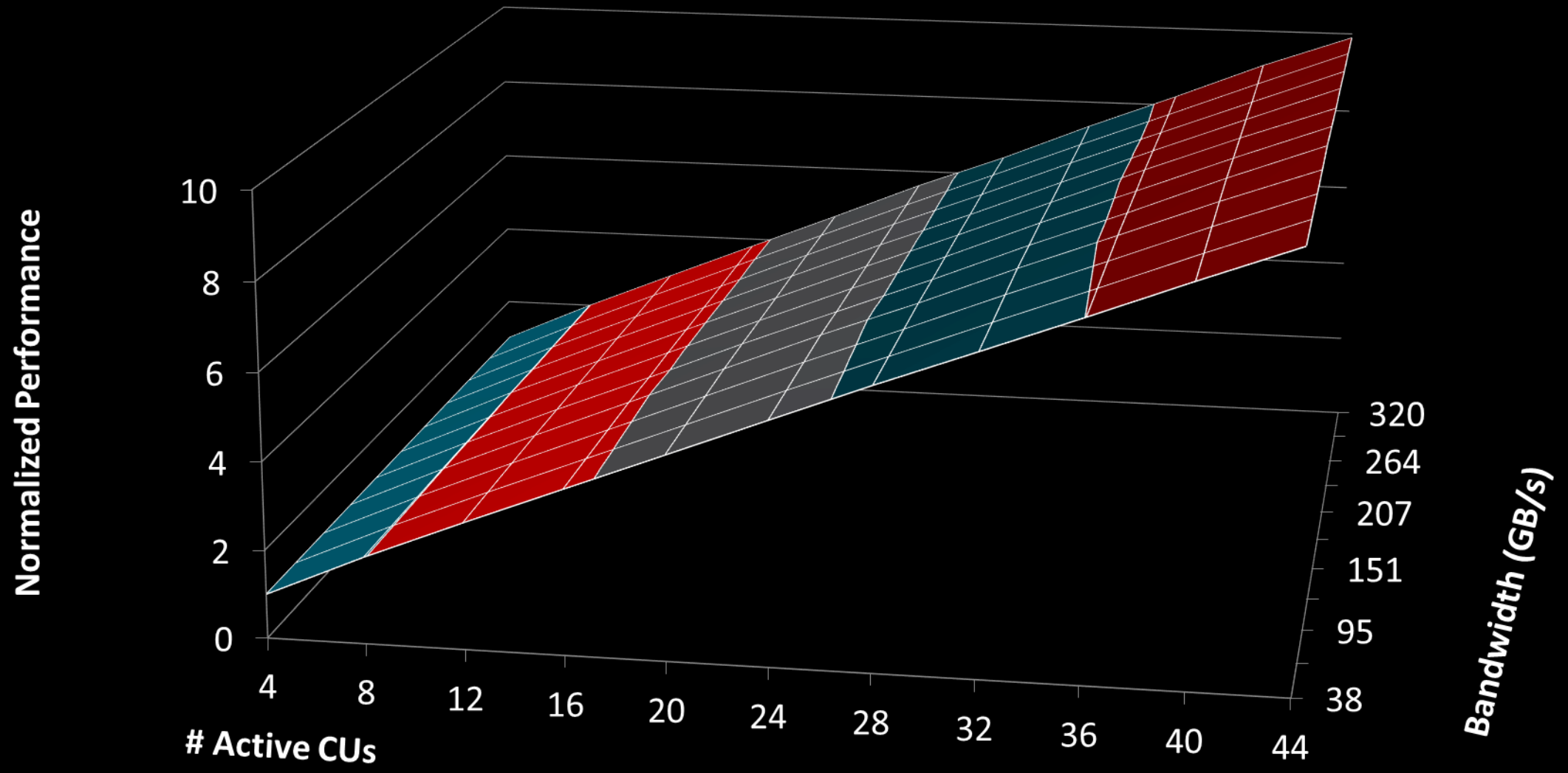
- Gather from running application of interest on real hardware:
  - Measured performance and power
  - Information about how the application used the hardware (e.g., performance counters)
- Estimate performance and power for different hardware design points



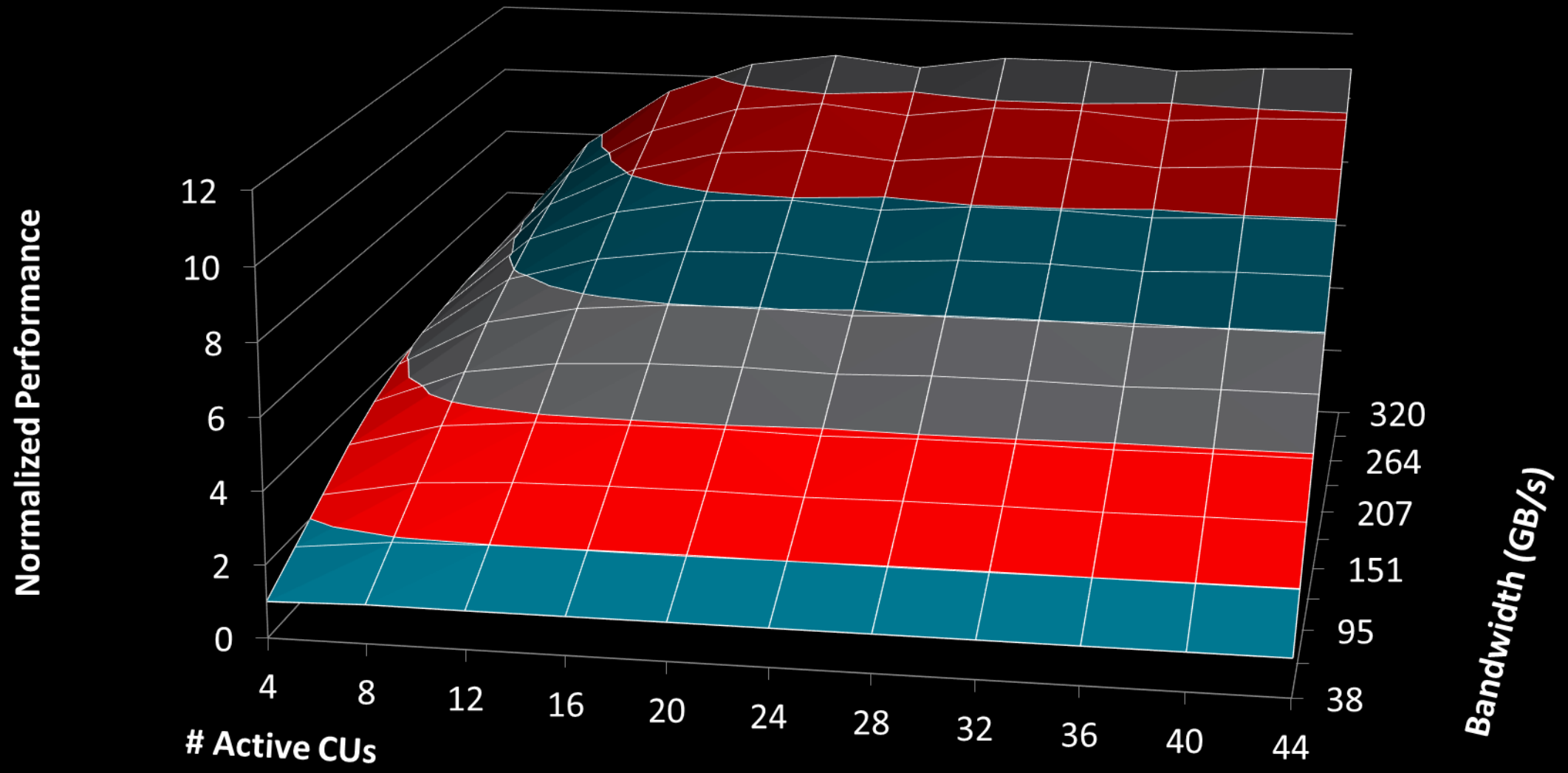
THE “HIGH-LEVEL” DESIGN SPACE EXPLORED IN THIS TALK

- Changes to the following GPU parameters:
  - Number of parallel compute units (CUs)
  - Core frequency
  - Memory bandwidth
- Changes to GPU kernel:
  - Performance
  - Dynamic power

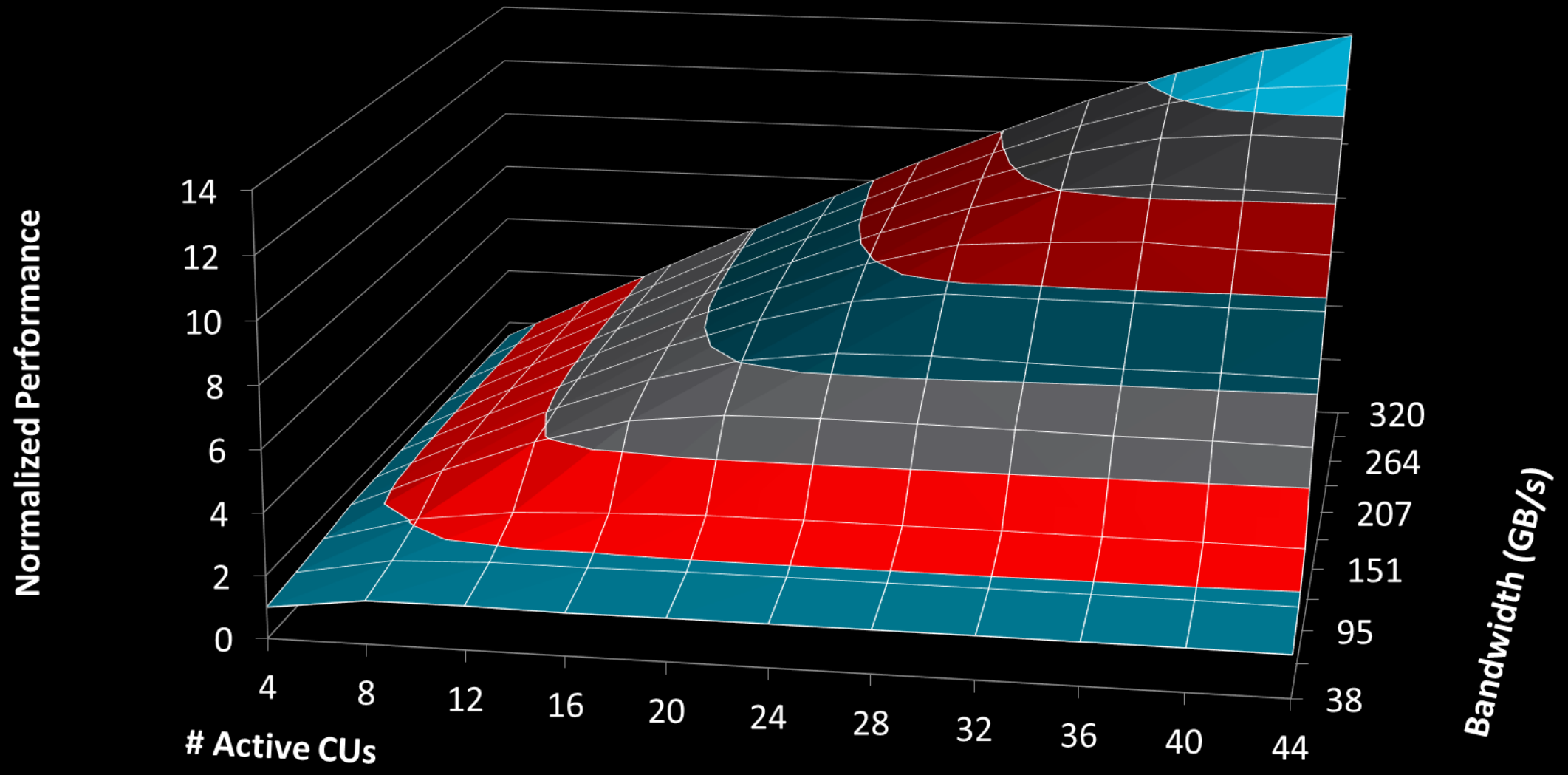
GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (1)



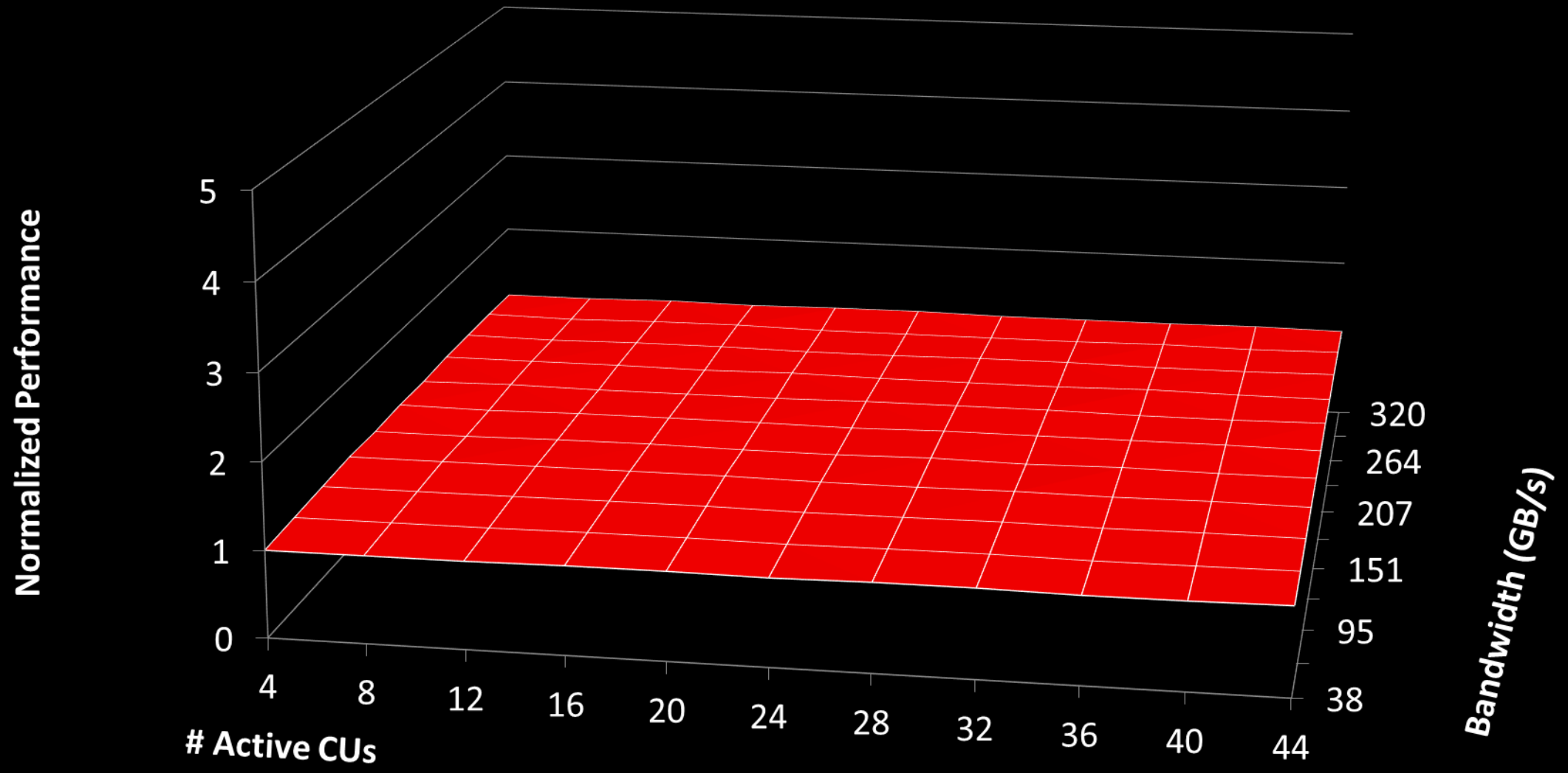
GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (2)



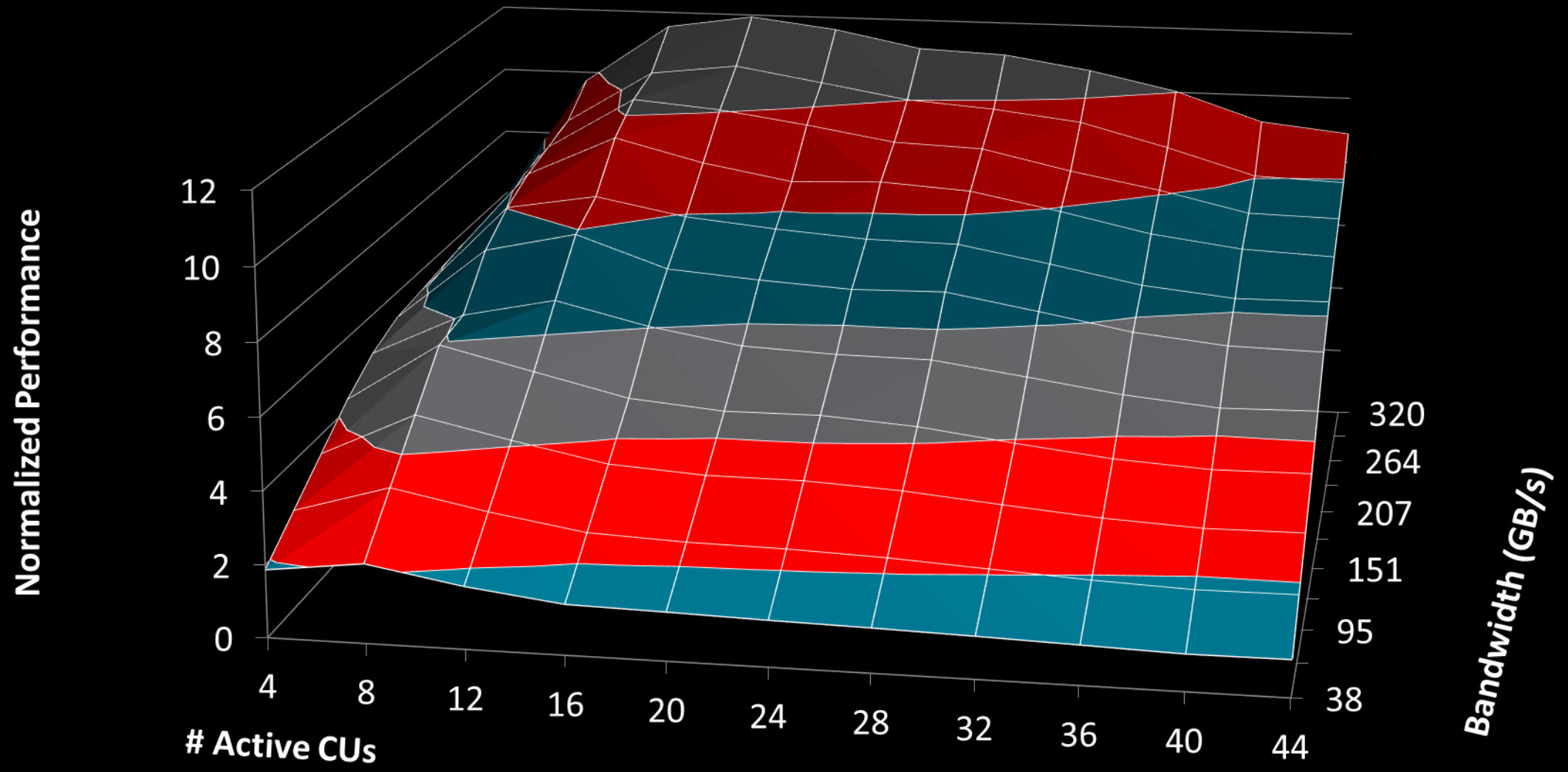
GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (3)



