

Seminar in Computer Architecture Meeting 1: Introduction

Prof. Onur Mutlu

ETH Zürich

Fall 2019

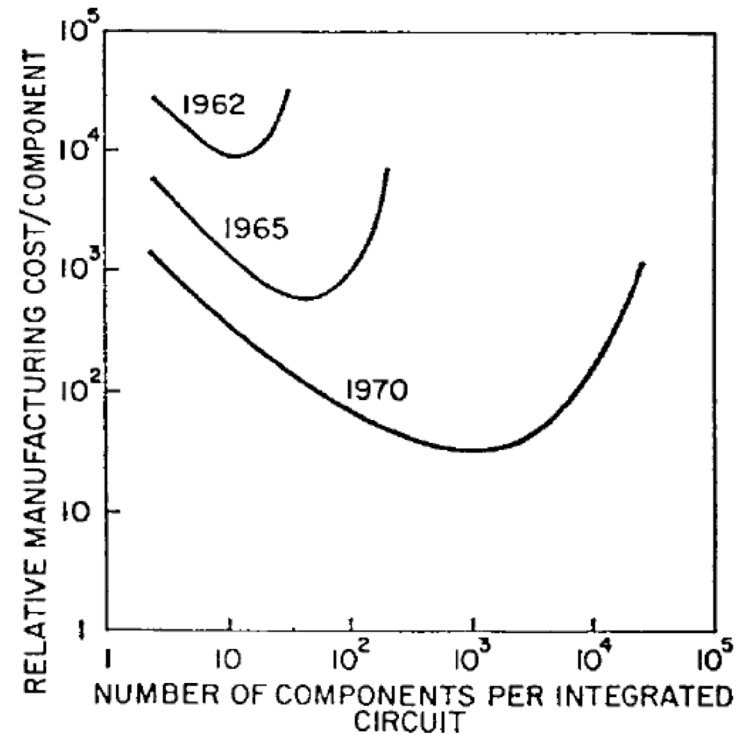
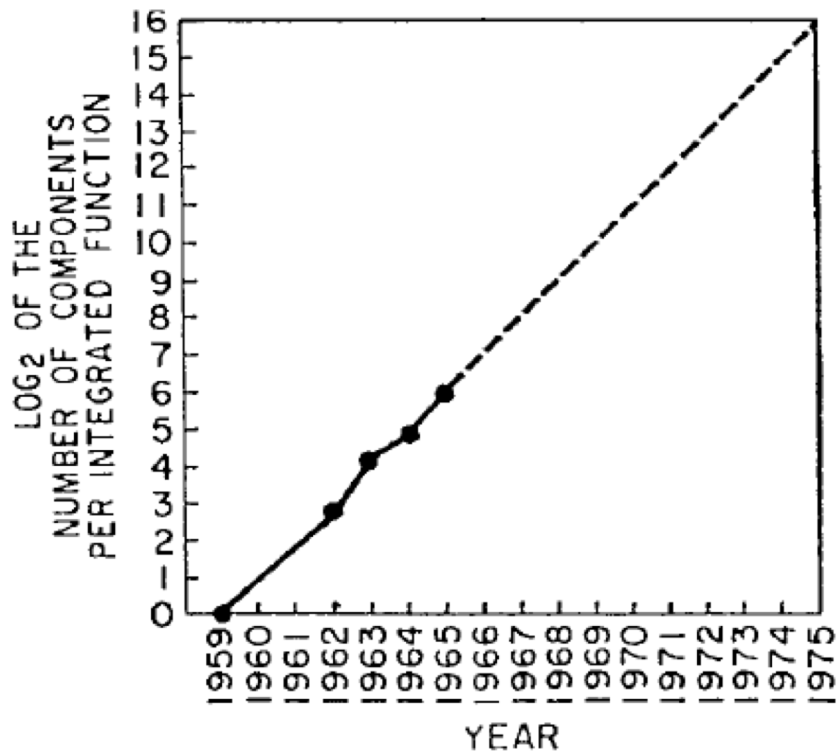
19 September 2019

Why Study Computer Architecture?

What is Computer Architecture?

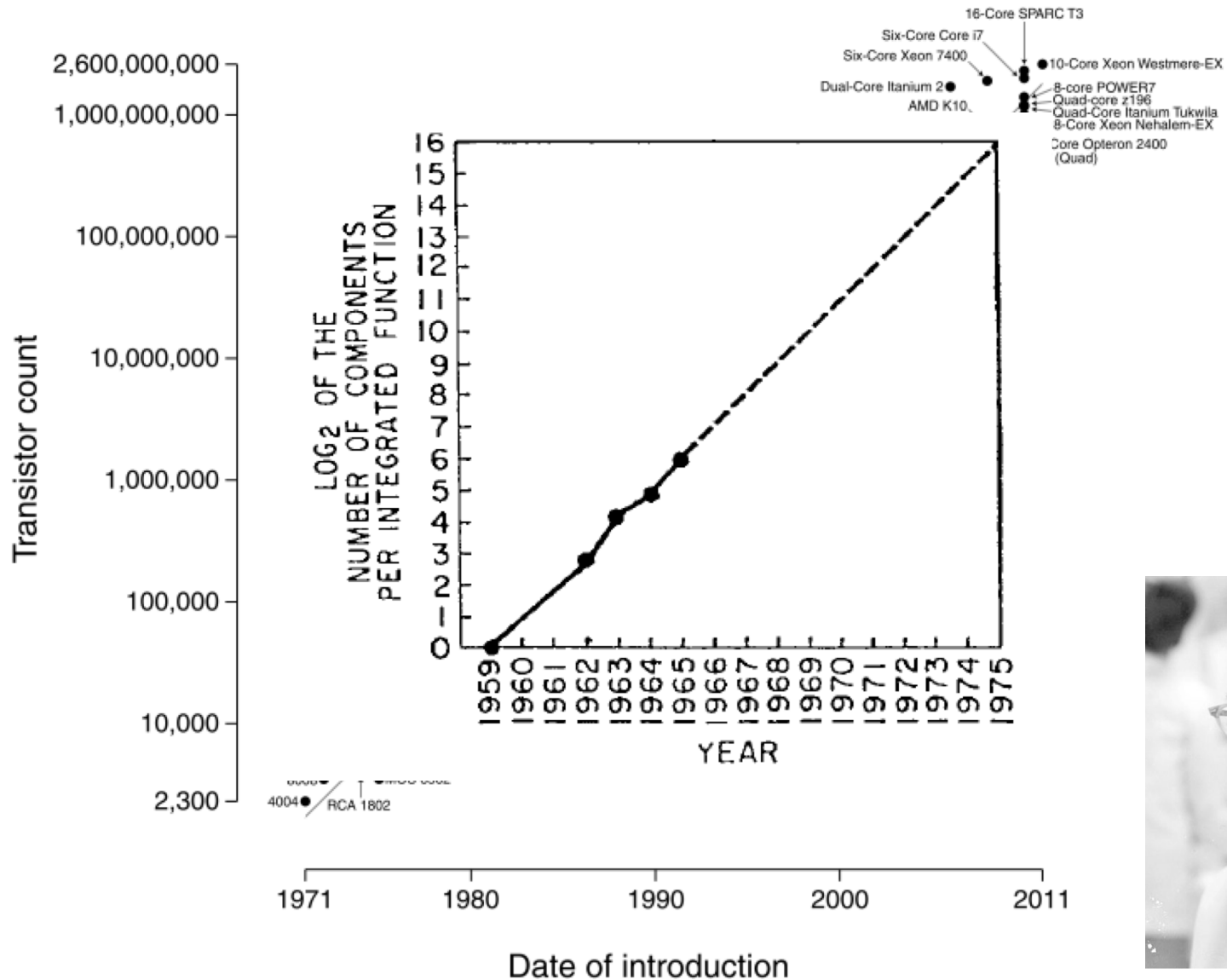
- The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

An Enabler: Moore's Law



Moore, “Cramming more components onto integrated circuits,”
Electronics Magazine, 1965. Component counts double every other year

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Number of transistors on an integrated circuit doubles ~ every two years

Recommended Reading

- Moore, “Cramming more components onto integrated circuits,” Electronics Magazine, 1965.
- Only 3 pages
- A quote:
"With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65 000 components on a single silicon chip."
- Another quote:
"Will it be possible to remove the heat generated by tens of thousands of components in a single silicon chip?"

Why Study Computer Architecture?

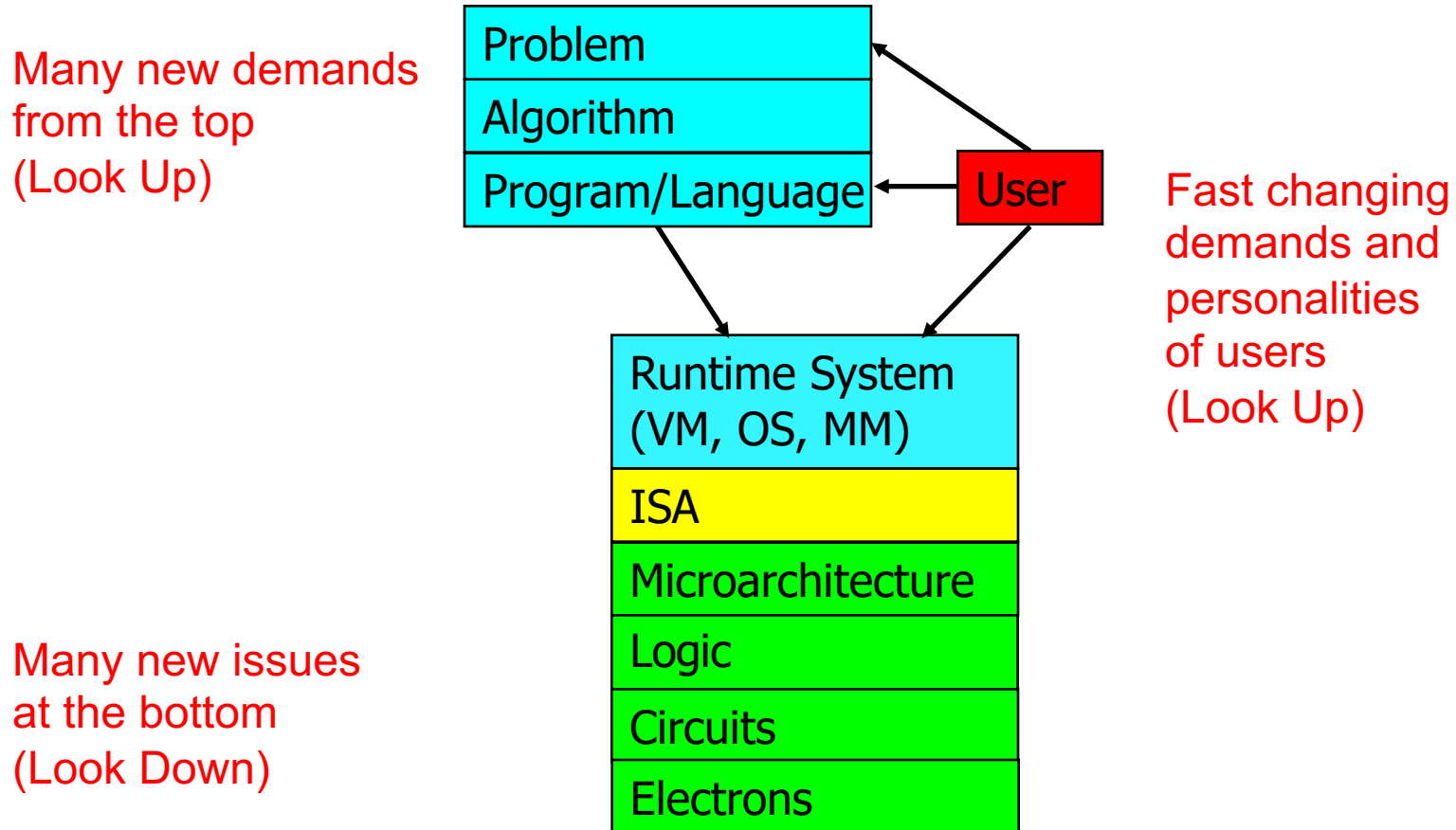
- **Enable better systems:** make computers faster, cheaper, smaller, more reliable, ...
 - By exploiting advances and changes in underlying technology/circuits
- **Enable new applications**
 - Life-like 3D visualization 20 years ago? Virtual reality?
 - Self-driving cars?
 - Personalized genomics? Personalized medicine?
- **Enable better solutions** to problems
 - Software innovation is built on trends and changes in computer architecture
 - > 50% performance improvement per year has enabled this innovation
- **Understand why computers work the way they do**