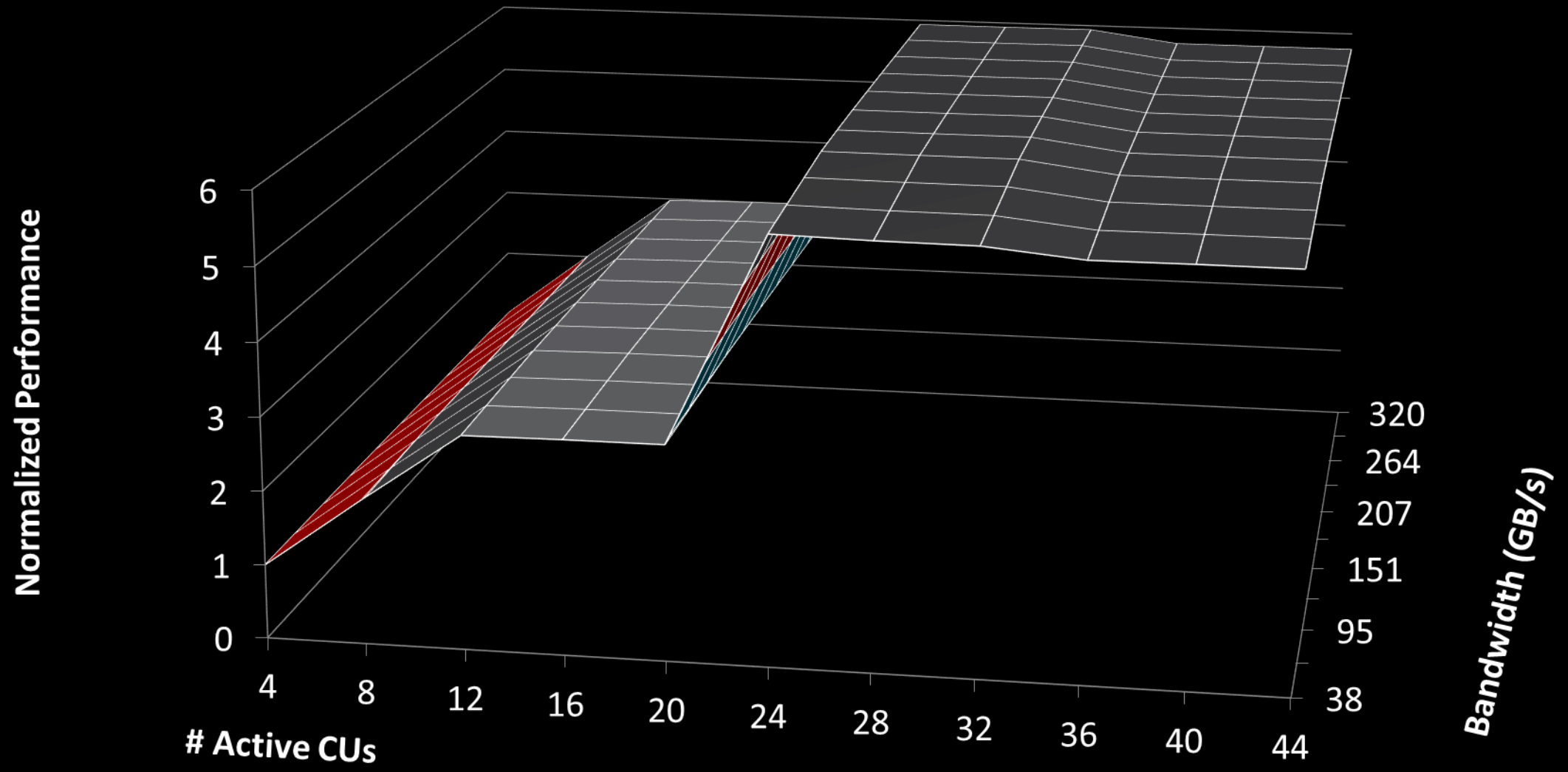
GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (4)



GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (5)

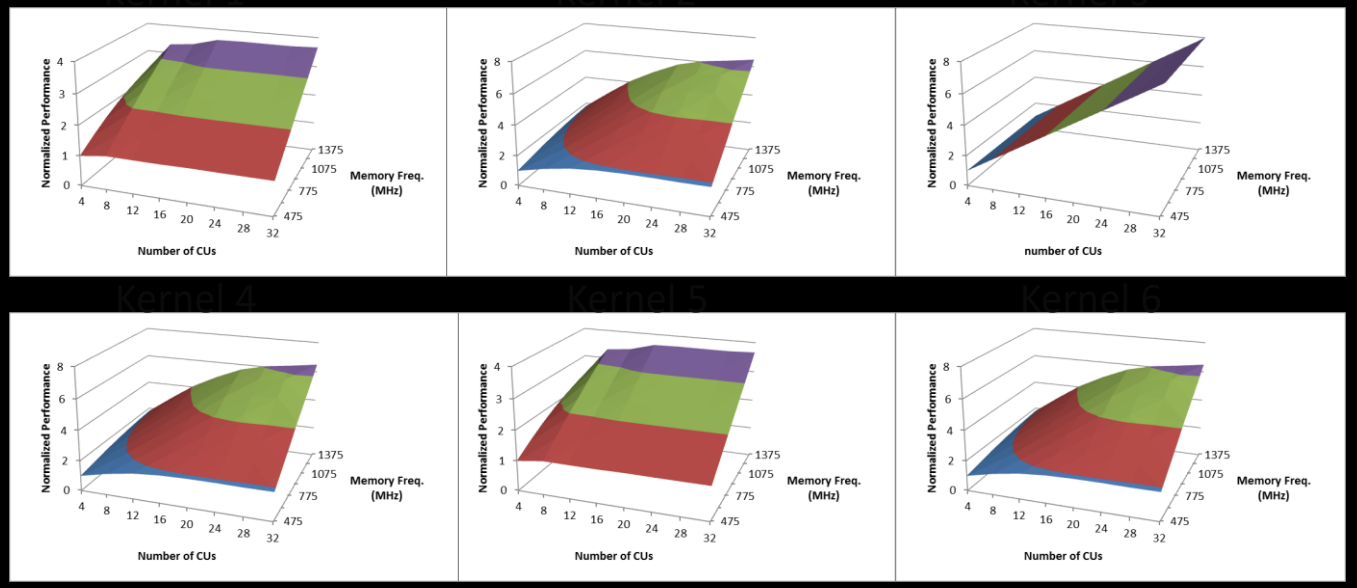


GPU KERNEL PERFORMANCE SCALING WITH HW DESIGNS (6)



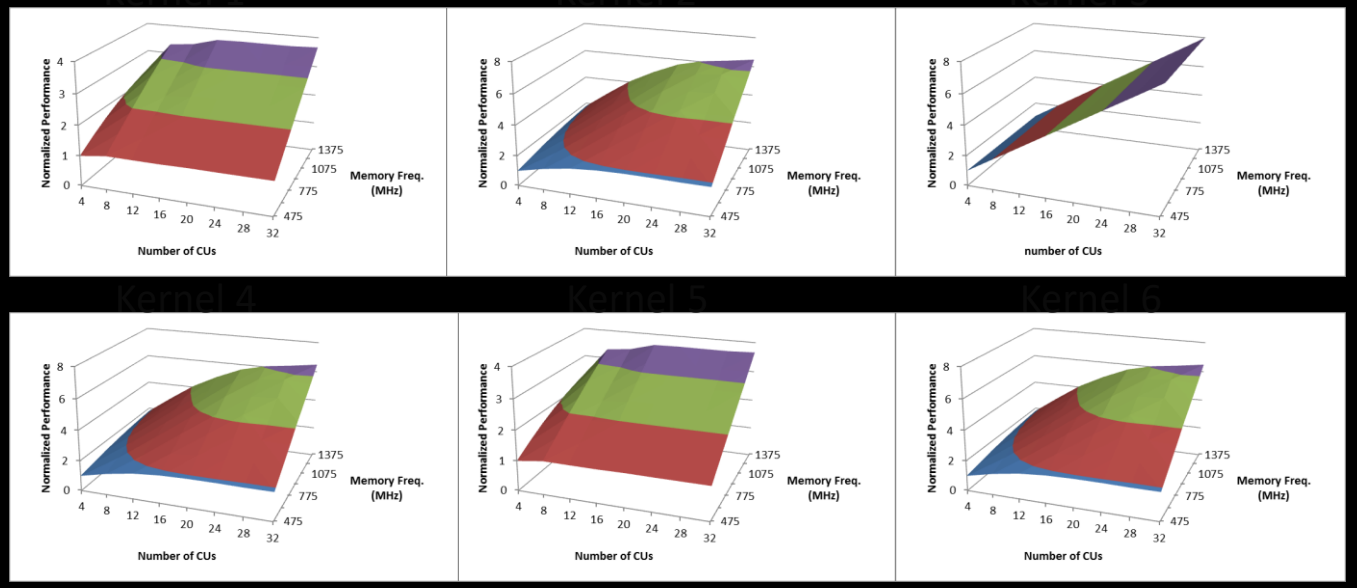
AUTOMATICALLY CLUSTERING SCALING CURVES

Training Set



AUTOMATICALLY CLUSTERING SCALING CURVES

Training Set



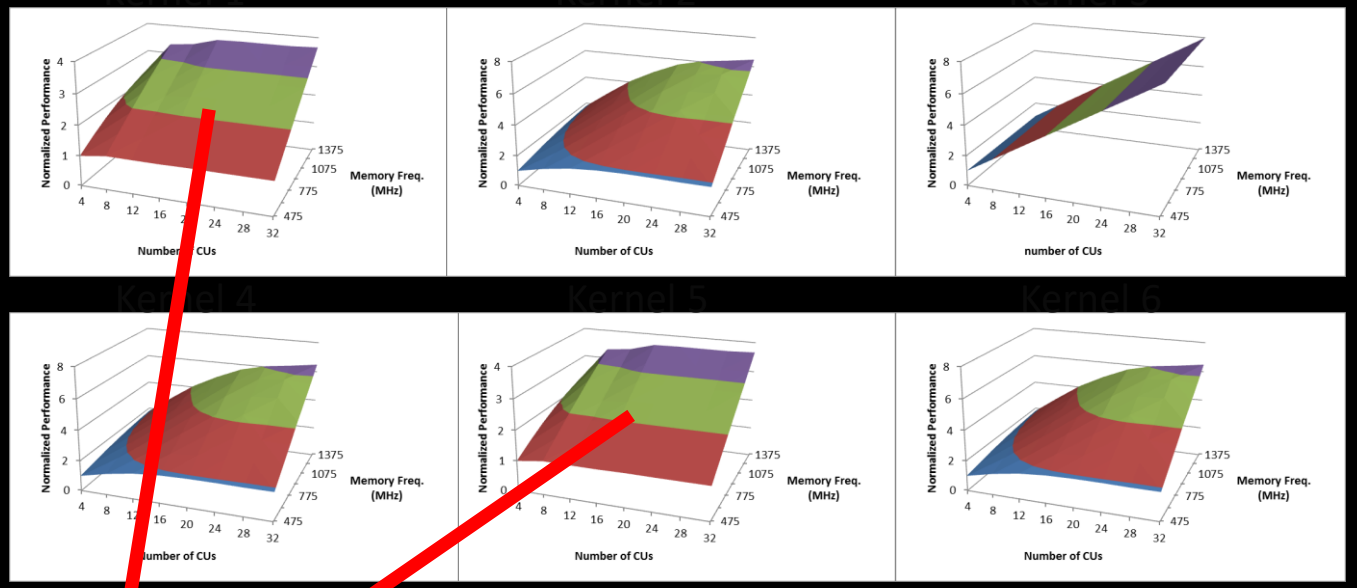
Cluster 1

Cluster 2

Cluster 3

AUTOMATICALLY CLUSTERING SCALING CURVES

Training Set



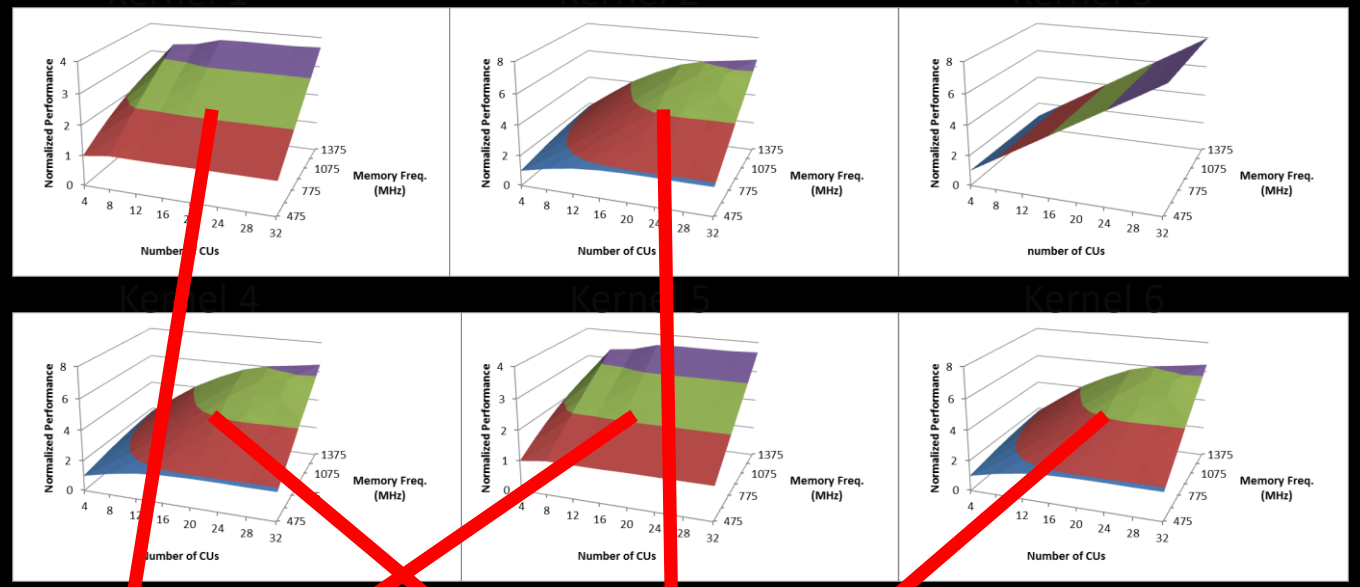
Cluster 1

Cluster 2

Cluster 3

AUTOMATICALLY CLUSTERING SCALING CURVES

Training Set



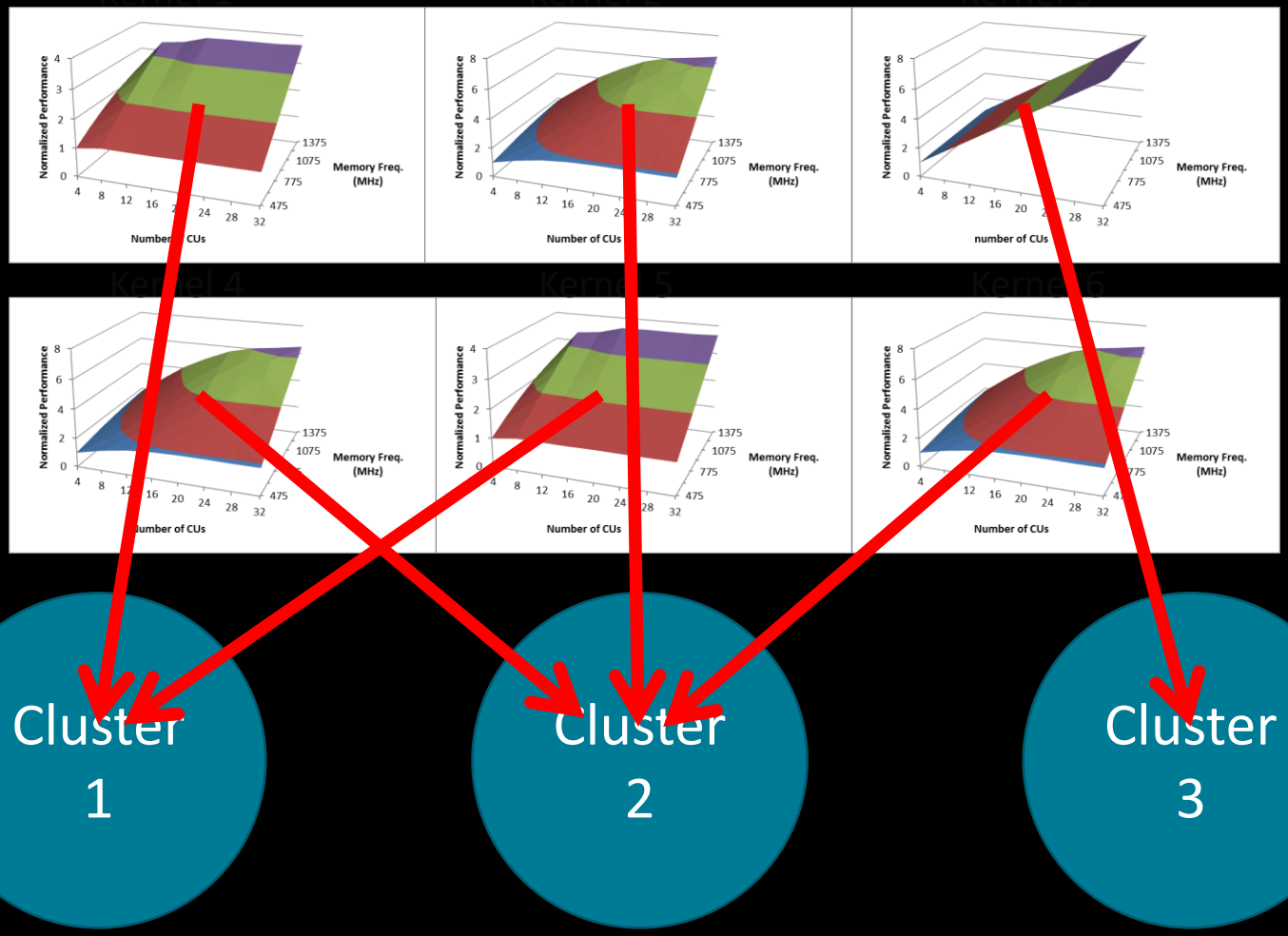
Cluster 1

Cluster 2

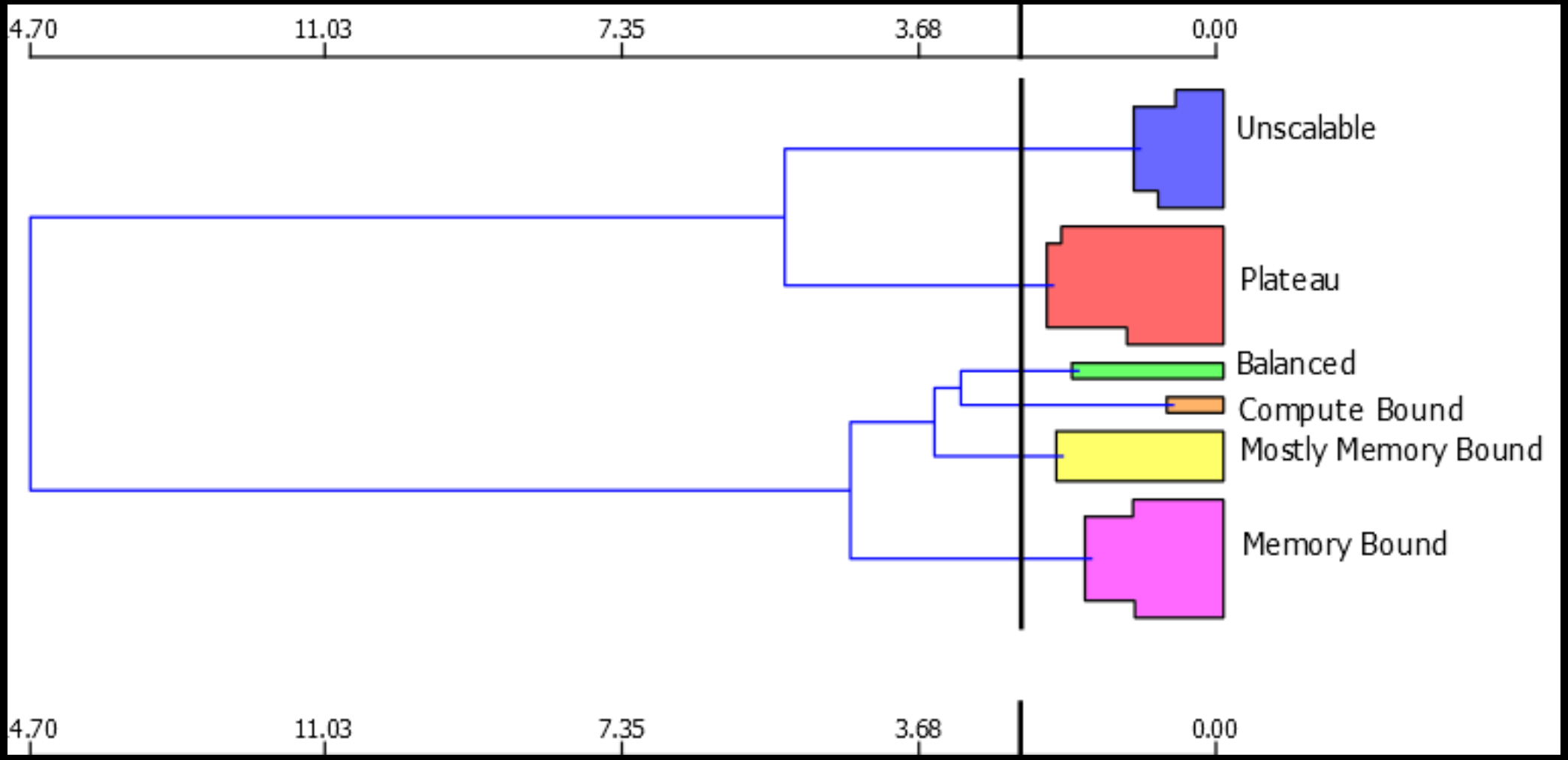
Cluster 3

AUTOMATICALLY CLUSTERING SCALING CURVES

Training Set



AUTOMATICALLY CLUSTERING SCALING CURVES



FINDING A SCALING CLUSTER OF AN UNKNOWN KERNEL

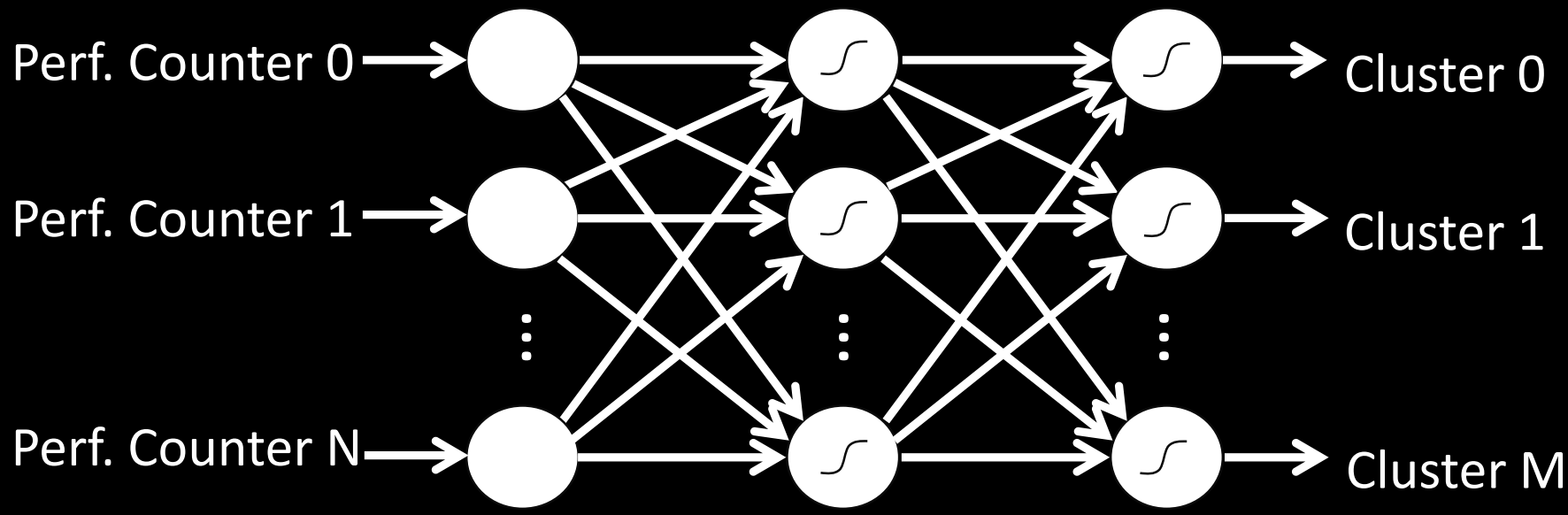
- Performance counters indicate how the kernel uses the hardware

FINDING A SCALING CLUSTER OF AN UNKNOWN KERNEL

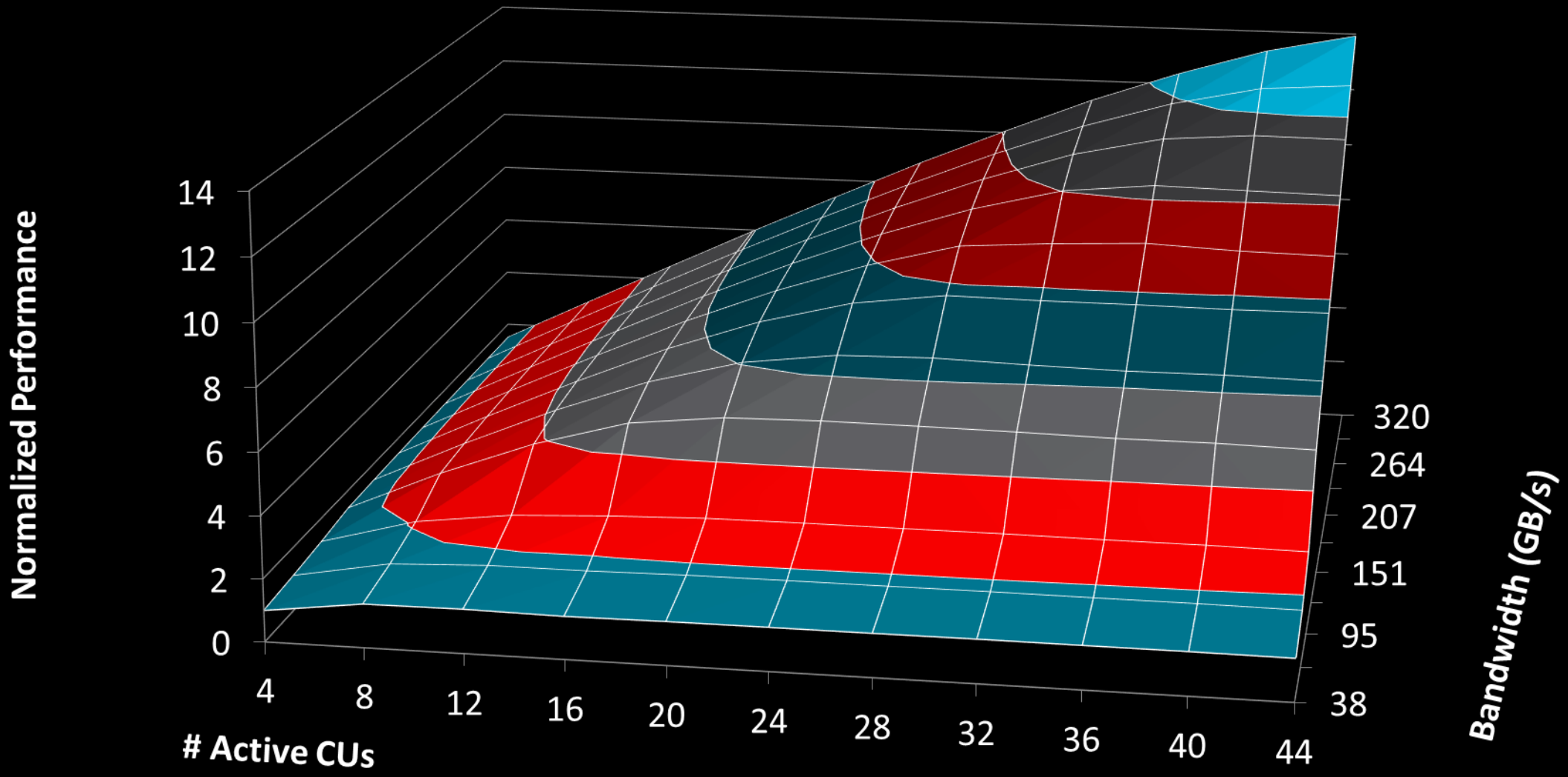
- Performance counters indicate how the kernel uses the hardware
  - Counters may thus help indicate which cluster a kernel is in **with only one measurement**

FINDING A SCALING CLUSTER OF AN UNKNOWN KERNEL

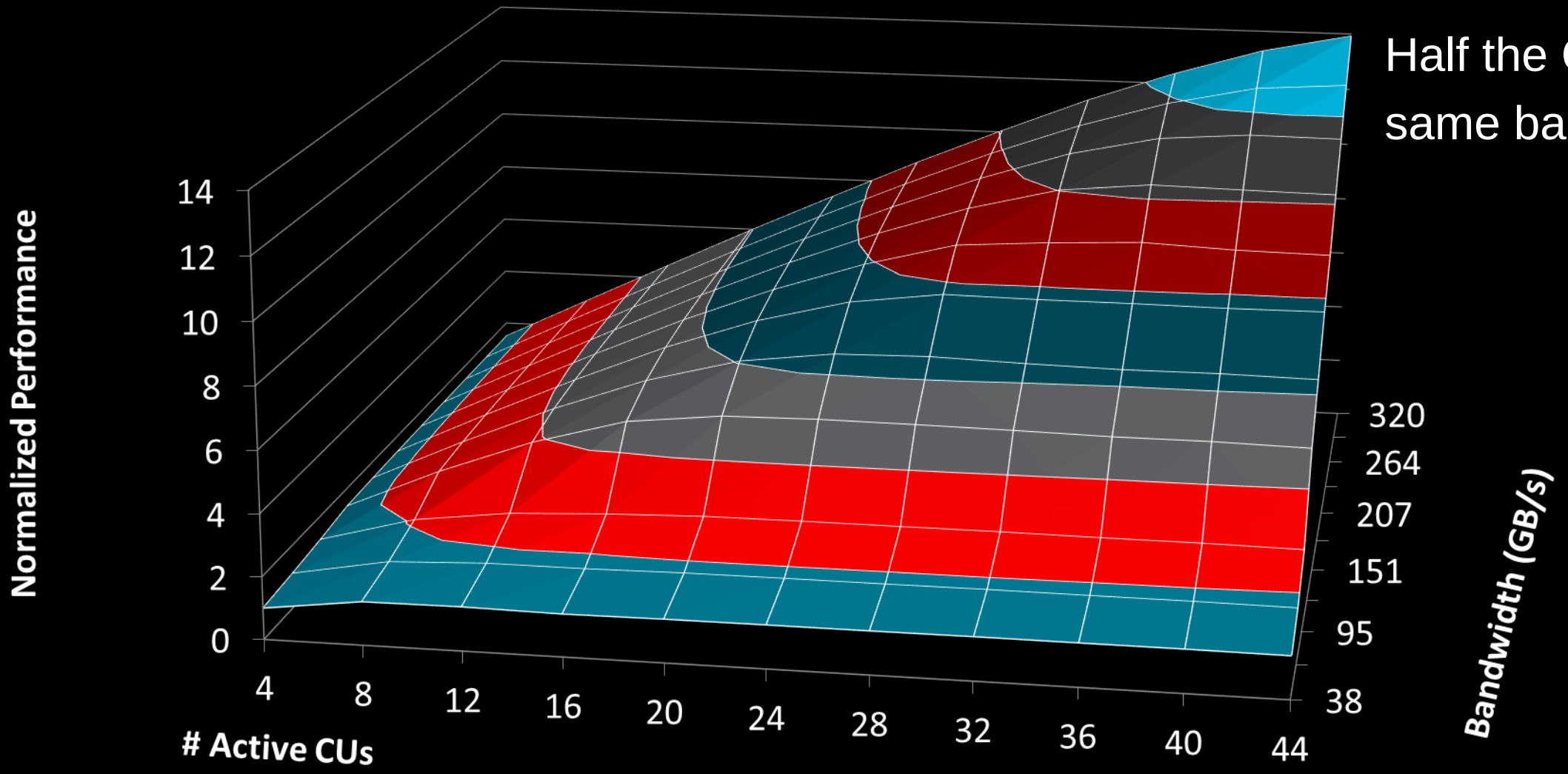
- Performance counters indicate how the kernel uses the hardware
  - Counters may thus help indicate which cluster a kernel is in **with only one measurement**



ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE

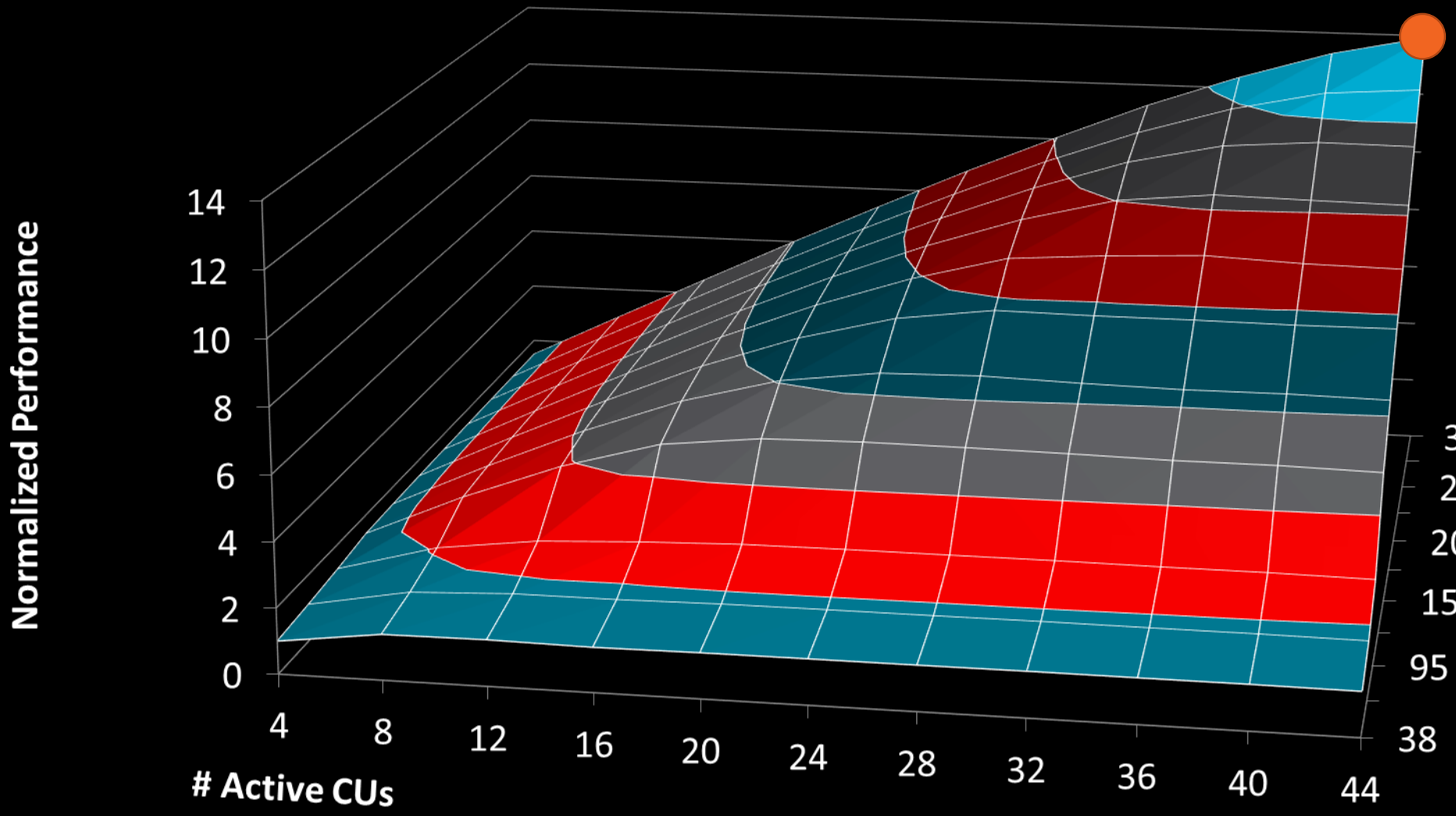


ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



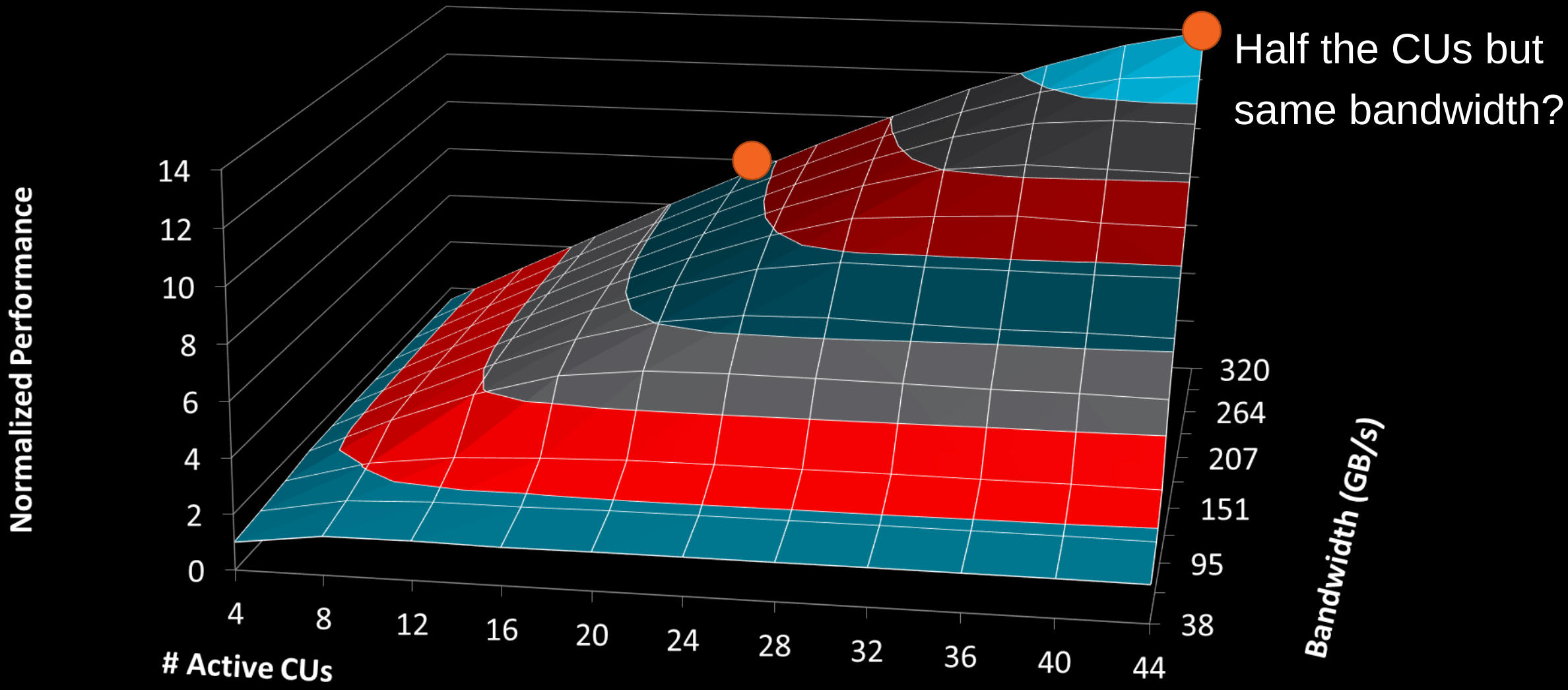
Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



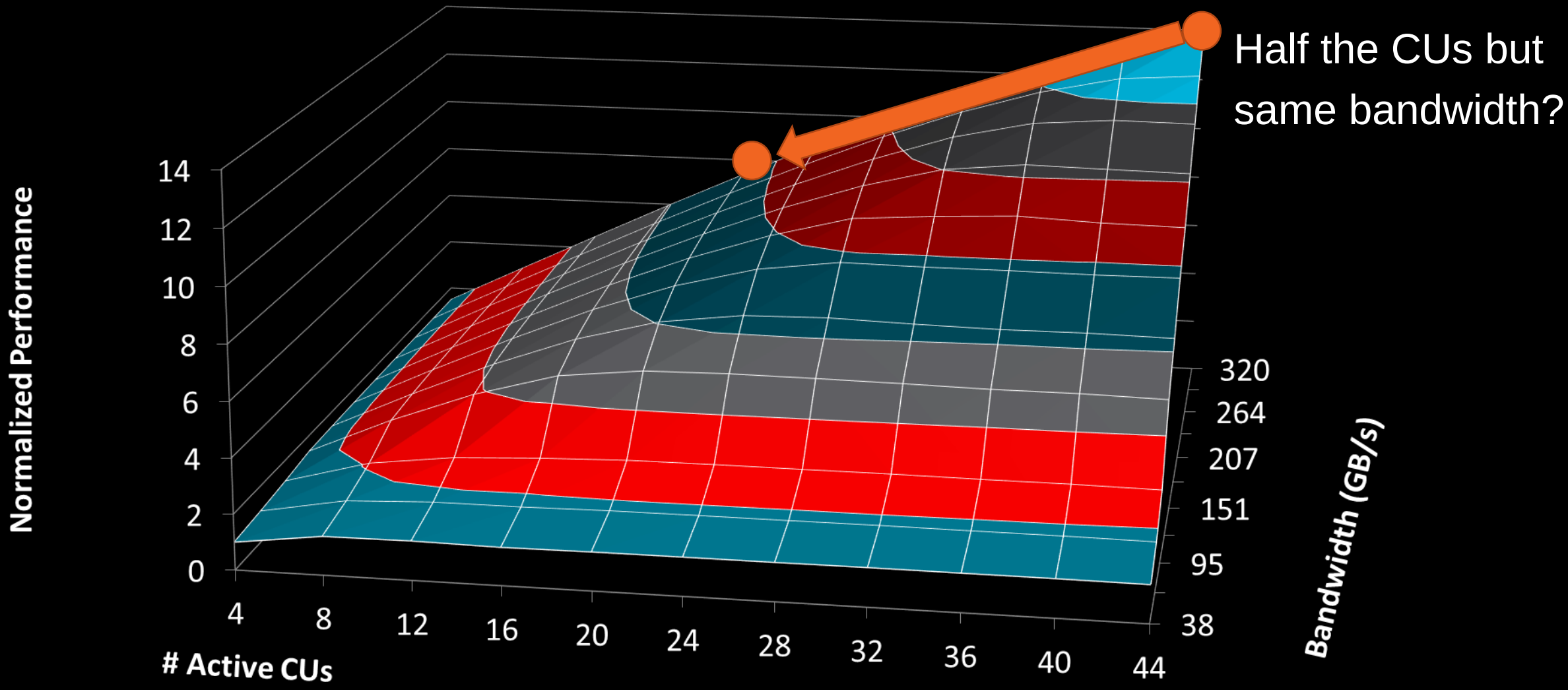
Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE

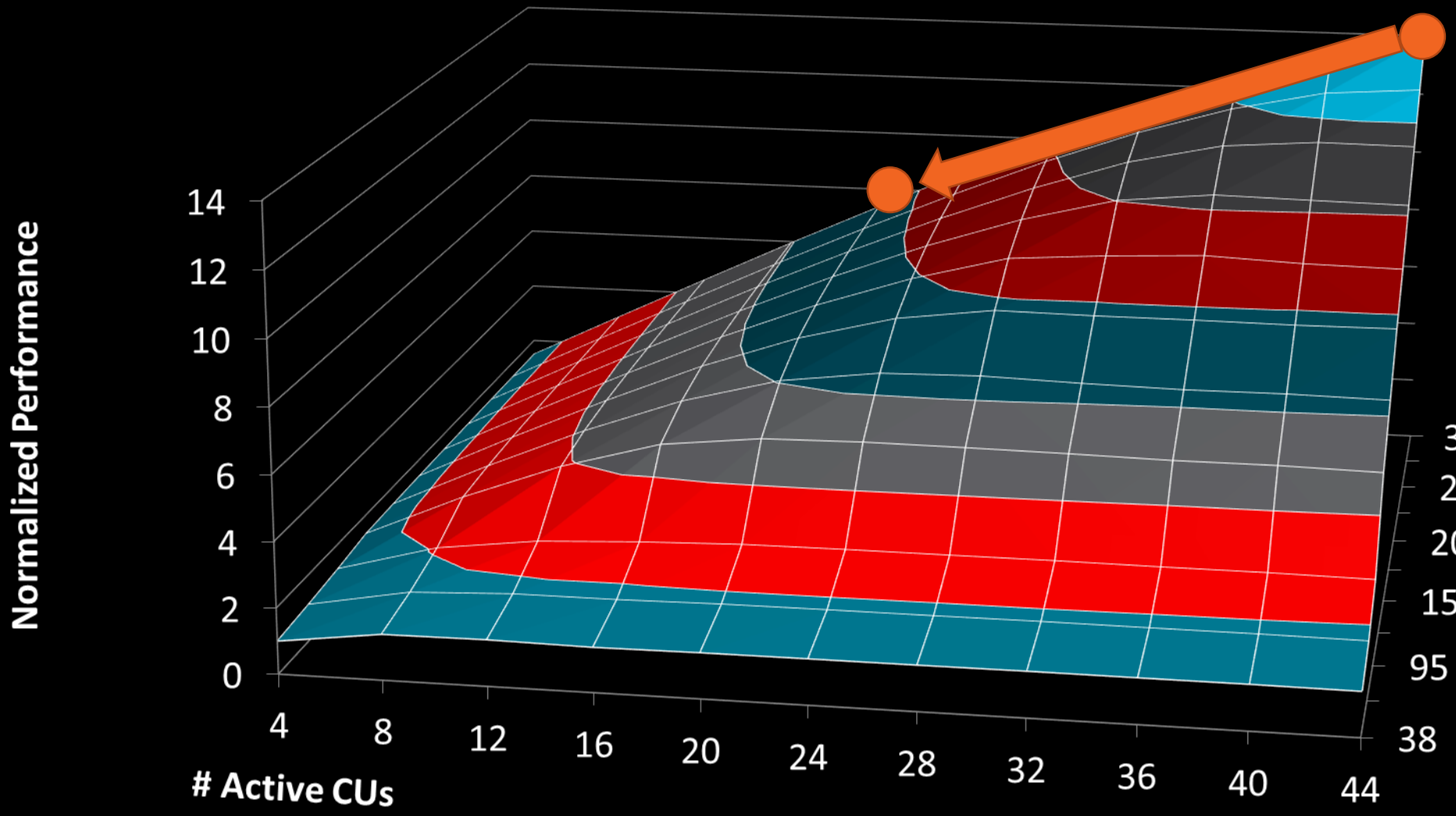


Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



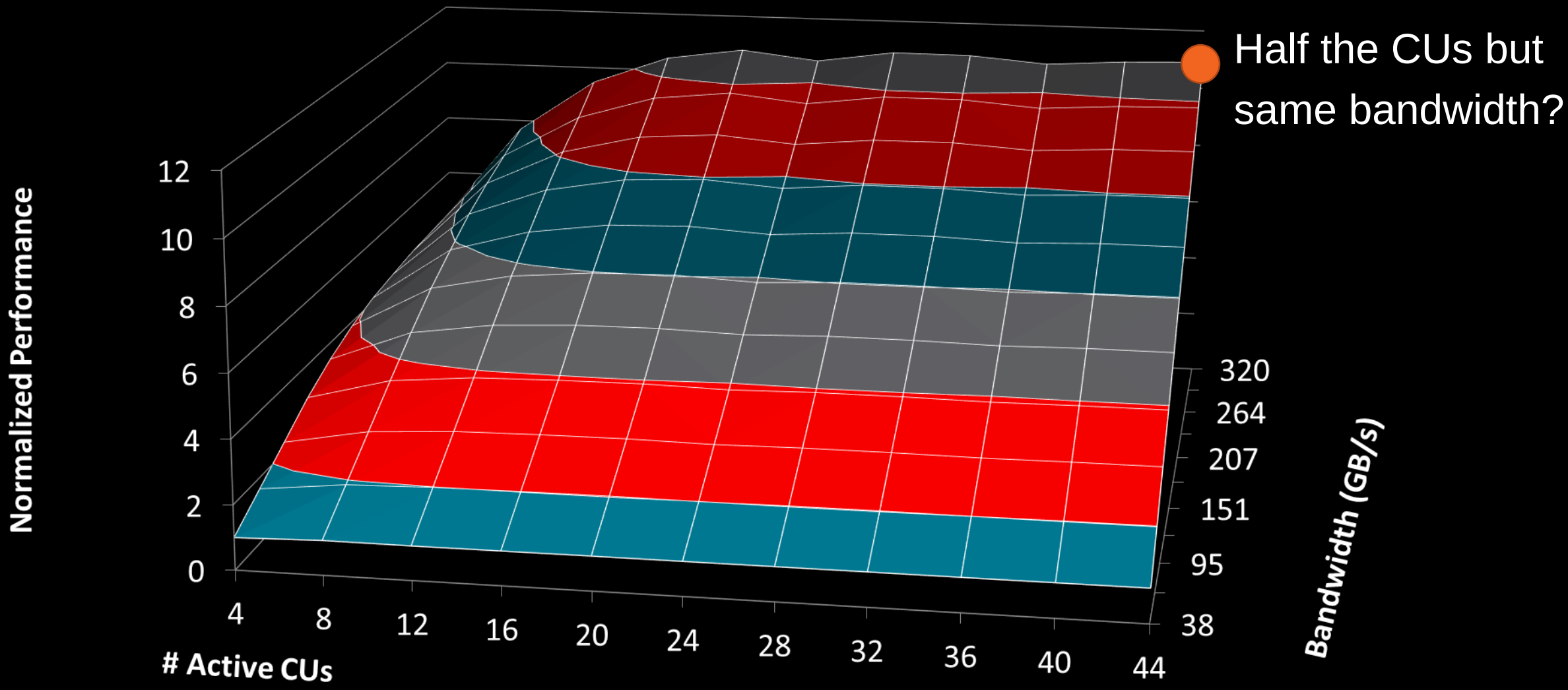
ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



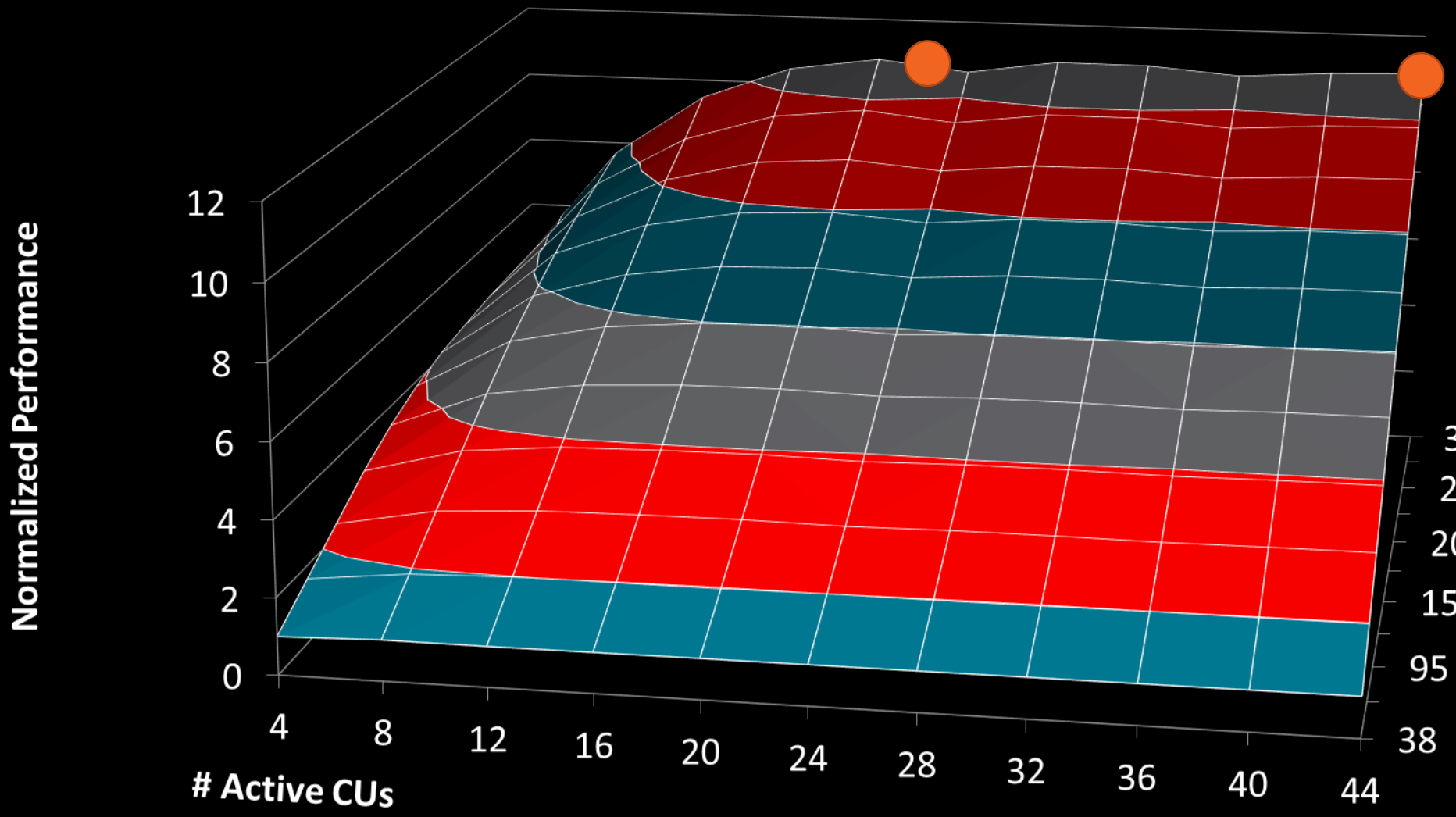
Half the CUs but same bandwidth?