Computer Architecture Today (I)

- Today is a very exciting time to study computer architecture
 - Industry is in a large paradigm shift (to novel architectures)
 - many different potential system designs possible
 - **Many difficult problems** *motivating* and *caused by* the shift
 - Huge hunger for data and new data-intensive applications
 - Power/energy/thermal constraints
 - Complexity of design
 - Difficulties in technology scaling
 - Memory wall/gap
 - Reliability problems
 - Programmability problems
 - Security and privacy issues
 - No clear, definitive answers to these problems
-

Computer Architecture Today (II)

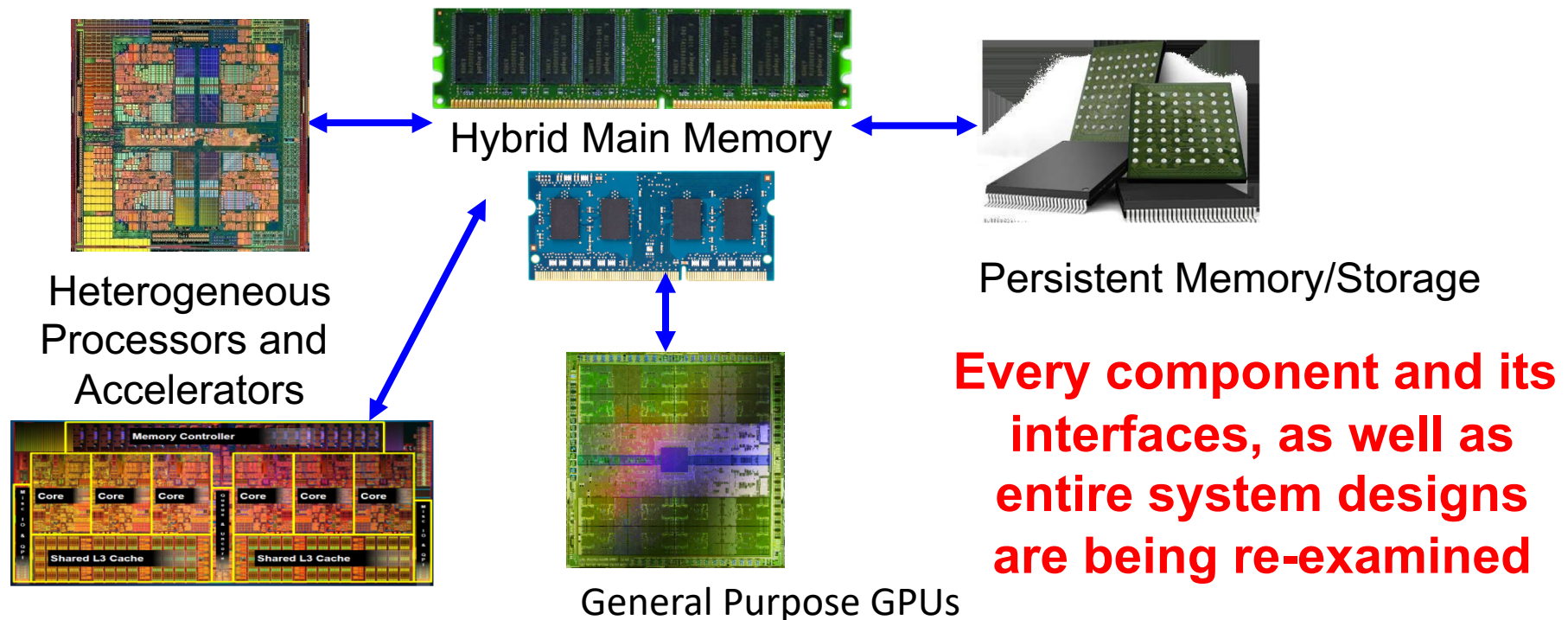
- These problems affect all parts of the computing stack – if we do not change the way we design systems



- No clear, definitive answers to these problems
-

Computer Architecture Today (III)

- Computing landscape is very different from 10-20 years ago
- Both UP (software and humanity trends) and DOWN (technologies and their issues), FORWARD and BACKWARD, and the resulting requirements and constraints



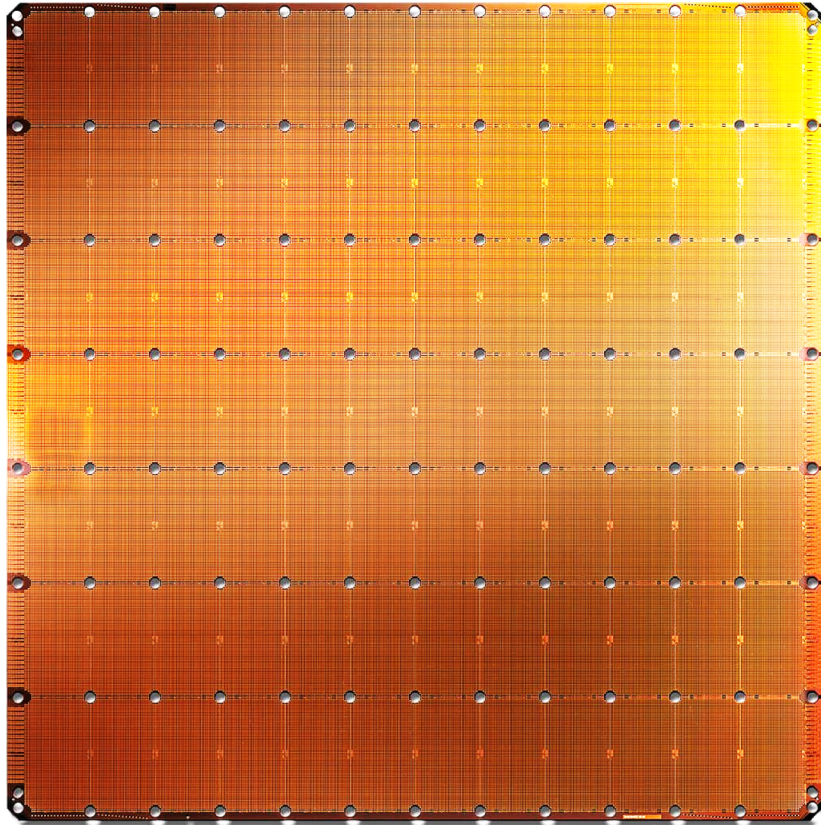
Many Interesting Things
Are Happening Today
in Computer Architecture

Intel Optane Persistent Memory (2019)

- Non-volatile main memory
- Based on 3D-XPoint Technology



Cerebras's Wafer Scale Engine (2019)