~60% Performance

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE

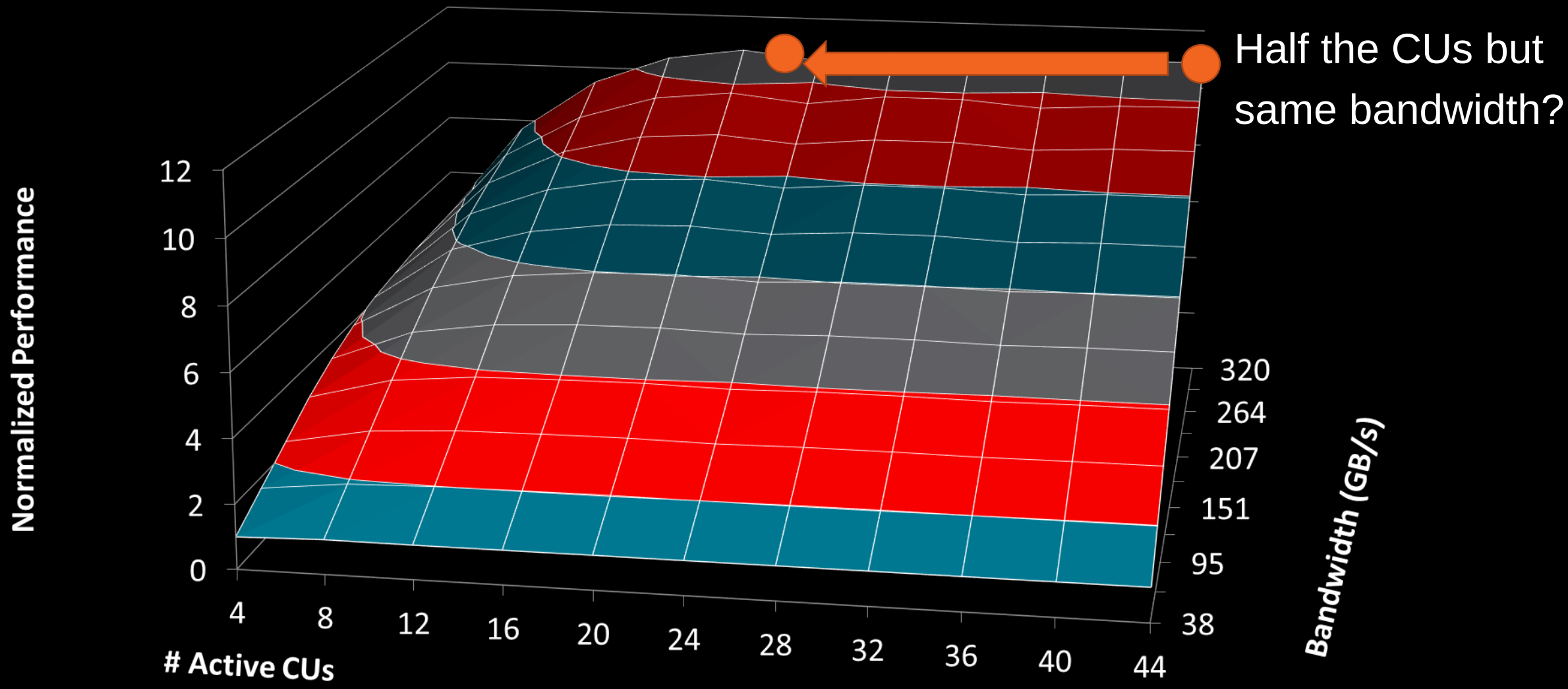


ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE

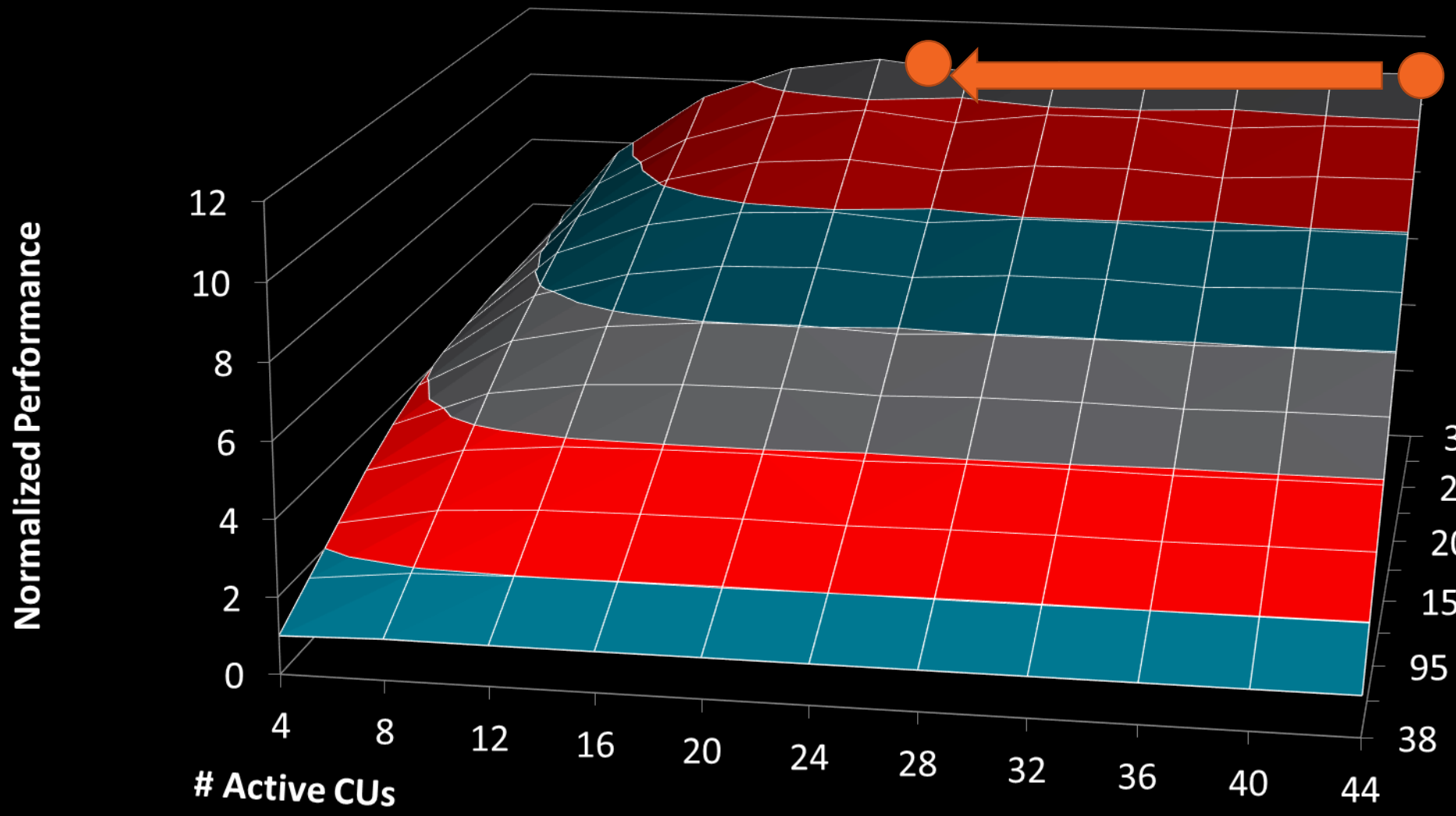


Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



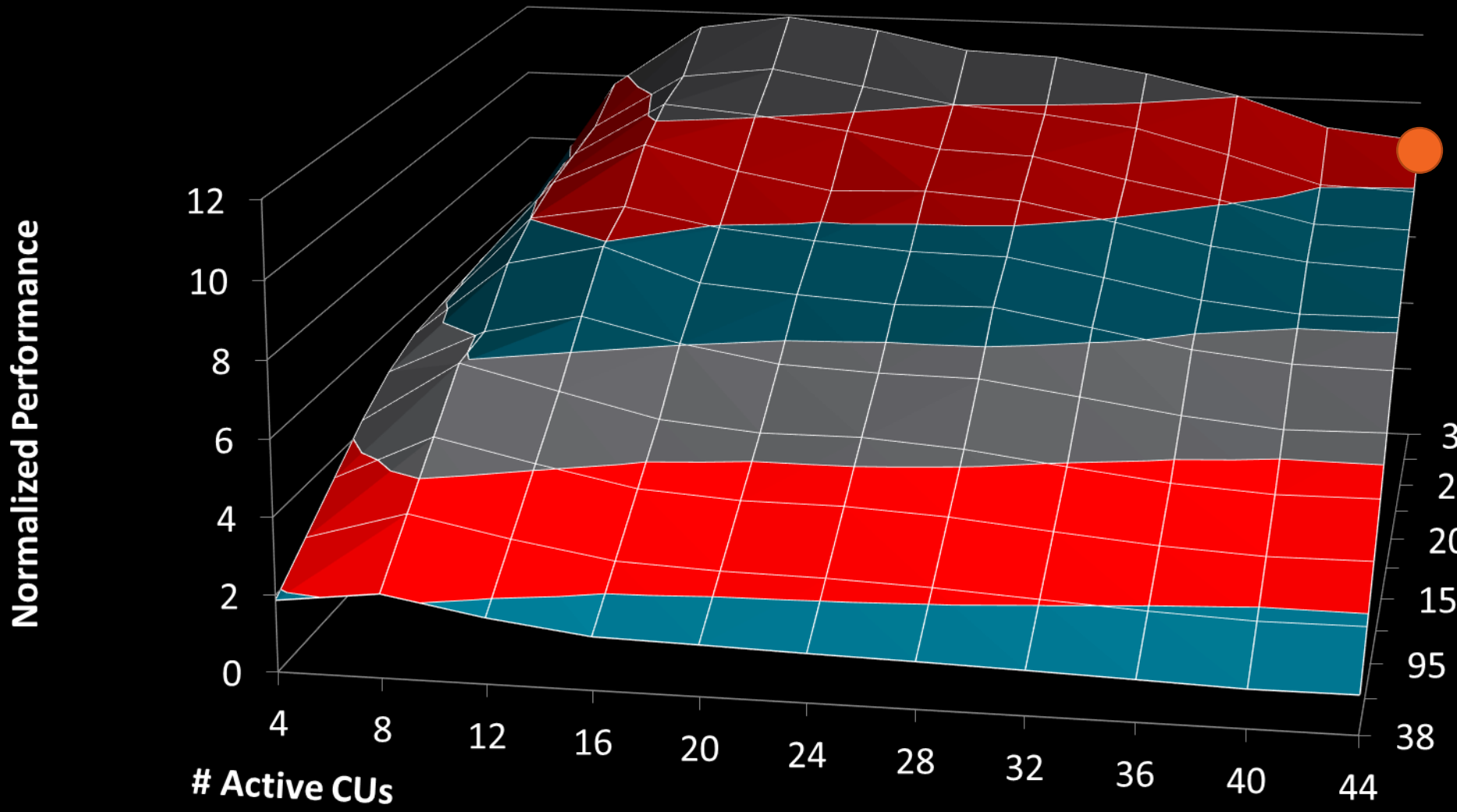
ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



Half the CUs but same bandwidth?

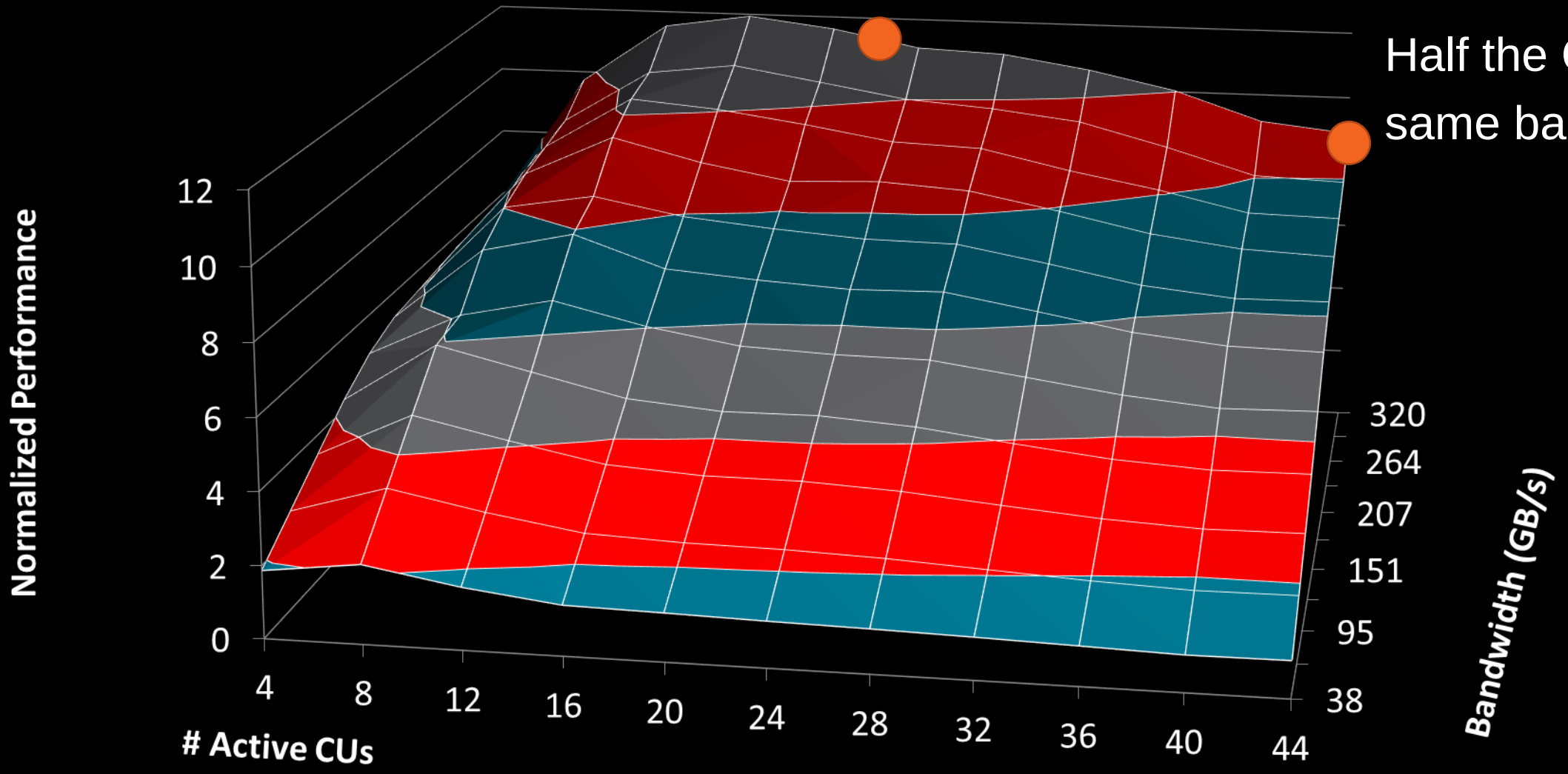
~100% Performance

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



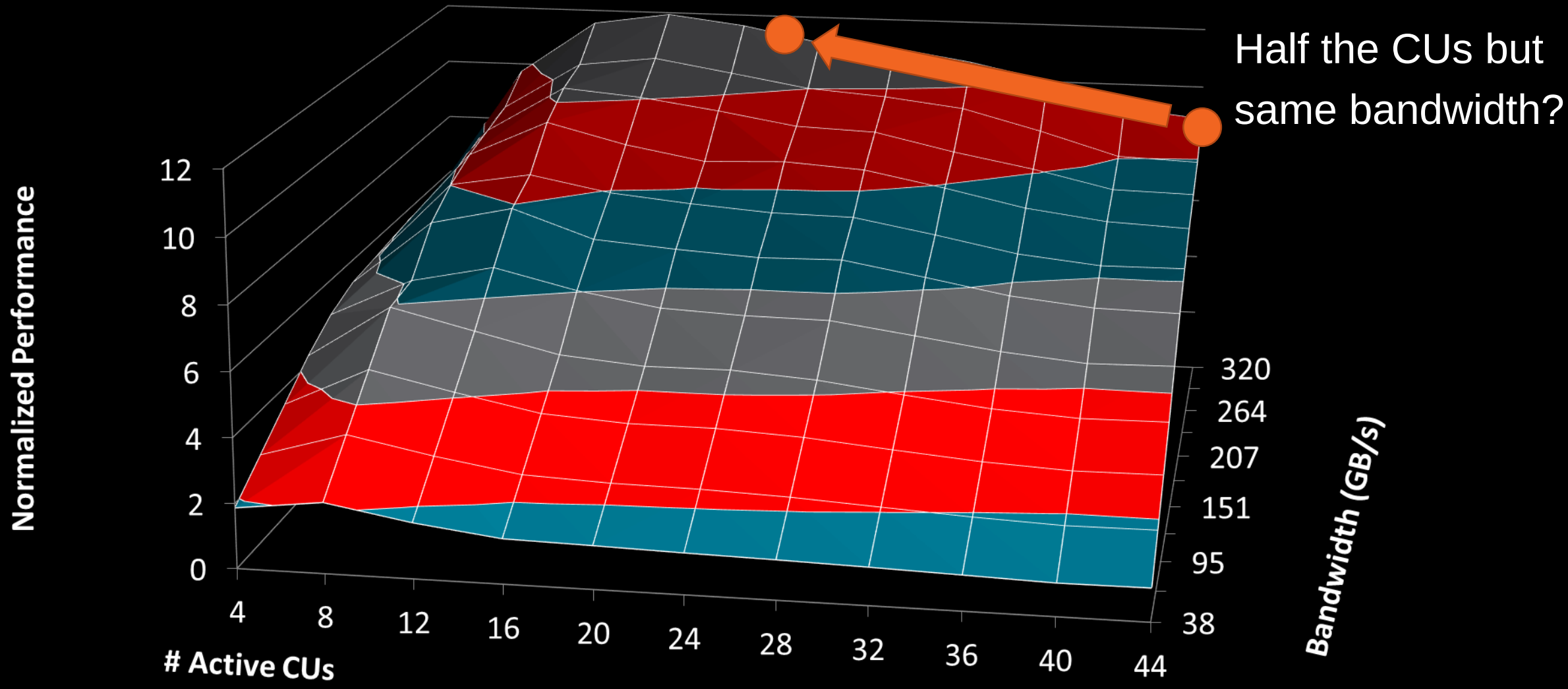
Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE

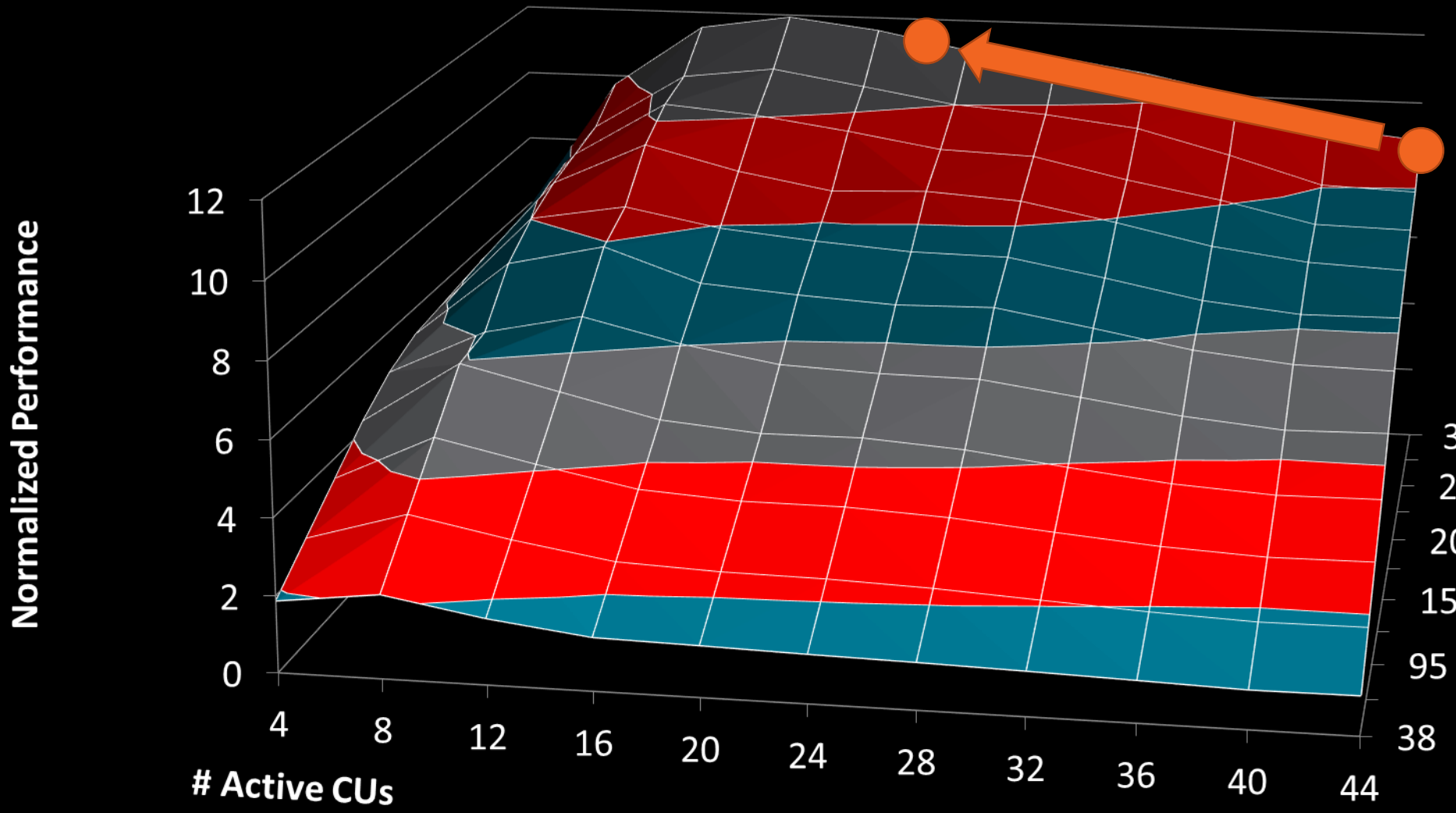


Half the CUs but same bandwidth?

ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



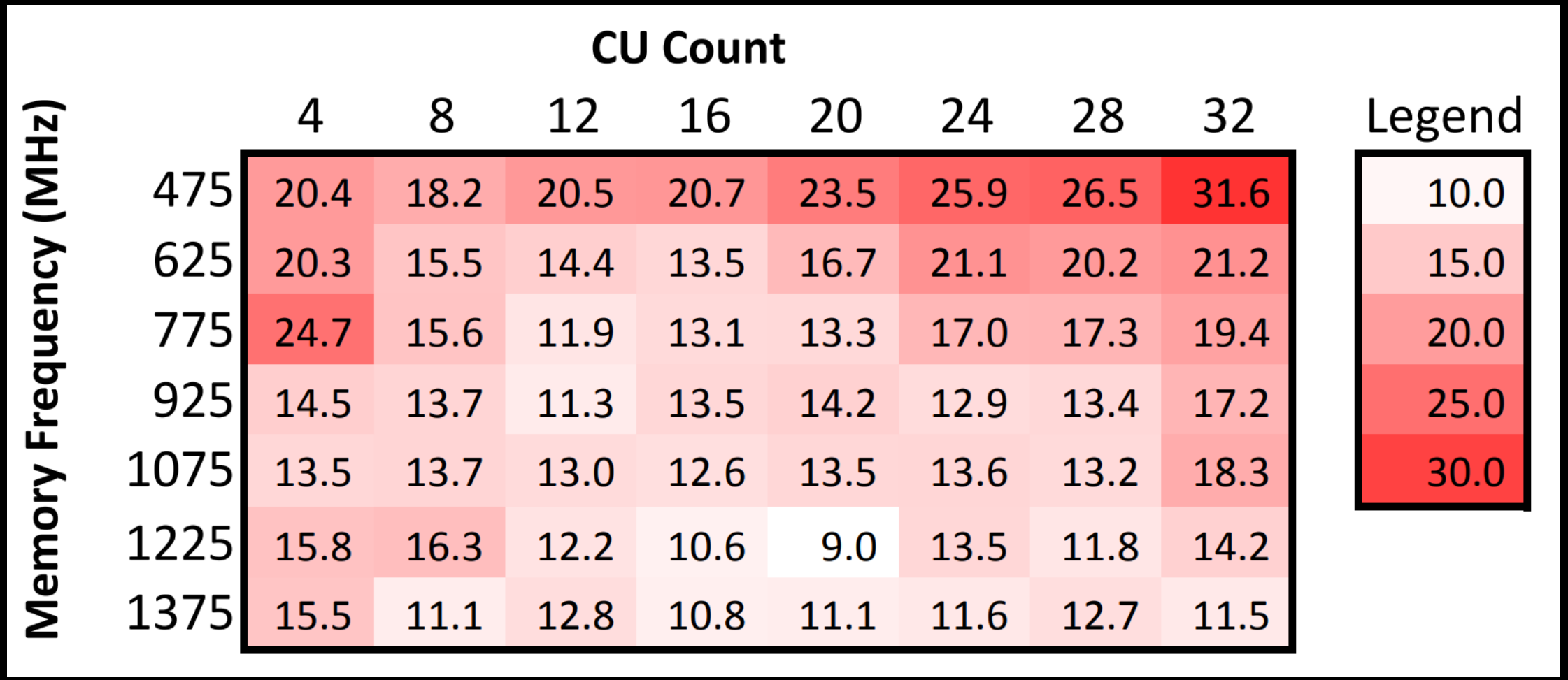
ESTIMATE KERNEL CHANGES USING CLUSTER'S CURVE



Half the CUs but same bandwidth?

~115% Performance

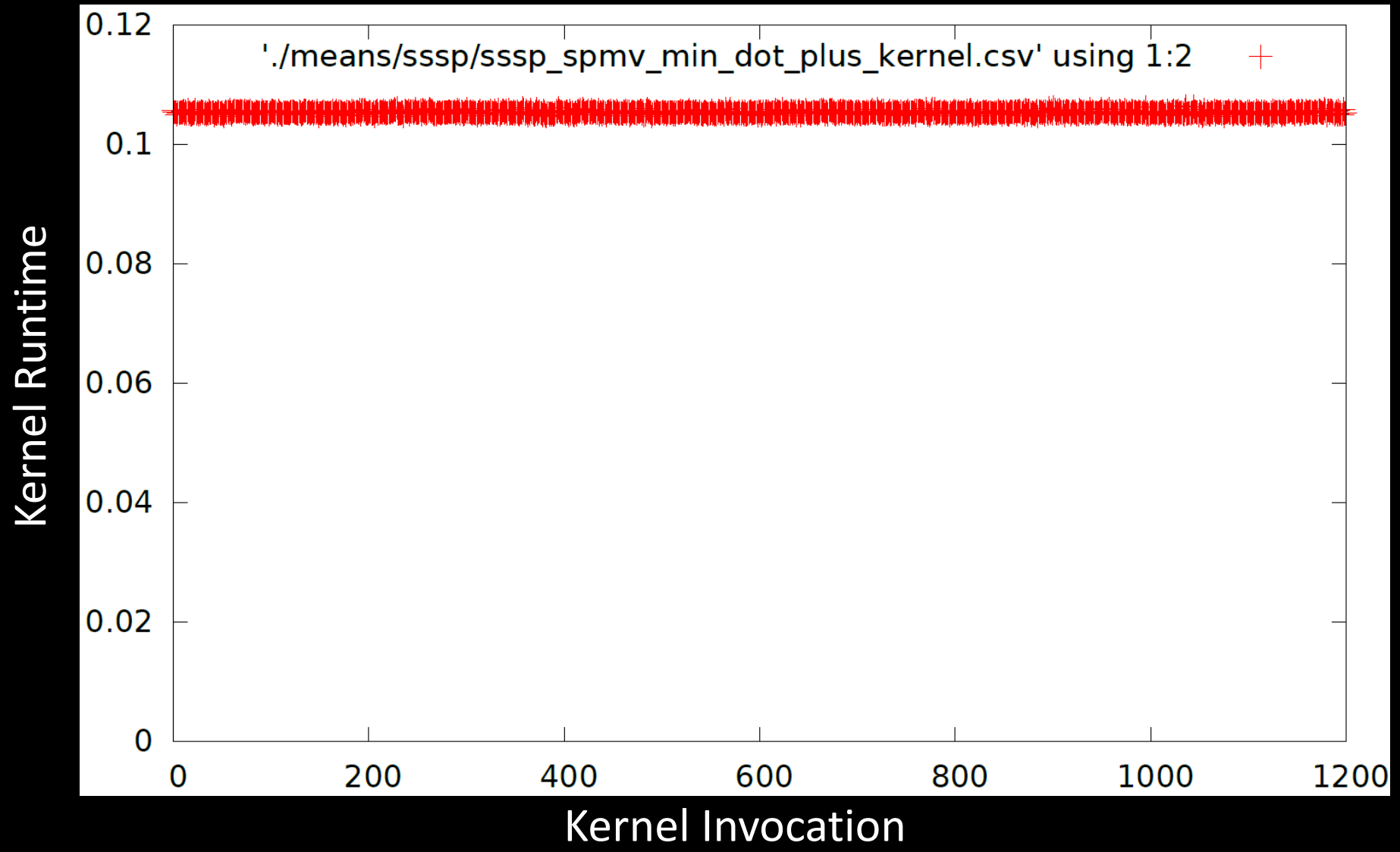
ACCURACY FROM ONE HW POINT TO ALL OTHERS



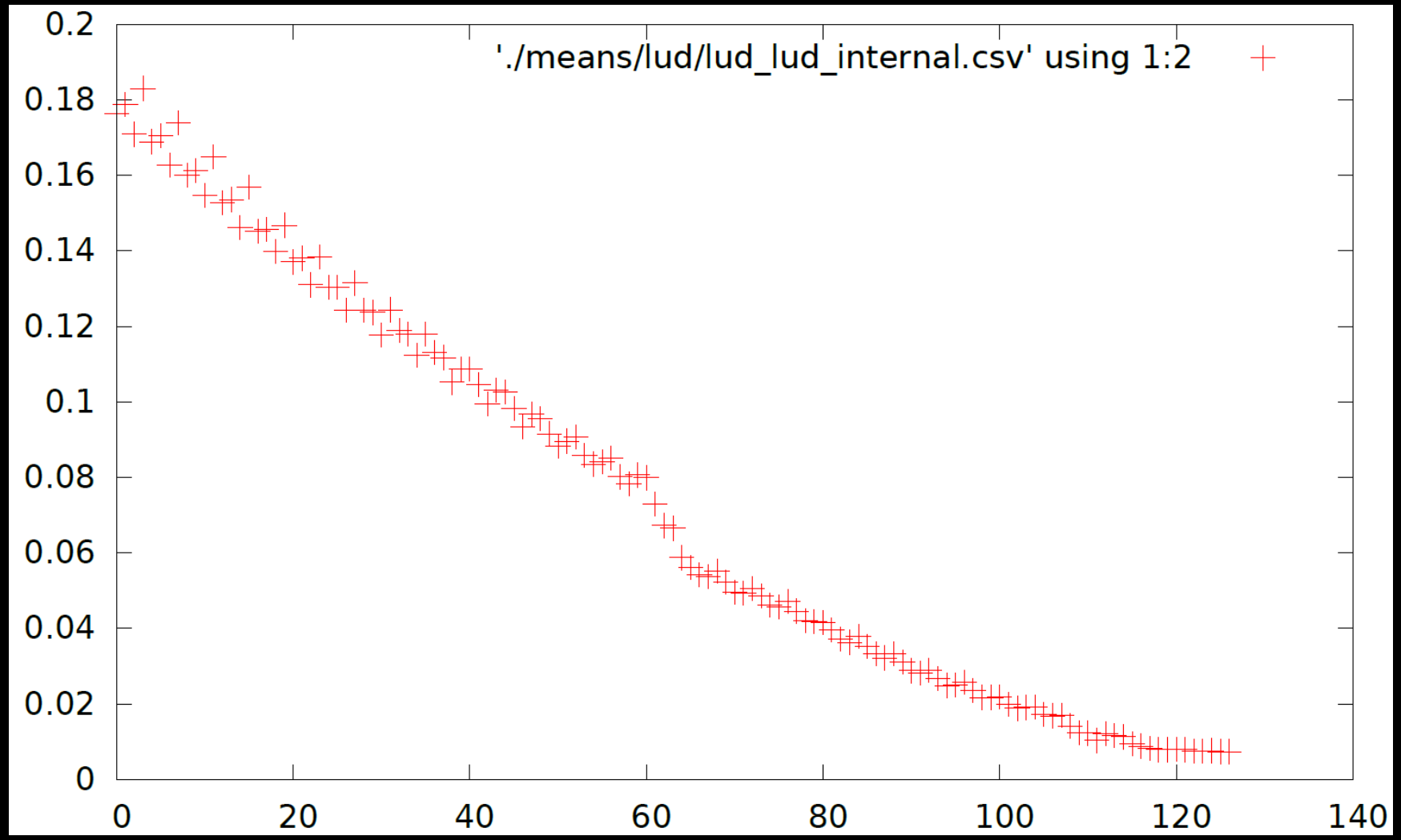
PREPARING GOOD DATA IS CHALLENGING

- This model needs a lot of data:
  - Multiple applications, many kernels
  - Numerous design points per application
- What kernels are representative?
- How to get clean performance and power data?