Cerebras WSE

1.2 Trillion transistors
46,225 mm²

- The largest ML accelerator chip
- 400,000 cores



Largest GPU

21.1 Billion transistors
815 mm²

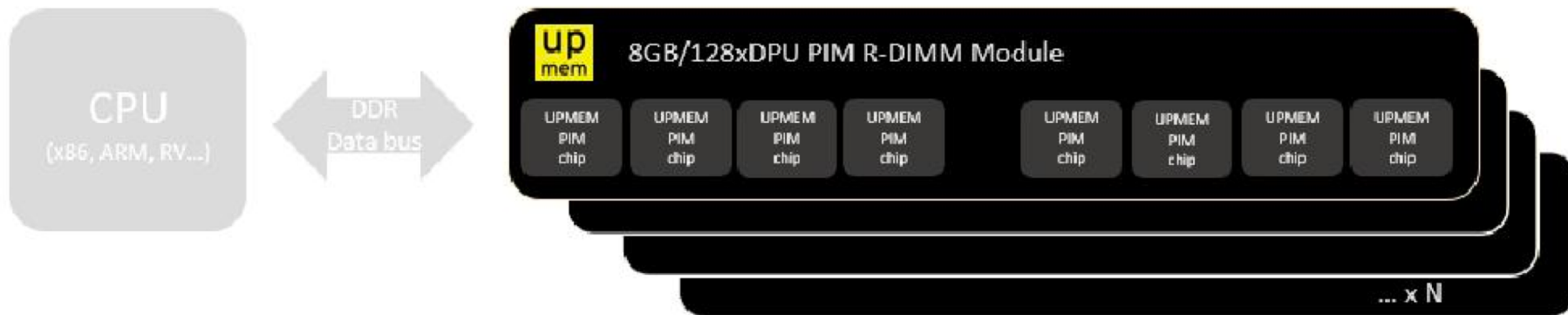
NVIDIA TITAN V

<https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning>

<https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning>

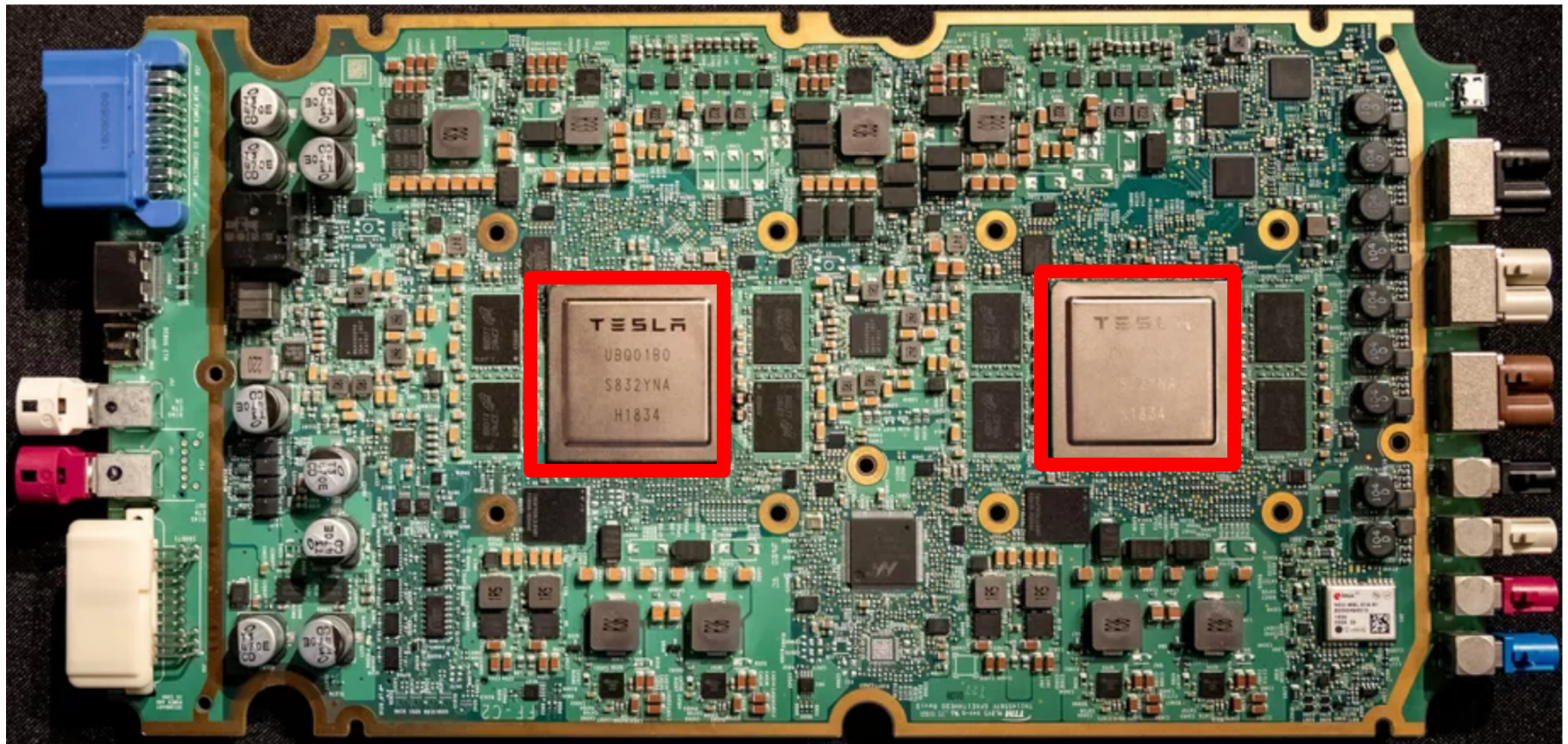
UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
 - DDR4 R-DIMM modules
 - 8GB+128 DPUs (16 PIM chips)
 - Standard 2x-nm DRAM process
 - **Large amounts of** compute & memory bandwidth



TESLA Full Self-Driving Computer (2019)

- ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
- Two redundant chips for better safety.



Google TPU Generation I (~2016)

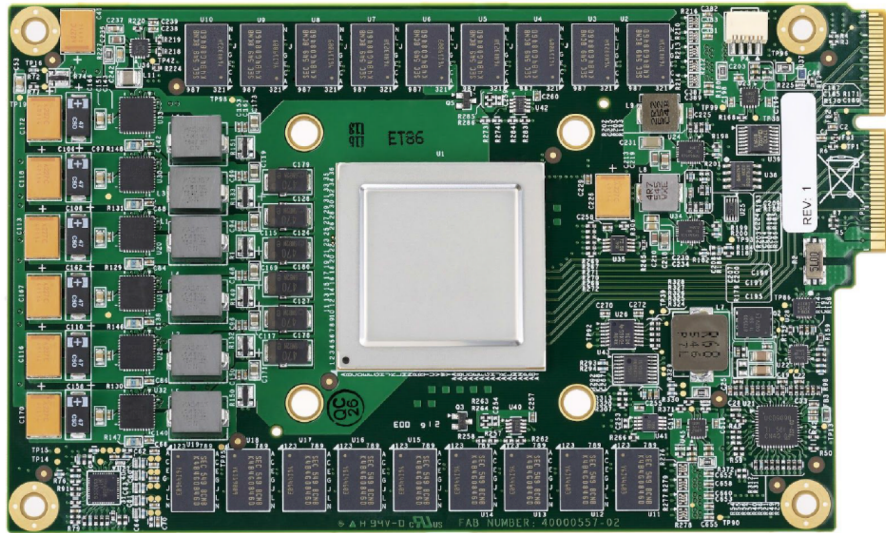


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

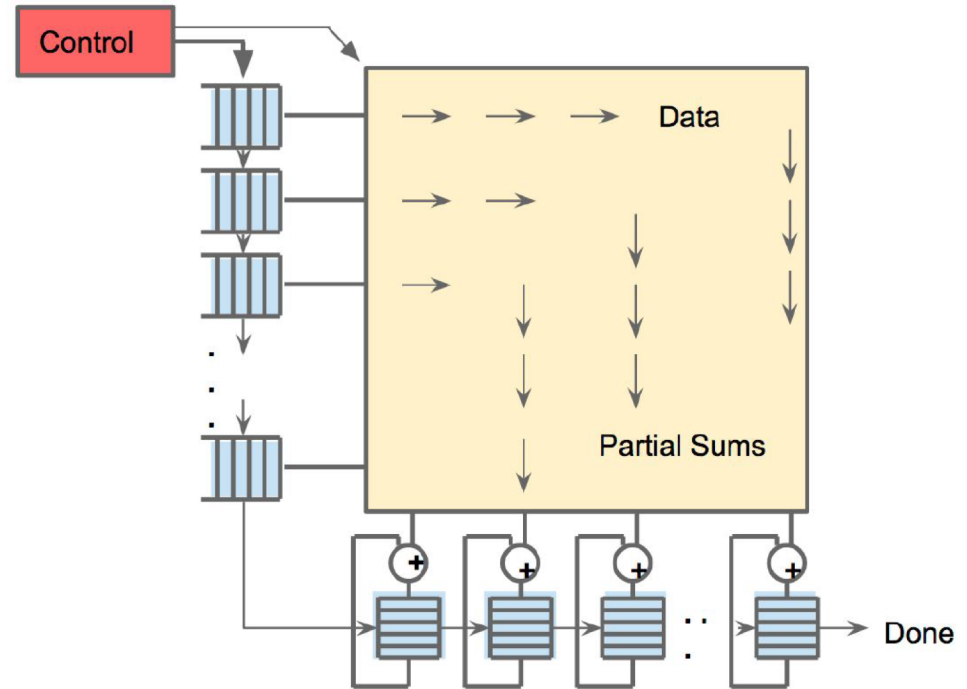
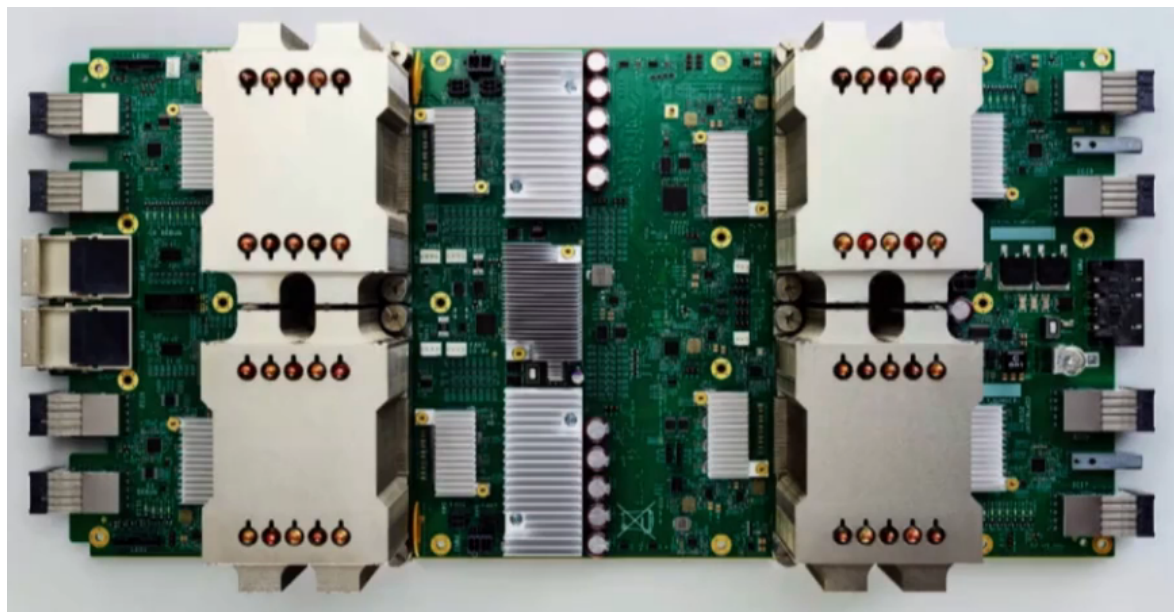


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Jouppi et al., “In-Datcenter Performance Analysis of a Tensor Processing Unit”, ISCA 2017.

Google TPU Generation II (2017)



<https://www.nextplatform.com/2017/05/17/first-depth-look-googles-new-second-generation-tpu/>

4 TPU chips
vs 1 chip in TPU1

High Bandwidth Memory
vs DDR3

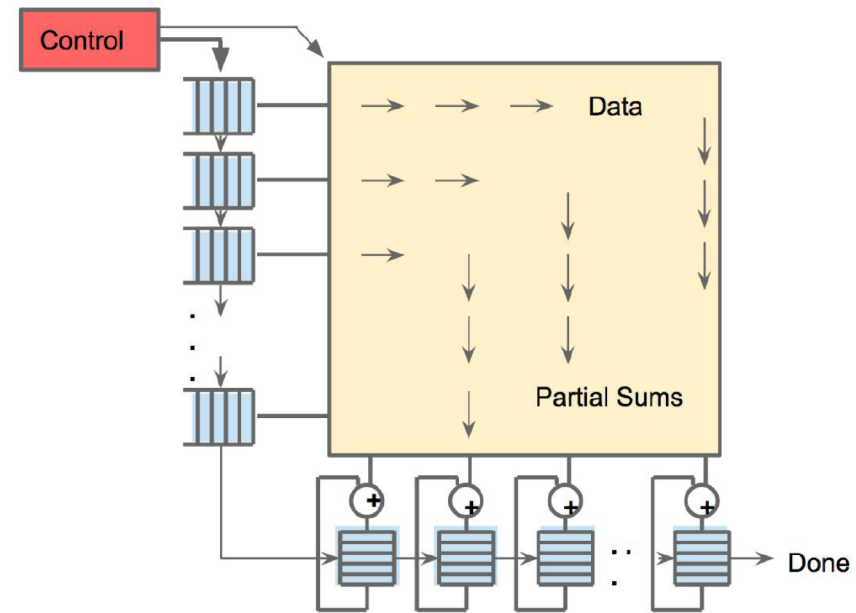
Floating point operations
vs FP16

45 TFLOPS per chip
vs 23 TOPS

Designed for **training**
and **inference**
vs only inference

An Example Modern Systolic Array: TPU (II)

As reading a large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing reads and writes of the Unified Buffer [Kun80][Ram91][Ovt15b]. Figure 4 shows that data flows in from the left, and the weights are loaded from the top. A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wavefront. The weights are preloaded, and take effect with the advancing wave alongside the first data of a new block. Control and data are pipelined to give the illusion that the 256 inputs are read at once, and that they instantly update one location of each of 256 accumulators. From a correctness perspective, software is unaware of the systolic nature of the matrix unit, but for performance, it does worry about the latency of the unit.



Jouppi et al., “In-Datacenter Performance Analysis of a Tensor Processing Unit”, ISCA 2017.