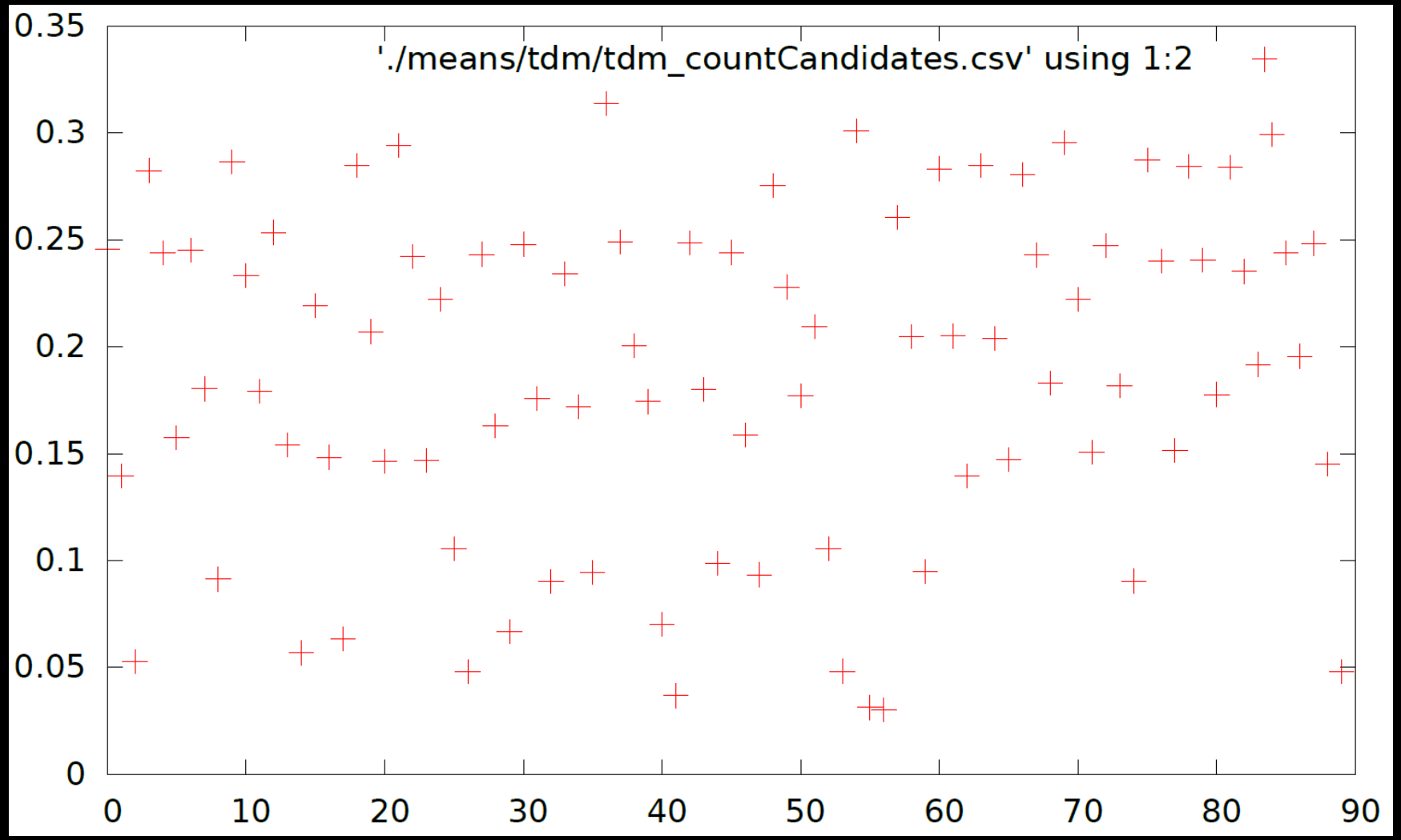
CHOOSING REPRESENTATIVE KERNELS



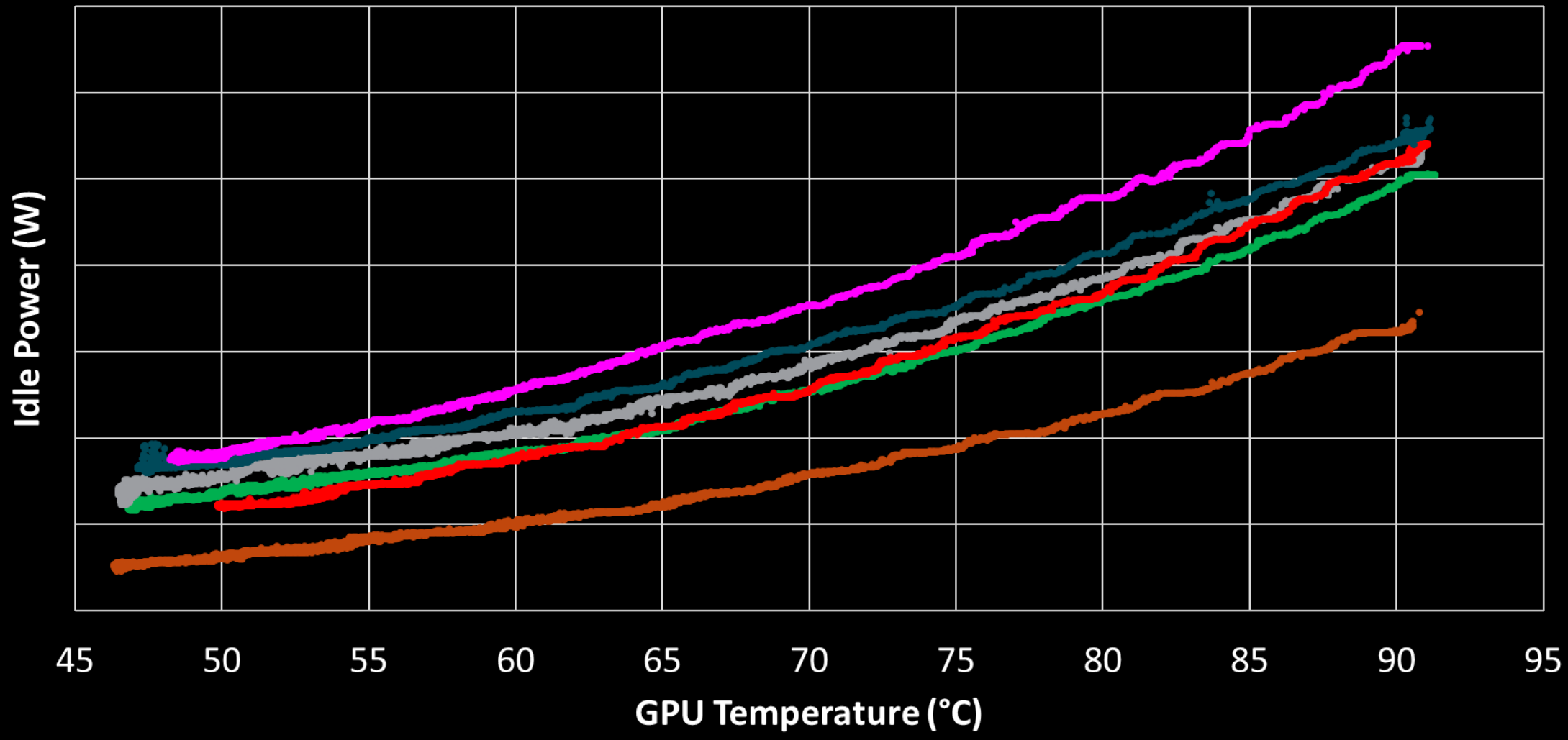
CHOOSING REPRESENTATIVE KERNELS



CHOOSING REPRESENTATIVE KERNELS



POWER MEASUREMENTS MUST ACCOUNT FOR STATIC POWER



SUMMARY

- Modern systems have a large heterogeneous hardware design space
- We need tools to do early design space exploration to help guide designs
- ML techniques help perform hardware-driven scaling studies
- Reasonable accuracy and very fast!
- But building clean, meaningful data sets presents a challenge

PAPERS ABOUT THIS TOPIC

- J. L. Greathouse, A. Lyashevsky, M. Meswani, N. Jayasena, M. Ignatowski, “Simulation of Exascale Nodes through Runtime Hardware Monitoring,” ModSim 2013
- B. Su, J. L. Greathouse, J. Gu, M. Boyer, L. Shen, Z. Wang, “Implementing a Leading Loads Performance Predictor on Commodity Processors,” USENIX ATC 2014
- D. P. Zhang, N, Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, M. Ignatowski, “TOP-PIM: Throughput-Oriented Programmable Processing in Memory,” HPDC 2014
- **G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, D. Chiou, “GPGPU Performance and Power Estimation Using Machine Learning,” HPCA 2015**
- A. Majumdar, G. Wu, K. Dev, J. L. Greathouse, I. Paul, W. Huang, A. K. Venugopal, L. Piga, C. Freitag, S. Puthoor, “A Taxonomy of GPGPU Performance Scaling,” IISWC 2015
- T. Vijayaraghavan, Y. Eckert, G. H. Loh, M. J. Schulte, M. Ignatowski, B M. Beckmann, W. C. Brantley, J. L. Greathouse, W. Huang, A. Karunanithi, O. Kayiran, M. Meswani, I. Paul, M. Poremba, S. Raasch, S. K. Reinhardt, G. Sadowski, V. Sridharan, “Design and Analysis of an APU for Exascale Computing,” HPCA 2017

QUESTIONS?

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

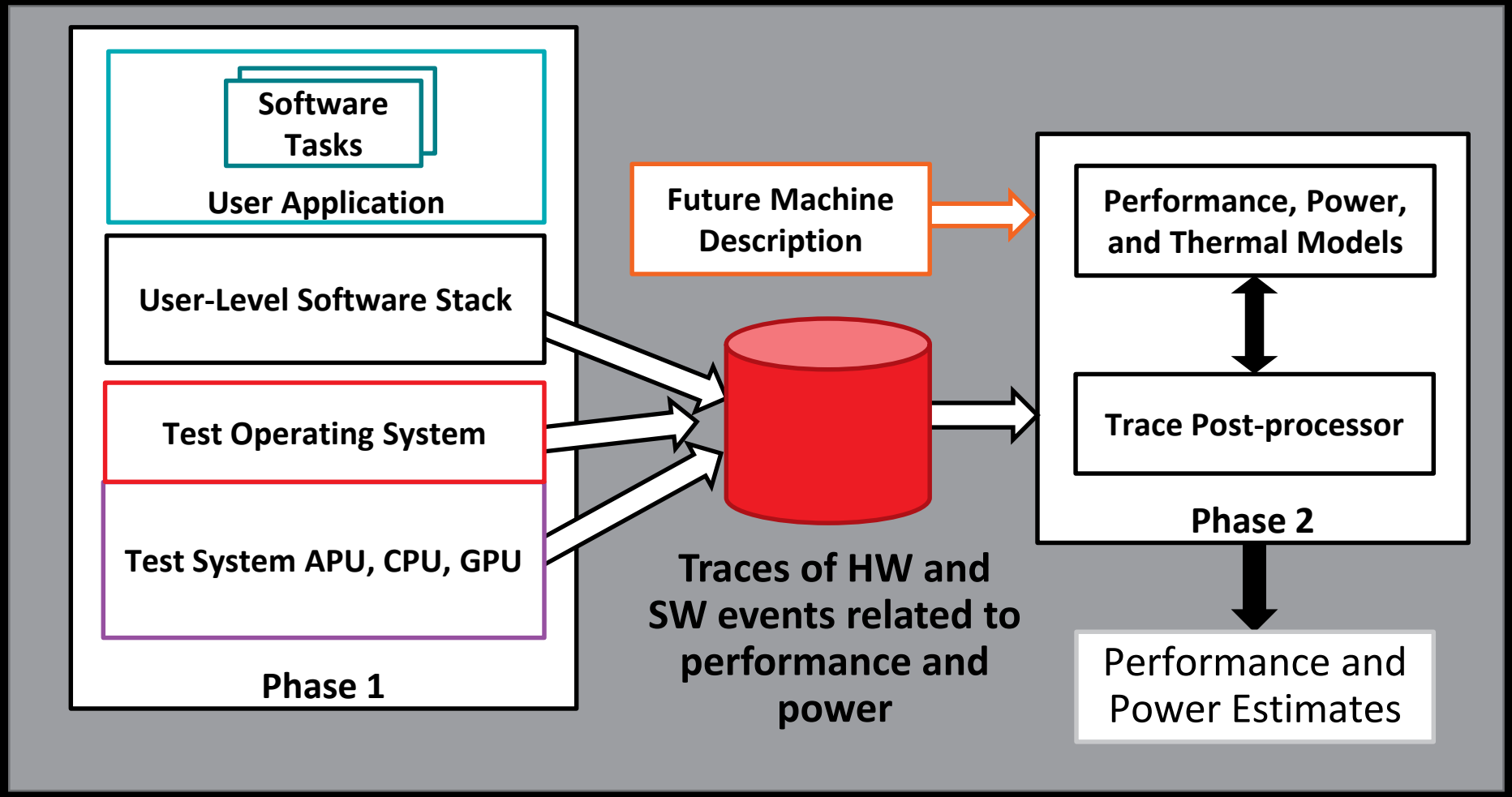
The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

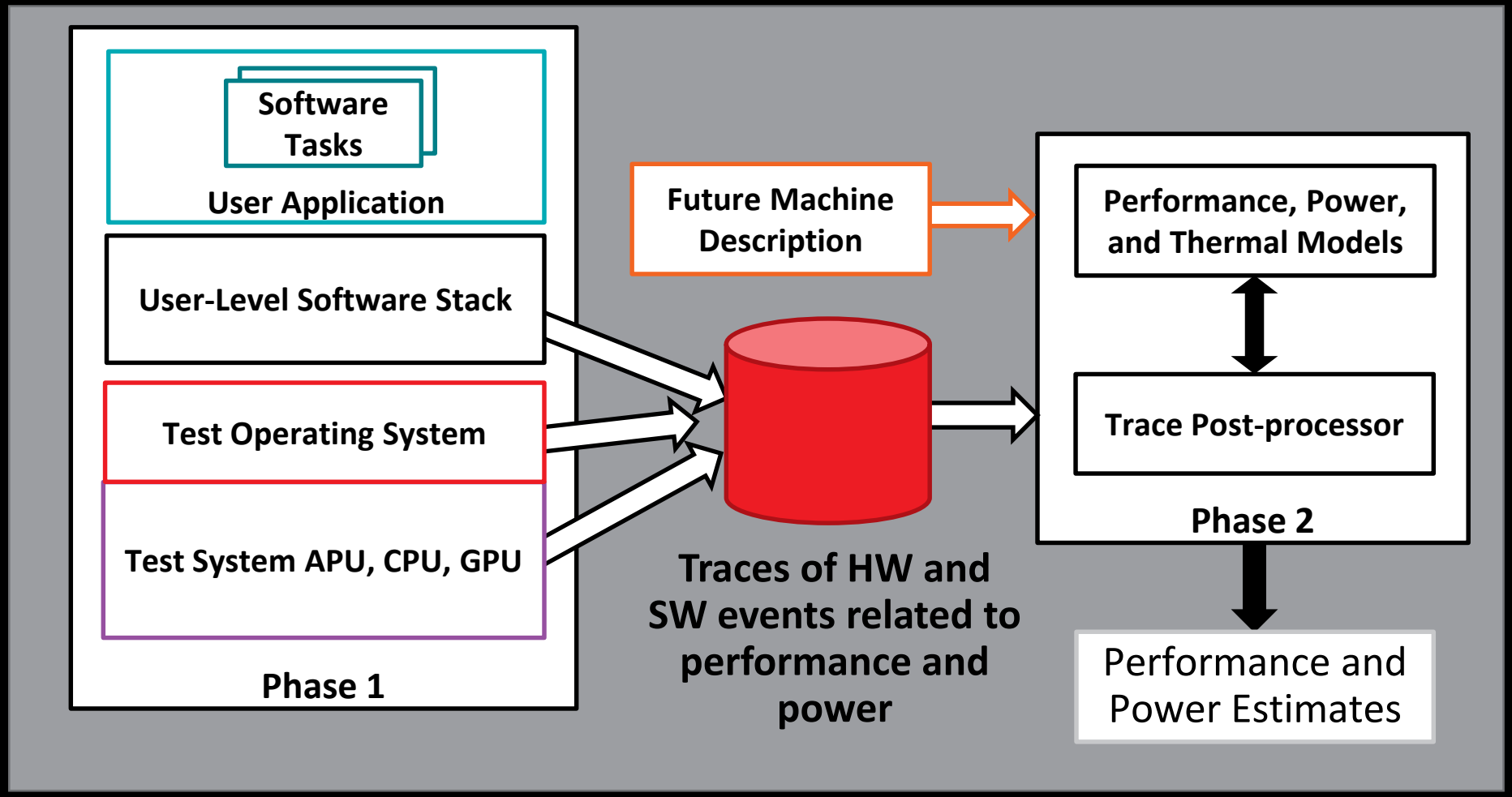
BACKUP SLIDES

USE HARDWARE MEASUREMENTS TO GUIDE EXPLORATION



USE HARDWARE MEASUREMENTS TO GUIDE EXPLORATION

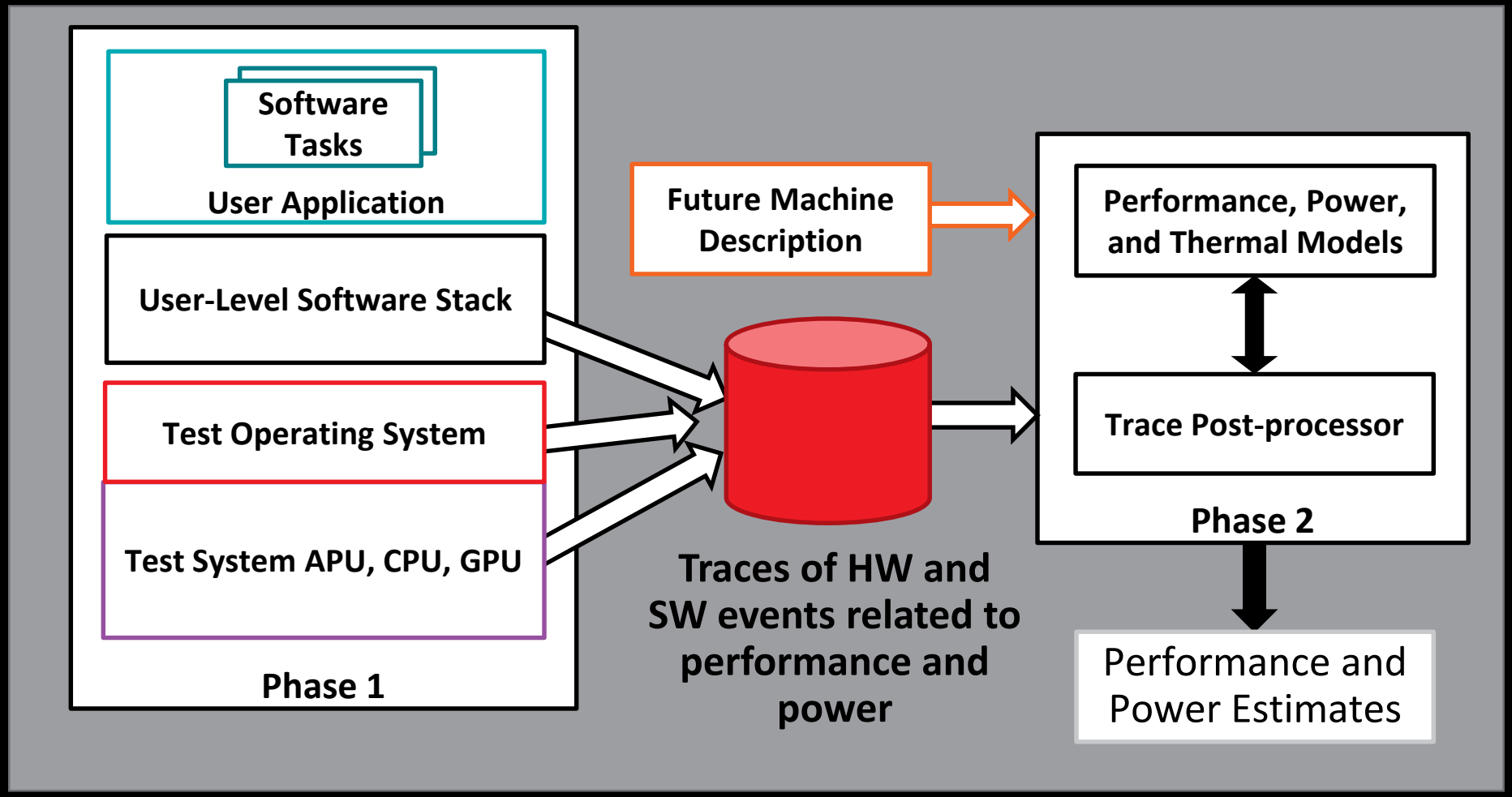
Step 1: Measure application on real hardware