An Example Modern Systolic Array: TPU (III)

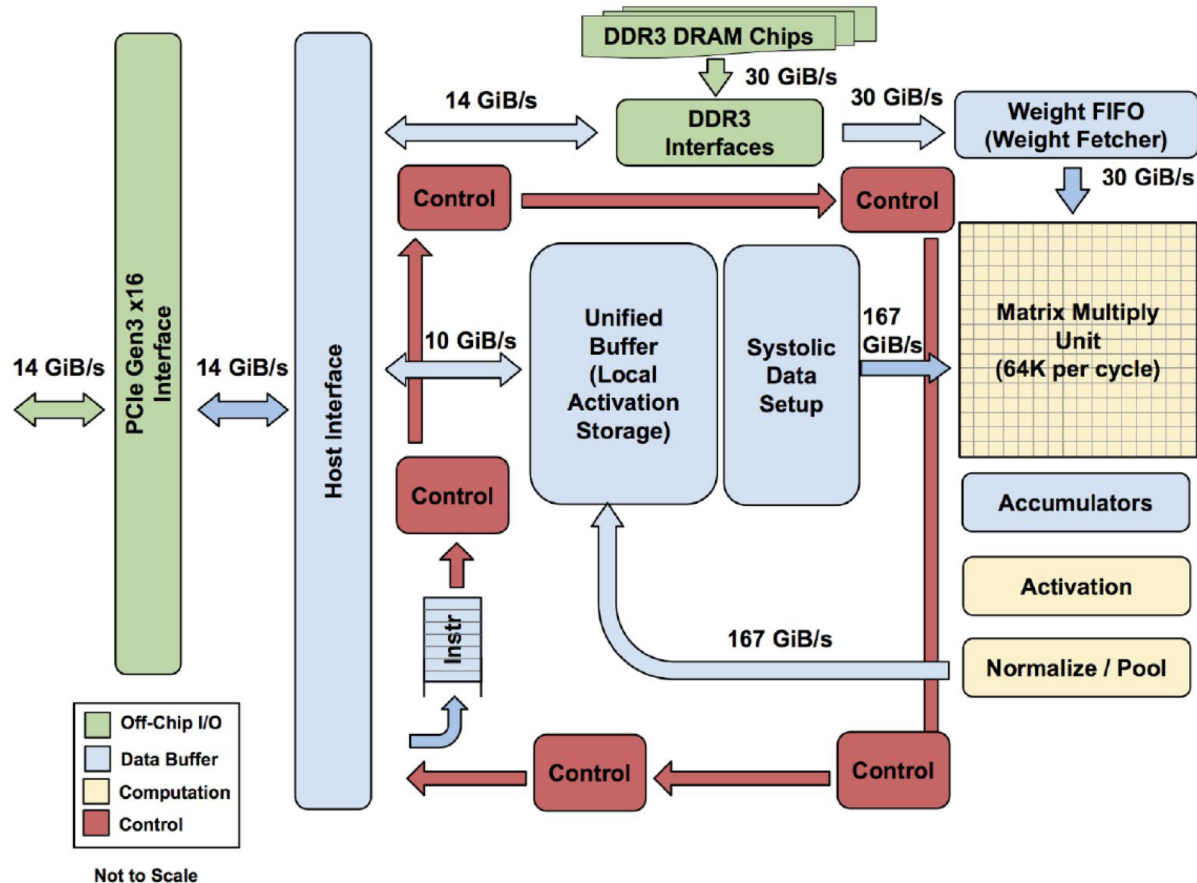


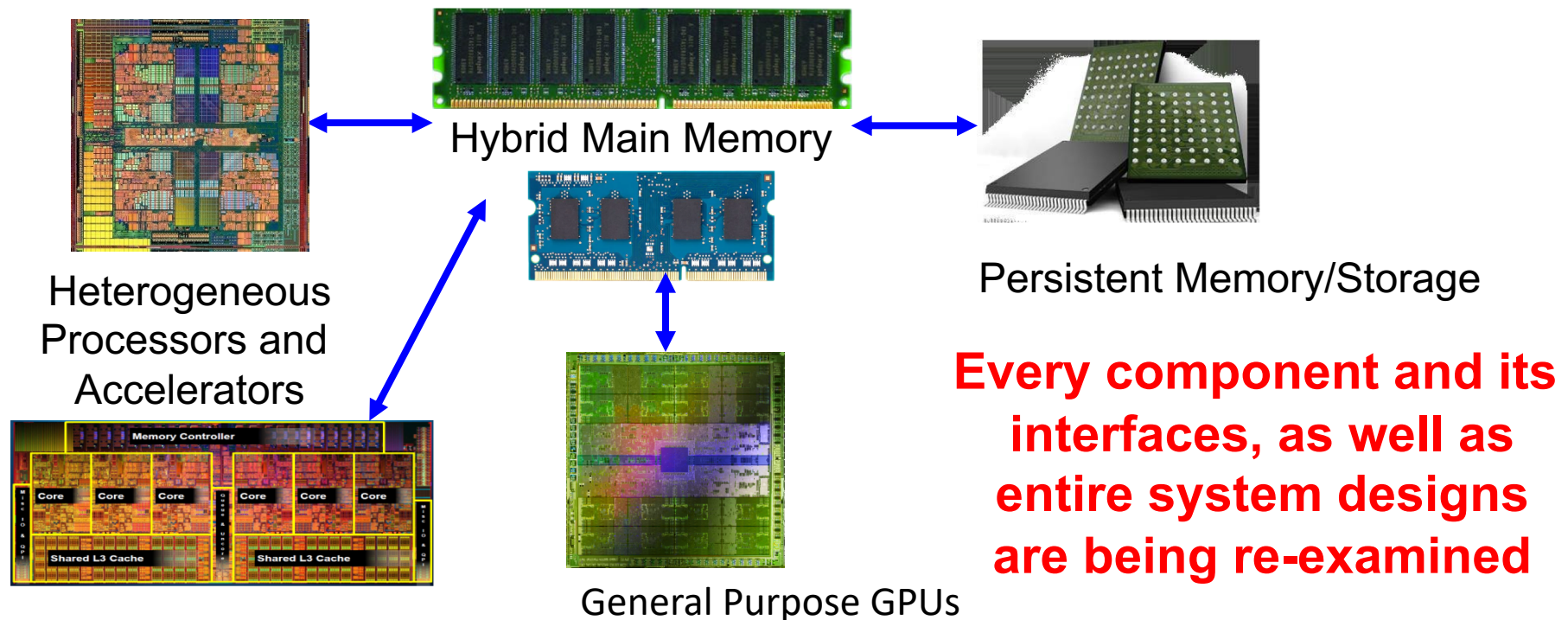
Figure 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.

Many Concepts Being Investigated Today

- **New Computing Paradigms**
 - Processing in Memory
 - Neuromorphic Computing
- **New Accelerators**
 - Machine Learning
 - Graph Analytics
 - Genome Analysis
- **New Systolic Architectures**
- **New Memories**

Computer Architecture Today

- Computing landscape is very different from 10-20 years ago
- Applications and technology both demand novel architectures



Computer Architecture Today (IV)

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- You can invent new paradigms for computation, communication, and storage
- Recommended book: Thomas Kuhn, “[The Structure of Scientific Revolutions](#)” (1962)
 - Pre-paradigm science: no clear consensus in the field
 - Normal science: dominant theory used to explain/improve things (business as usual); exceptions considered anomalies
 - Revolutionary science: underlying assumptions re-examined

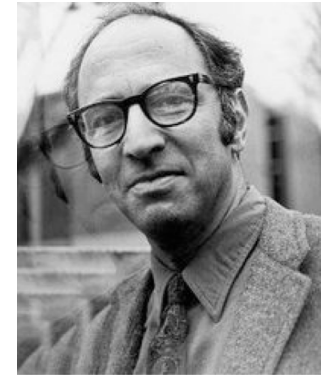
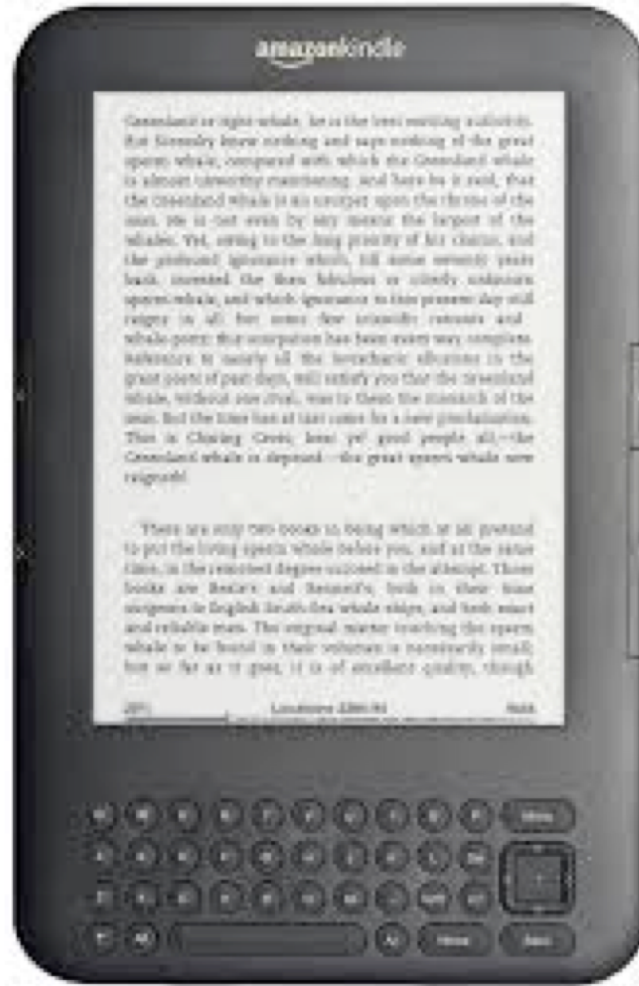
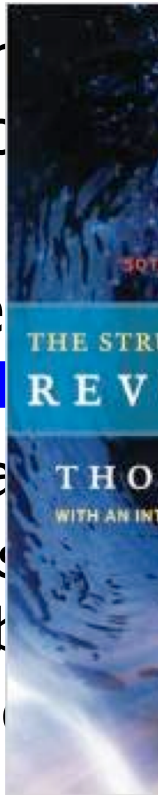
Computer Architecture Today (IV)

- You can revolutionize the way computers are built, if you understand both hardware and software (and change each accordingly)

- You can improve communication

- Recommendation of Scientific Literature

- Pre-prepare things (books)
- Normal science
- Revolutionary science



Structure of
 Normal science
 anomalies
 examined

Takeaways

- It is an exciting time to be understanding and designing computing architectures
- Many challenging and exciting problems in platform design
 - That no one has tackled (or thought about) before
 - That can have huge impact on the world's future
- Driven by huge hunger for data (Big Data), new applications (ML/AI, graph analytics, genomics), ever-greater realism, ...
 - We can easily collect more data than we can analyze/understand
- Driven by significant difficulties in keeping up with that hunger at the technology layer
 - Five walls: Energy, reliability, complexity, security, scalability

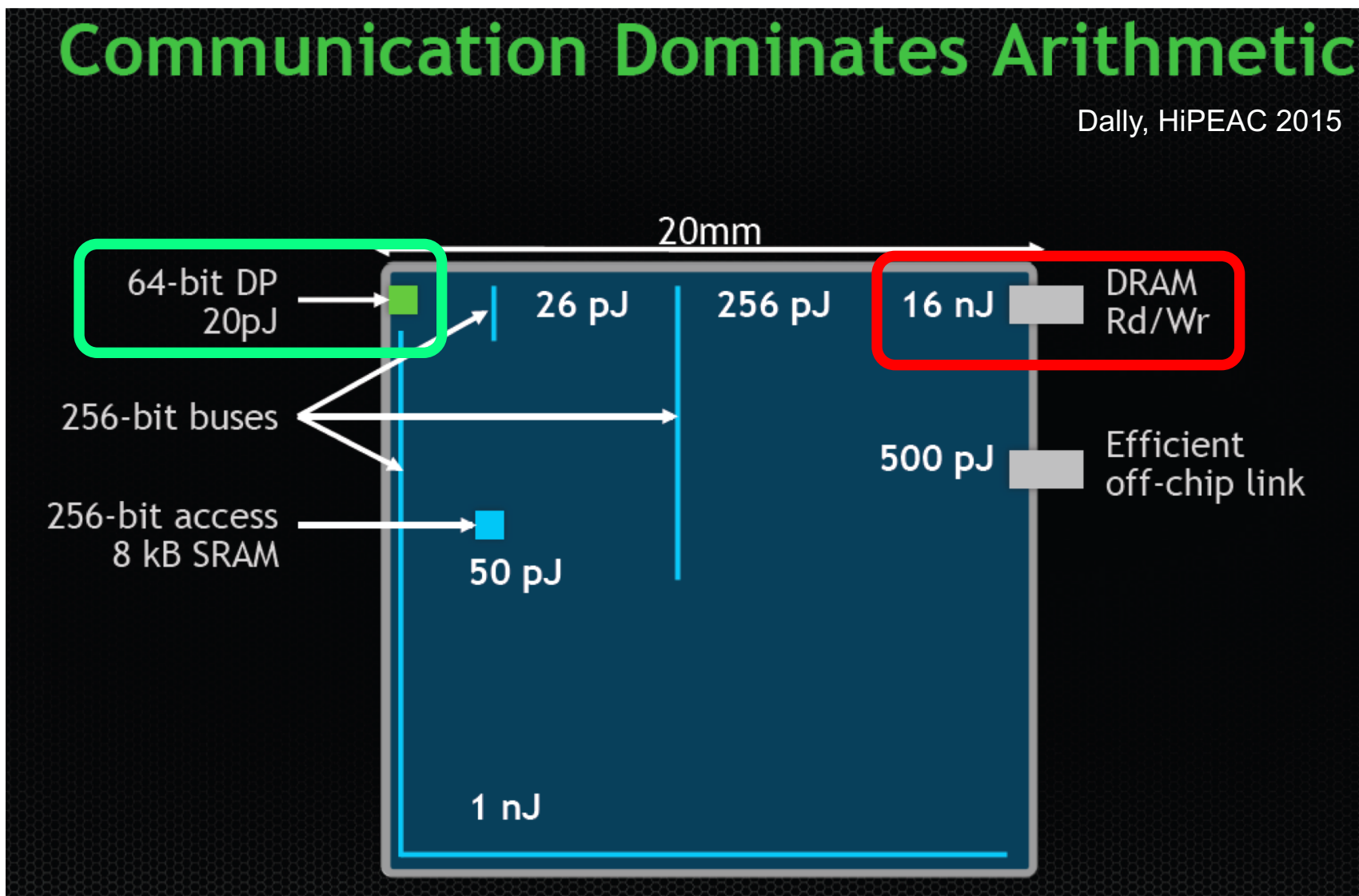
Increasingly Demanding Applications

- Dream, and they will come

Increasingly Diverging/Complex Tradeoffs

Communication Dominates Arithmetic

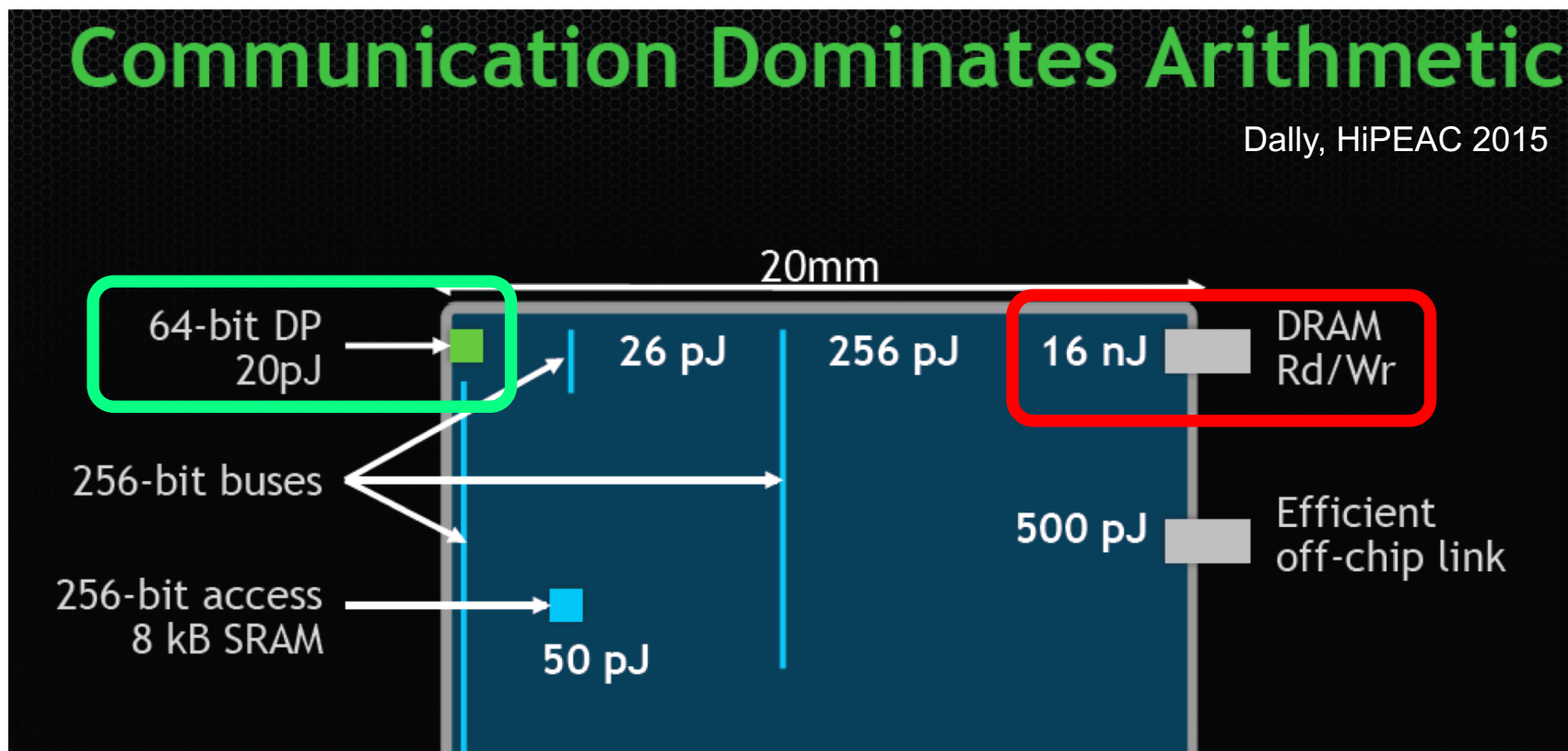
Dally, HiPEAC 2015



Increasingly Diverging/Complex Tradeoffs

Communication Dominates Arithmetic

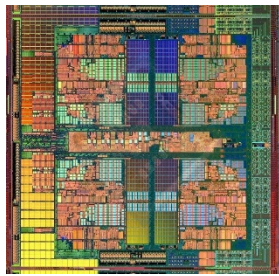
Dally, HiPEAC 2015



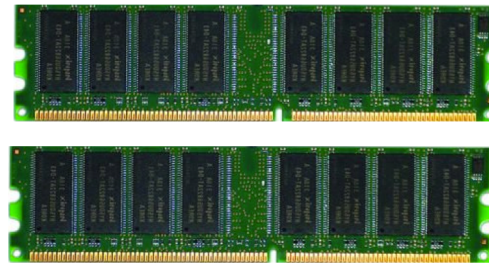
A memory access consumes $\sim 1000X$ the energy of a complex addition

Increasingly Complex Systems

Past systems



Microprocessor



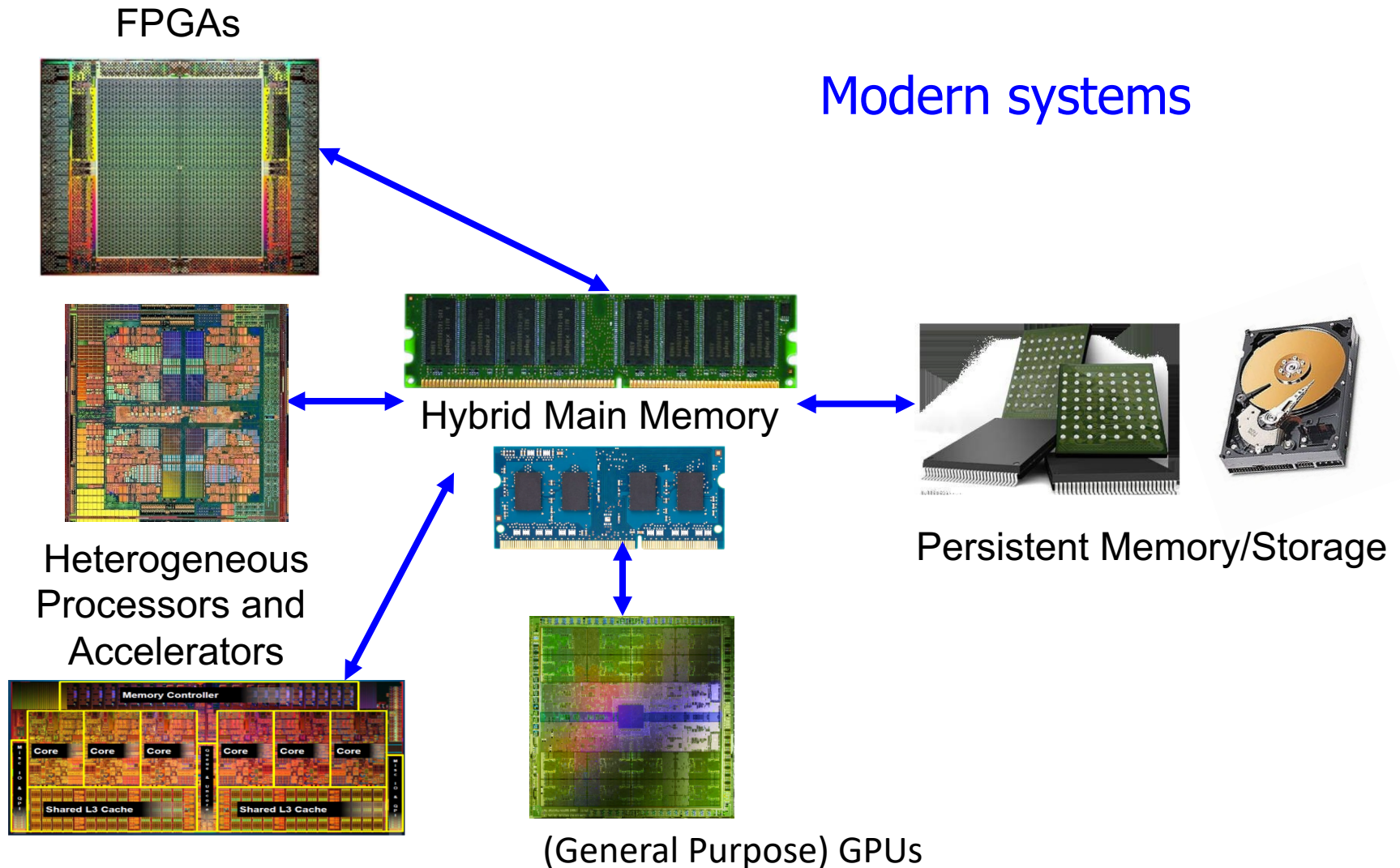
Main Memory



Storage (SSD/HDD)