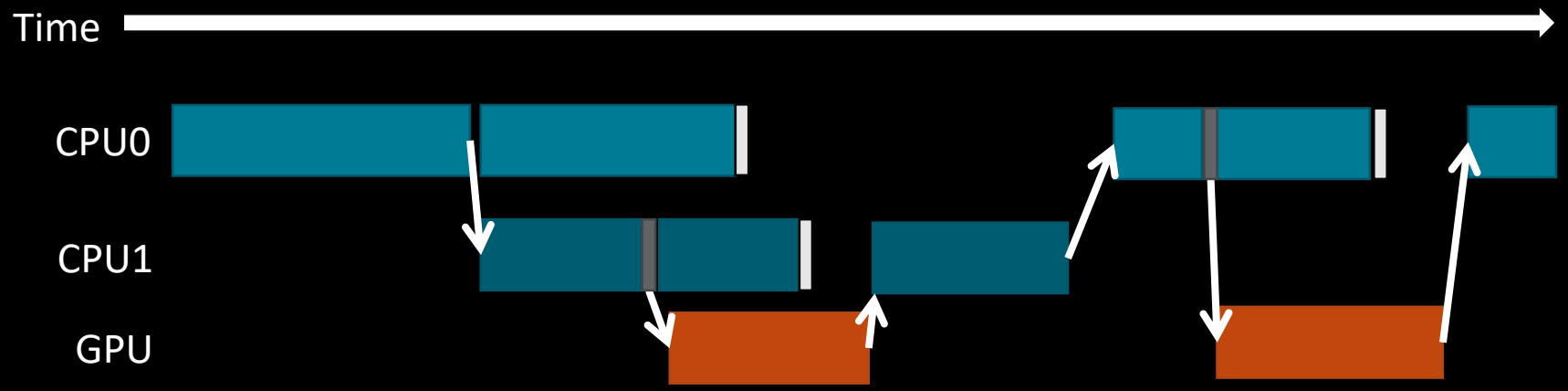
USE HARDWARE MEASUREMENTS TO GUIDE EXPLORATION

Step 1: Measure application on real hardware

Step 2: Estimate how application will work on future hardware

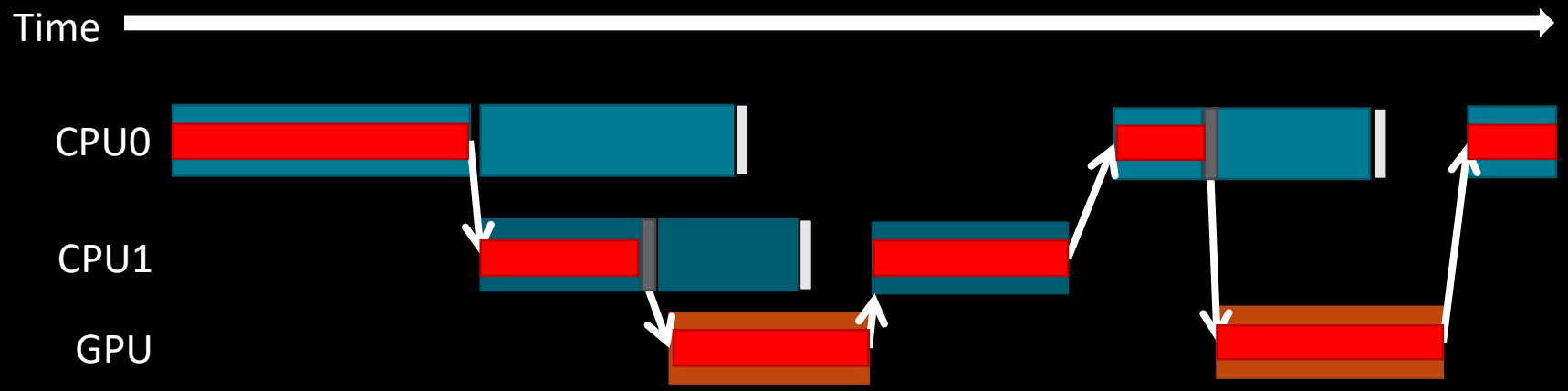


APPLICATION'S USE OF ALL HARDWARE MATTERS



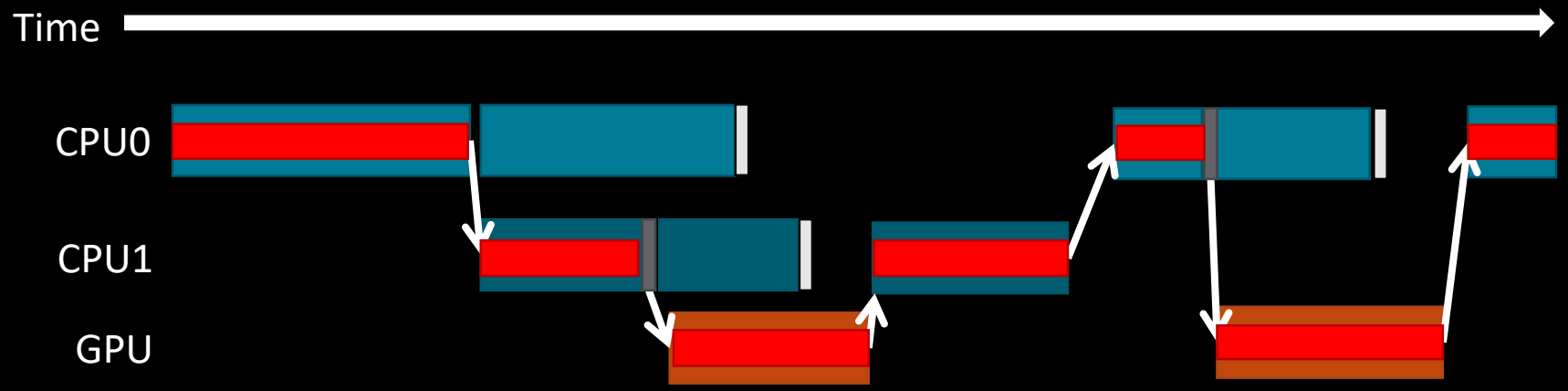
Example when everything except CPU1 gets 2x faster:

APPLICATION'S USE OF ALL HARDWARE MATTERS

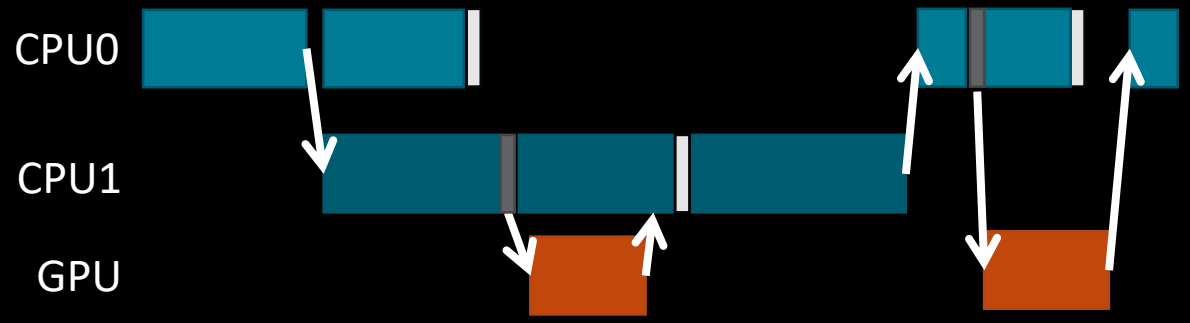


Example when everything except CPU1 gets 2x faster:

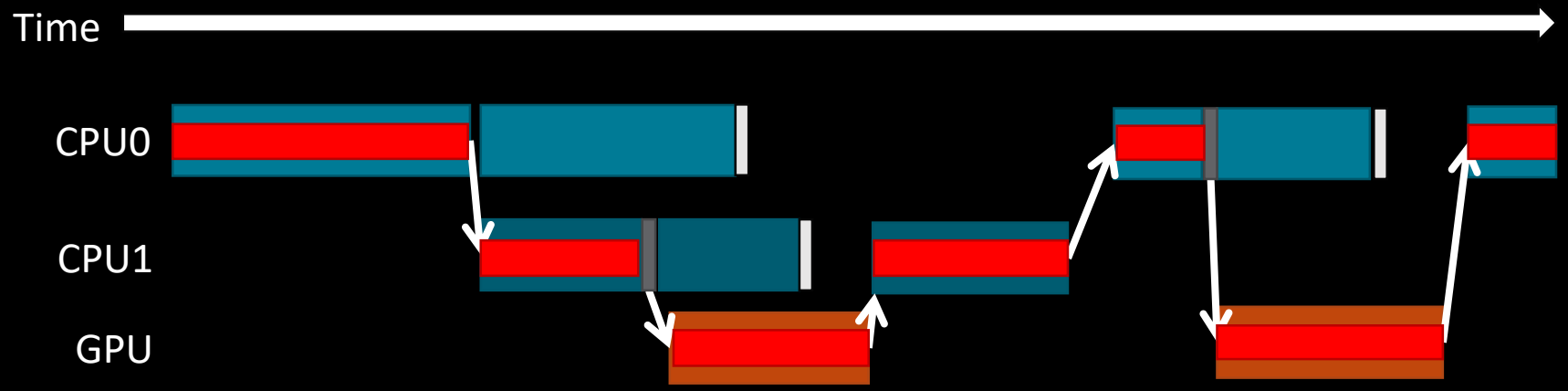
APPLICATION'S USE OF ALL HARDWARE MATTERS



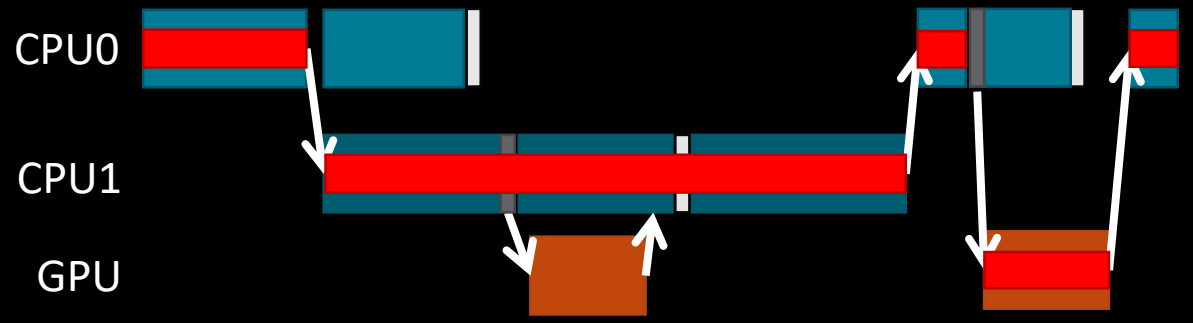
Example when everything except CPU1 gets 2x faster:



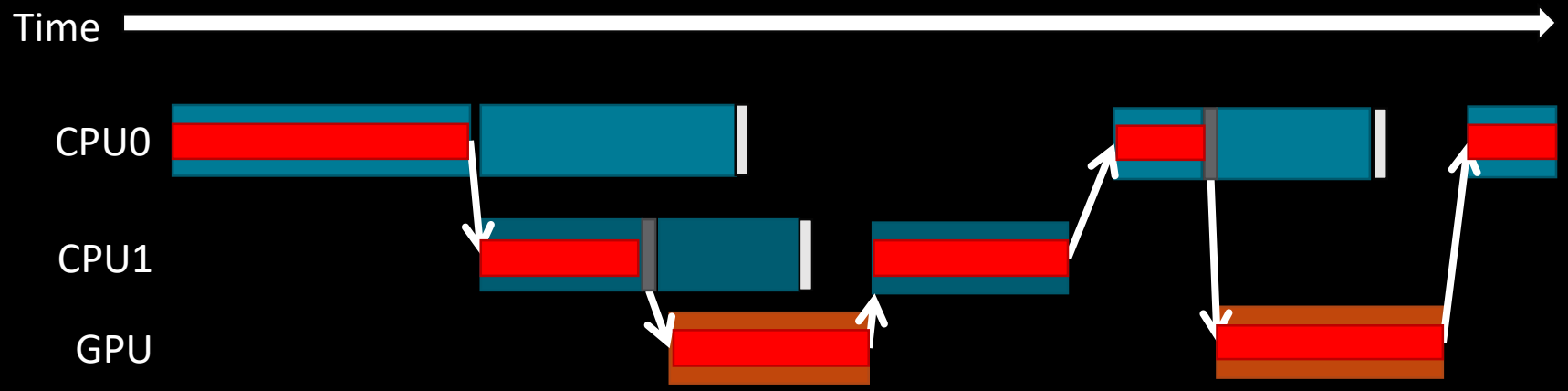
APPLICATION'S USE OF ALL HARDWARE MATTERS



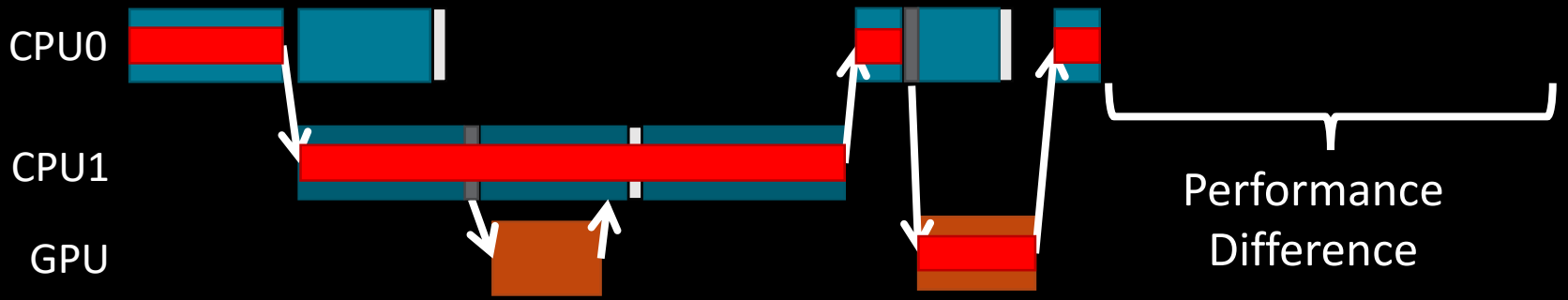
Example when everything except CPU1 gets 2x faster:



APPLICATION'S USE OF ALL HARDWARE MATTERS



Example when everything except CPU1 gets 2x faster:

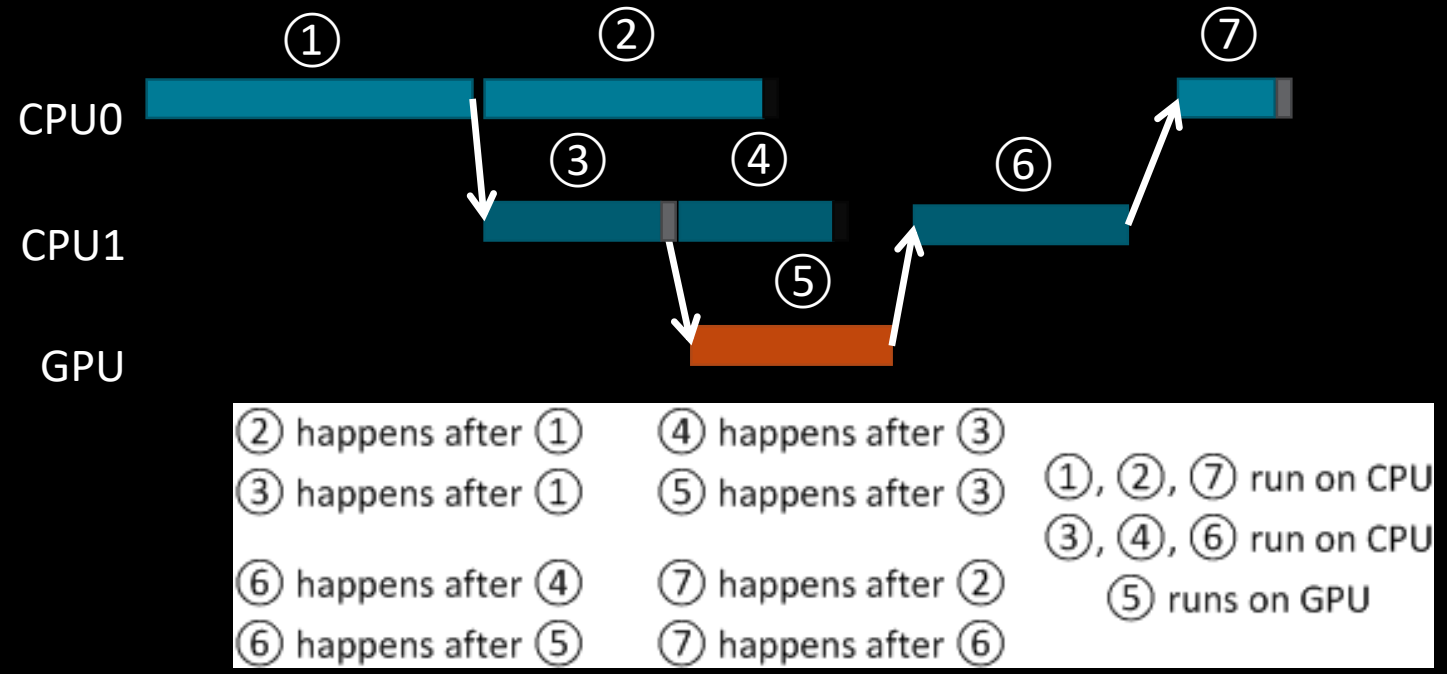


Less than 2x gain because CPU1 now on new critical path

RECONSTRUCTING APPLICATION CRITICAL PATHS

In phase 1, gather SW-level relationship between each segments

Use these relationships to build a legal execution order on simulated system



Gather ordering from library calls like pthread_create(), clWaitForEvents(), etc.

Could also split segments on user API calls, program phases