Increasingly Complex Systems



The Role of This Course

Seminar in Comp Arch

- We will cover **fundamental** and **cutting-edge** research papers in computer architecture
- Multiple components that are aimed at improving students'
 - **technical skills** in computer architecture
 - **critical thinking and analysis**
 - **technical presentation** of concepts and papers
 - in both spoken and written forms
 - **familiarity with key research directions**

Key Goal

(Learn how to)
rigorously
analyze, present, discuss
papers and ideas
in computer architecture

Steps to Achieve the Key Goal

■ Steps for the Presenter

- Read
- Absorb, read more (other related works)
- Critically analyze; think; synthesize
- Prepare a clear and rigorous talk
- Present
- Answer questions
- Analyze and synthesize (in meeting, after, and at course end)

■ Steps for the Participants

- Discuss
- Ask questions
- Analyze and synthesize (in meeting, after, and at course end)

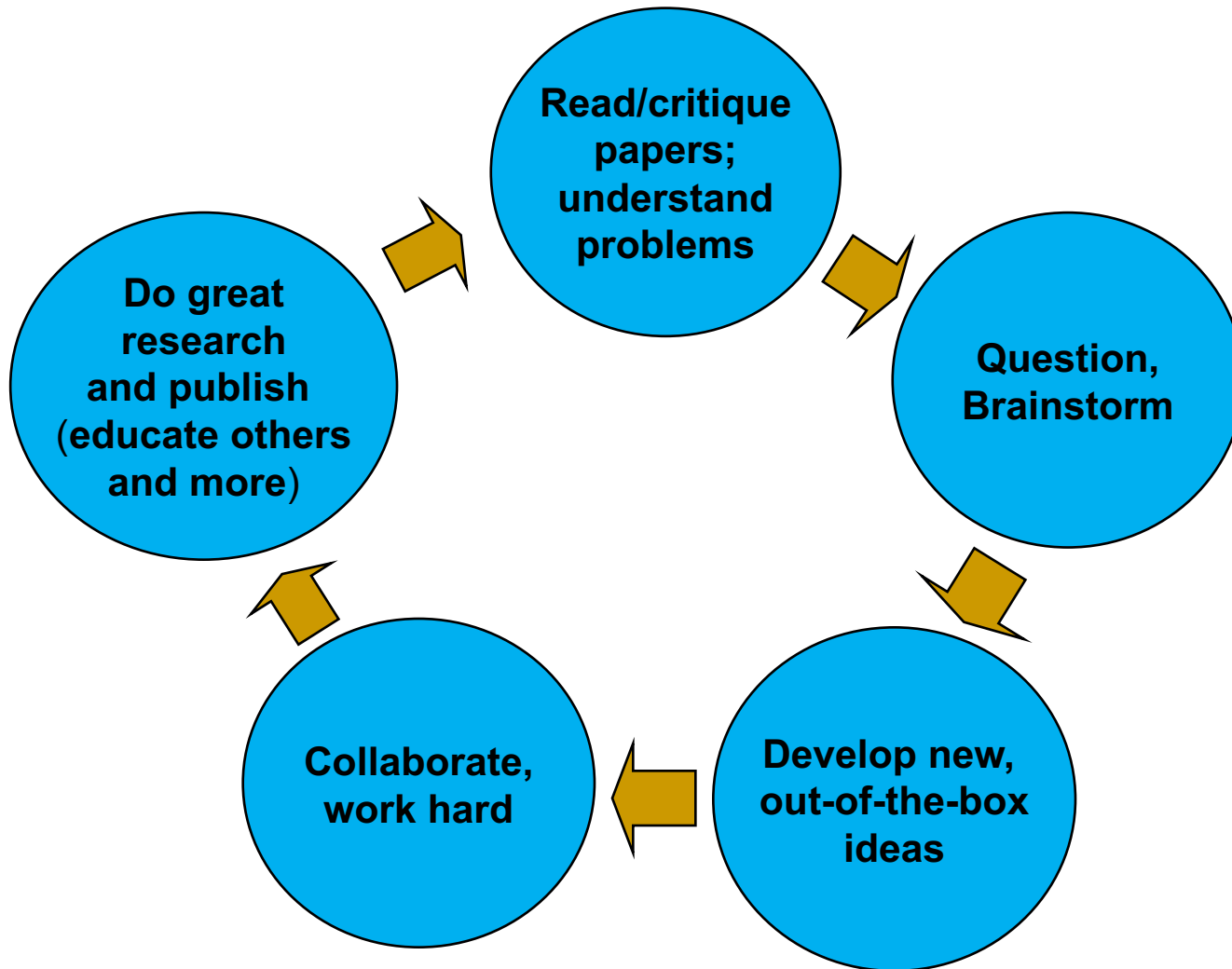
Topics of Papers and Discussion

- hardware security;
- architectural acceleration mechanisms for key applications like machine learning, graph processing and bioinformatics;
- memory systems;
- interconnects;
- processing inside memory;
- various fundamental and emerging ideas/paradigms in computer architecture;
- hardware/software co-design and cooperation;
- fault tolerance;
- energy efficiency;
- heterogeneous and parallel systems;
- new execution models, etc.

Recap: Some Goals of This Course

- Teach/enable/empower you to:
 - Think critically
 - Think broadly
 - Learn how to understand, analyze and present papers and ideas
 - Get familiar with key first steps in research
 - Get familiar with key research directions

The Virtuous Cycle of Scientific Progress



Course Info and Logistics

Course Info: Who Are We?



■ Onur Mutlu

- ❑ Full Professor @ ETH Zurich CS, since September 2015
- ❑ Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-...
- ❑ PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
- ❑ <https://people.inf.ethz.ch/omutlu/>
- ❑ omutlu@gmail.com (Best way to reach me)
- ❑ <https://people.inf.ethz.ch/omutlu/projects.htm>

■ Research and Teaching in:

- ❑ Computer architecture, computer systems, hardware security, bioinformatics
- ❑ Memory and storage systems
- ❑ Hardware security, safety, predictability
- ❑ Fault tolerance
- ❑ Hardware/software cooperation
- ❑ Architectures for bioinformatics, health, medicine
- ❑ ...

Course Info: Who Are We?

■ Instructors:

- Dr. Juan Gomez Luna
- Dr. Mohammed Alser

■ Teaching Assistants

- Dr. Jawad Haj-Yahya
- Dr. Lois Orosa
- Dr. Jisung Park
- Rahul Bera
- Can Firtina
- Nika Mansouri Ghiasi
- Hasan Hassan
- Geraldo De Oliveira Junior
- Jeremie Kim
- Konstantinos Kanellopoulos
- Minesh Patel
- Giray Yaglikci

■ Get to know them and their research

- They will be your mentors – you will have to meet them at least twice before your presentations

Course Requirements and Expectations

- **Attendance required for all meetings**
 - Sign in sheet
- **Each student presents one paper**
 - Prepare for presentation with engagement from the mentor
 - Full presentation + questions + discussion
- **Non-presenters participate during the meeting**
 - Ask questions, contribute thoughts/ideas
 - Better if you read/skim the paper beforehand
- **Everyone comments on papers in the online review system**
 - After presentation
- **Write synthesis report at the end of semester (sample synthesis report online)**

Course Website

- https://safari.ethz.ch/architecture_seminar/fall2019
- All course materials to be posted
- Plus other useful information for the course
- Check frequently for announcements and due dates

Homework 0

- Due Sept. 26
 - https://safari.ethz.ch/architecture_seminar/fall2019/doku.php?id=homework
- Information about yourself
- All future grading is predicated on homework 0
- If it is not submitted on time, we cannot schedule you for a presentation.

Paper Review Preferences

- Due Sept. 24
- Check the website for instructions
- If it is not submitted on time, we cannot schedule you for a presentation.

How to Deliver a Good Talk

Anatomy of a Good Paper Review (Talk)

- 0: Title, Authors, Venue
- 1: Summary
 - What is the problem the paper is trying to solve?
 - What are the key ideas of the paper? Key insights?
 - What are the key mechanisms? What is the implementation?
 - What are the key results? Key conclusions?
- 2: Strengths (most important ones)
 - Does the paper solve the problem well? Is it well written? ...
- 3: Weaknesses (most important ones)
 - This is where you should **think critically**. Every paper/idea has a weakness. This does not mean the paper is necessarily bad. It means there is room for improvement and future research can accomplish this.
- 4: Thoughts/Ideas: Can you do better? Present your ideas.
- 5: Takeaways: What you learned/enjoyed/disliked? Why?
- 6: Discussion starters and questions.
- Review should be short and concise (20 minutes or < one page)

Suggested Paper Discussion Format

- Problem & Goal
- Key Ideas/solution
- Novelty
- Mechanisms & Implementation
- Major Results
- Takeaways/Conclusions

**~25 minute
Summary**

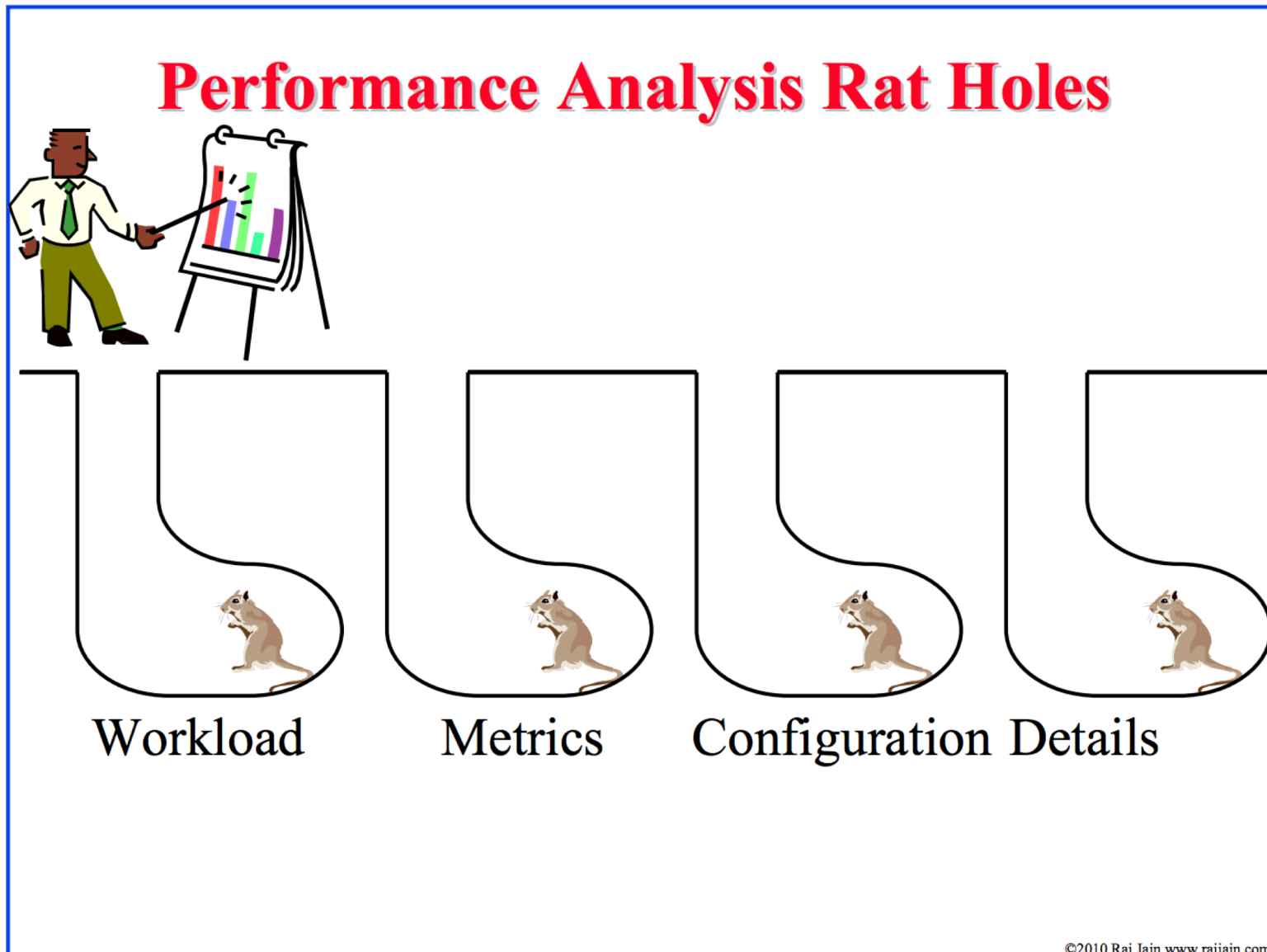
- Strengths
- Weaknesses
- Alternatives
- New ideas/problems
- **Brainstorming and Discussion**

**~10 min Critique
plus
~10 min Discussion**

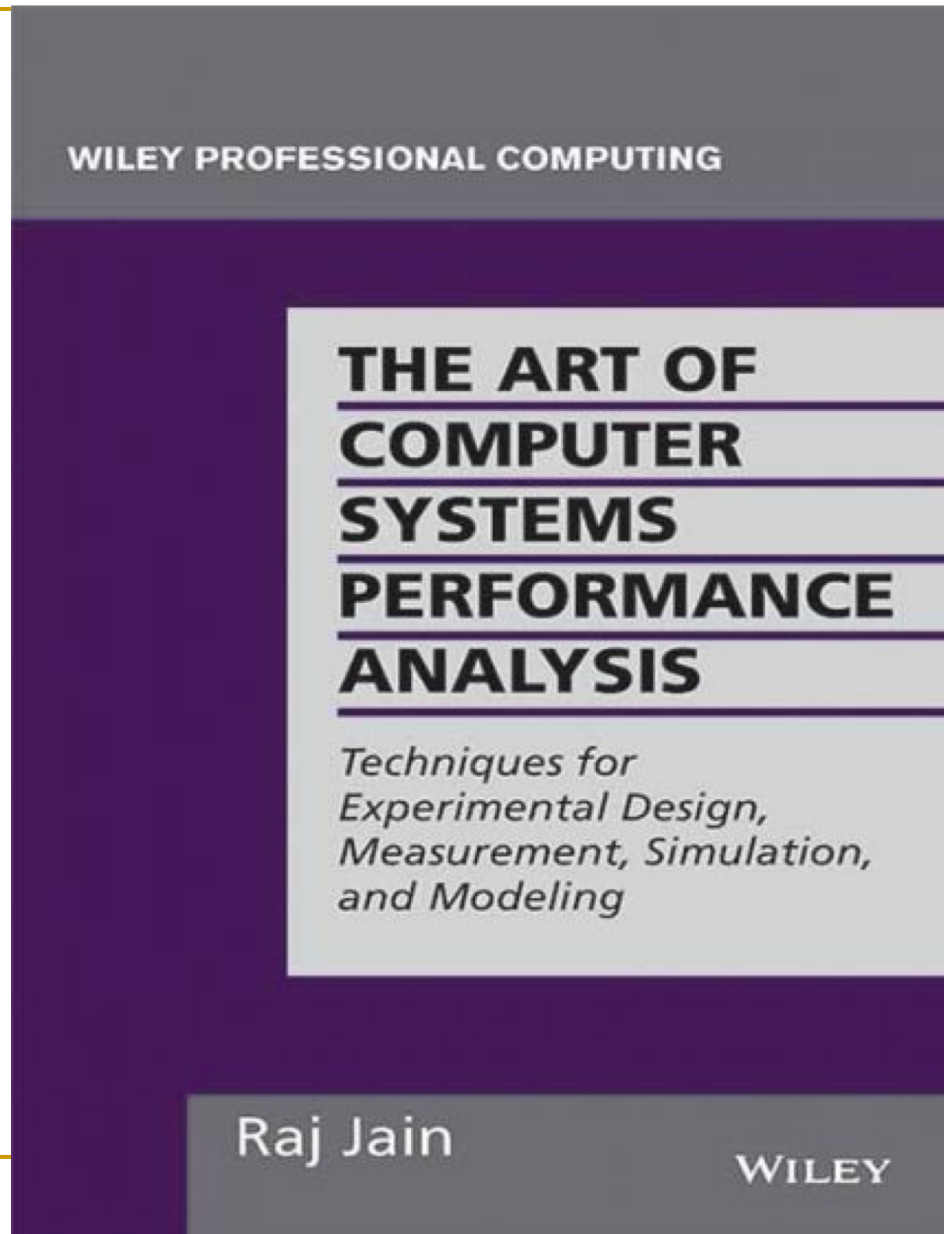
More Advice on Paper Review Talk

- When doing the paper reviews and analyses, be very critical
- Always think about better ways of solving the problem or related problems
 - Question the problem as well
 - Read background papers (both past and future)
- This is how things progress in science and engineering (or anywhere), and how you can make big leaps
 - By critical analysis
- A few sample text reviews provided online

Try to Avoid Rat Hole Discussions



Aside: A Recommended Book



Raj Jain, "[The Art of Computer Systems Performance Analysis](#)," Wiley, 1991.

10.8 DECISION MAKER'S GAMES

Even if the performance analysis is correctly done and presented, it may not be enough to persuade your audience—the decision makers—to follow your recommendations. The list shown in Box 10.2 is a compilation of reasons for rejection heard at various performance analysis presentations. You can use the list by presenting it immediately and pointing out that the reason for rejection is not new and that the analysis deserves more consideration. Also, the list is helpful in getting the competing proposals rejected!

There is no clear end of an analysis. Any analysis can be rejected simply on the grounds that the problem needs more analysis. This is the first reason listed in Box 10.2. The second most common reason for rejection of an analysis and for endless debate is the workload. Since workloads are always based on the past measurements, their applicability to the current or future environment can always be questioned. Actually workload is one of the four areas of discussion that lead a performance presentation into an endless debate. These “rat holes” and their relative sizes in terms of time consumed are shown in Figure 10.26. Presenting this cartoon at the beginning of a presentation helps to avoid these areas.

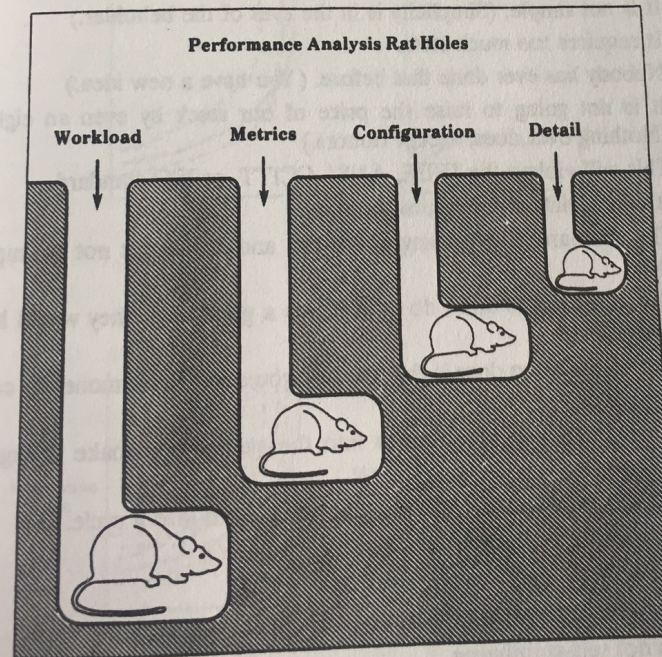


FIGURE 10.26 Four issues in performance presentations that commonly lead to endless discussion.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

Box 10.2 Reasons for Not Accepting the Results of an Analysis

1. This needs more analysis.
2. You need a better understanding of the workload.
3. It improves performance only for long I/O's, packets, jobs, and files, and most of the I/O's, packets, jobs, and files are short.
4. It improves performance only for short I/O's, packets, jobs, and files, but who cares for the performance of short I/O's, packets, jobs, and files; its the long ones that impact the system.
5. It needs too much memory/CPU/bandwidth and memory/CPU/bandwidth isn't free.
6. It only saves us memory/CPU/bandwidth and memory/CPU/bandwidth is cheap.
7. There is no point in making the networks (similarly, CPUs/disks/...) faster; our CPUs/disks (any component other than the one being discussed) aren't fast enough to use them.
8. It improves the performance by a factor of x , but it doesn't really matter at the user level because everything else is so slow.
9. It is going to increase the complexity and cost.
10. Let us keep it simple stupid (and your idea is not stupid).
11. It is not simple. (Simplicity is in the eyes of the beholder.)
12. It requires too much state.
13. Nobody has ever done that before. (You have a new idea.)
14. It is not going to raise the price of our stock by even an eighth. (Nothing ever does, except rumors.)
15. This will violate the IEEE, ANSI, CCITT, or ISO standard.
16. It may violate some future standard.
17. The standard says nothing about this and so it must not be important.
18. Our competitors don't do it. If it was a good idea, they would have done it.
19. Our competition does it this way and you don't make money by copying others.
20. It will introduce randomness into the system and make debugging difficult.
21. It is too deterministic; it may lead the system into a cycle.
22. It's not interoperable.
23. This impacts hardware.
24. That's beyond today's technology.
25. It is not self-stabilizing.
26. Why change—it's working OK.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

More Advice on Talks

- Kayvon Fatahalian, “Tips for Giving Clear Talks”
 - <http://graphics.stanford.edu/~kayvonf/misc/cleartalktips.pdf>
 - Many useful and simple principles here

“Every sentence matters”

“The audience prefers not to think” (about things you can just tell them)

**“Surprises are bad”: say why before what
(indicate why you are saying something before you say it)**

Explain every figure, graph, or equation

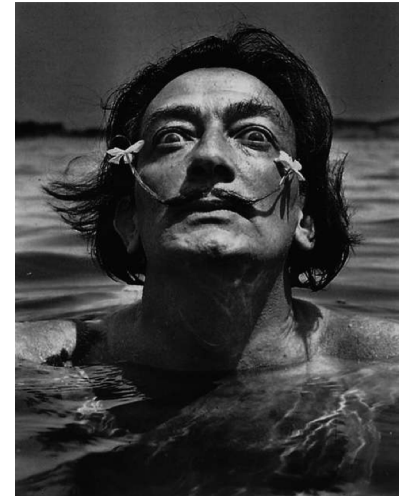
When improving the talk, the audience is always right

Who Painted This Painting?



Salvador Dali @ 1924

What About This?



Salvador Dali @ 1937

Takeaway

Learn the basic principles

before you

consciously choose to break them

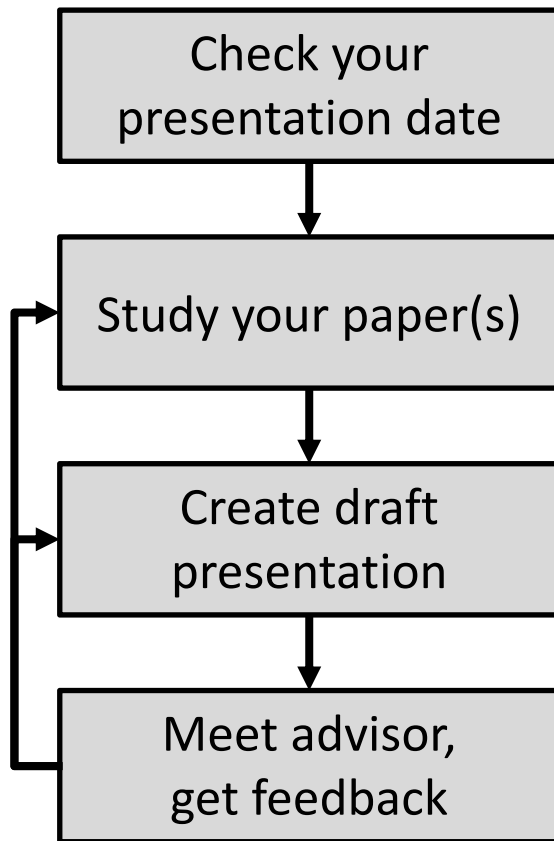
How to Participate

How to Make the Best Out of This?

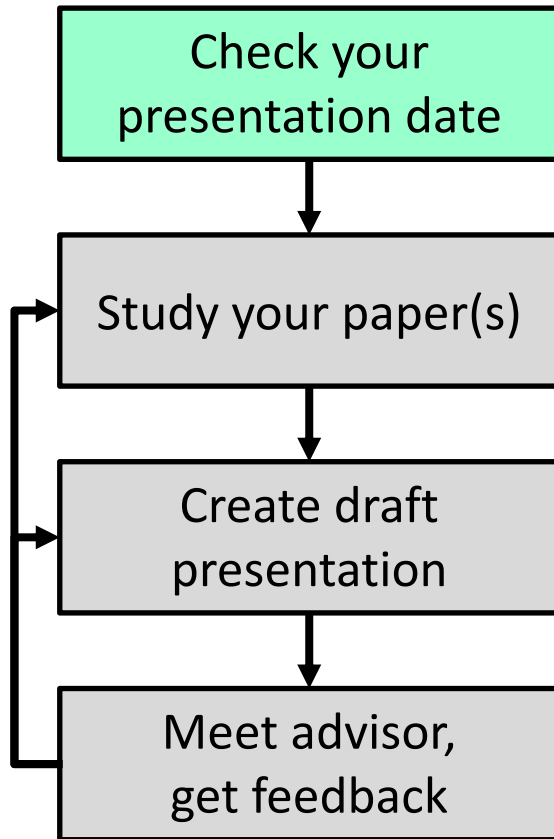
- Come prepared → Read and critically evaluate the paper
- Think new ideas
- Bring discussion points and questions; read other papers
- Be critical
- Brainstorm – be open to new ideas
- Pay attention and discuss+contribute
- Participate online before and after each meeting

Guided Talk Preparation

Preparing a Talk

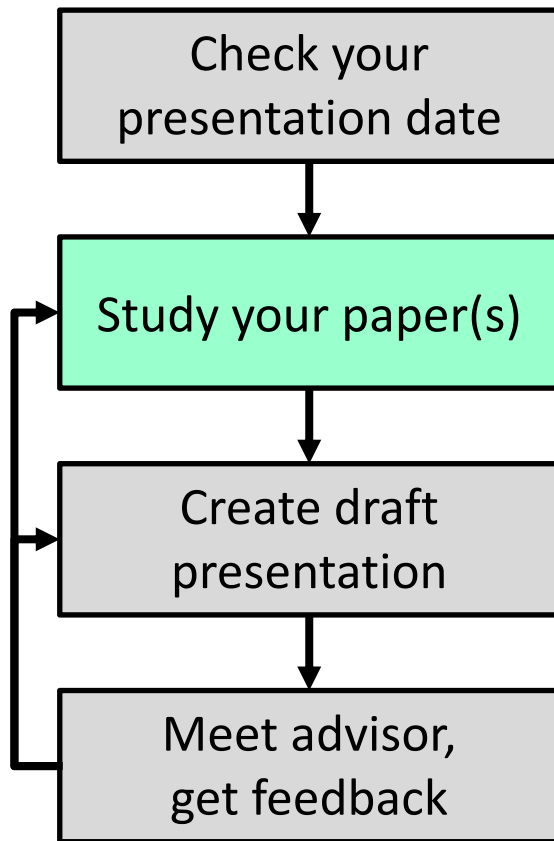


Preparing a Talk: Start Early



- Preparing a good presentation takes time
- Start early!

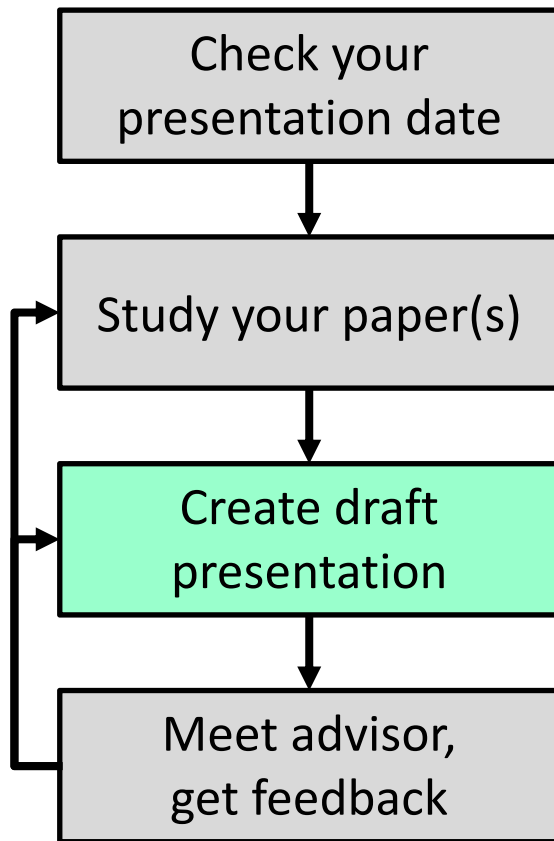
Preparing a Talk: Study Paper



- 3 'C's of reading
 - *Carefully*: look up terms, possibly read cited papers
 - *Critically*: find limitations, flaws
 - *Creatively*: think of improvements
- Try examples by hand
- Try tools if available

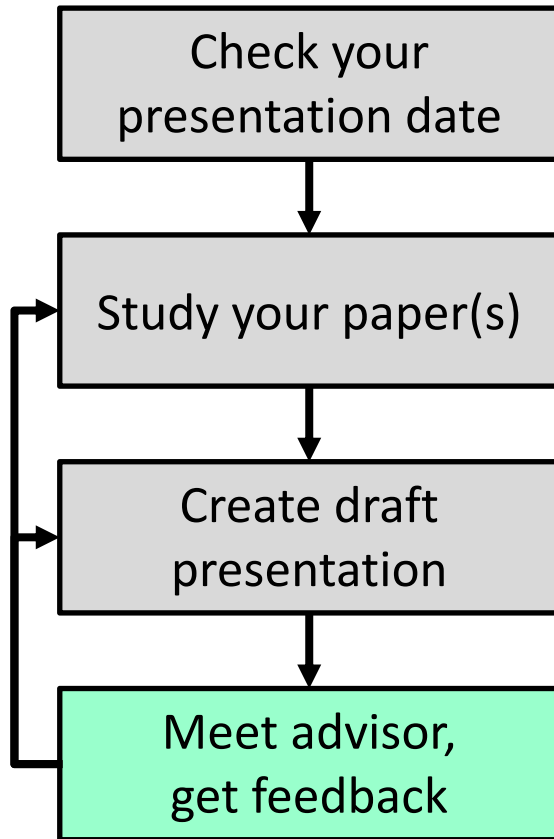
- Consult with TA if questions

Preparing a Talk: Create Draft



- Explain the motivation for the work
- Clearly present the technical solution and results
 - Include a demo if appropriate
- Outline limitations or improvements
- Focus on the key concepts
 - Do not present all of the details

Preparing a Talk: Get Feedback



- Prepare for the meeting
 - Schedule early
 - Send slides in advance
 - Write down questions
 - Make sure you address feedback
 - Take notes
 - Meetings are mandatory!
 - At least one week before the talk
 - Two meetings
-

Grading and Feedback

Grading Rubric

- **Quality of your presentation (60%)**
 - ❑ How well did you understand the material?
 - ❑ How well did you present it?
 - ❑ How well did you answer the questions?
 - ❑ Be prepared to explain technical terms
 - ❑ **We will take into account** the difficulty of the paper and the time you had to prepare.
- **Quality of the final synthesis paper (30%)**
 - ❑ How well did you understand some of the papers presented during the seminar?
- **Attendance (10%)**
 - ❑ Did you attend all sessions?
- **Participation (during class and online) (BONUS 10%)**
 - ❑ Did you ask good questions?

Feedback

- We will try to (briefly) discuss strengths/weaknesses of your talk in class
 - Let us know upfront if you would prefer **not** to
- You can arrange a meeting with your TA to get feedback

Expected Schedule

Schedule

- We will meet once a week, with two presentations per session
 - *Next meeting September 26*
 - *Your presentations start the week after (October 3)*
 - 22 presentations in total
 - Each presentation 50 minutes including questions and discussion

 - Paper assignment
 - Will be done online
 - Study the list of papers
 - **Check your email** and be responsive
-

Homework 0

- Due Sept. 26
 - https://safari.ethz.ch/architecture_seminar/fall2019/doku.php?id=homework
- Information about yourself
- All future grading is predicated on homework 0
- If it is not submitted on time, we cannot schedule you for a presentation.

Paper Review Preferences

- Due Sept. 24
- Check the website for instructions
- If it is not submitted on time, we cannot schedule you for a presentation.

Seminar in Computer Architecture Meeting 1: Introduction

Prof. Onur Mutlu

ETH Zürich

Fall 2019

19 September 2019

Seminar in Computer Architecture

Meeting 1: Example Review

Prof. Onur Mutlu

ETH Zürich

Fall 2019

19 September 2019

We Will Briefly Review This Paper

- Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda,
"Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning"
Proceedings of the 44th International Symposium on Microarchitecture (MICRO), Porto Alegre, Brazil, December 2011. Slides (pptx)

Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning

Sai Prashanth Muralidhara
Pennsylvania State University
smuralid@cse.psu.edu

Lavanya Subramanian
Carnegie Mellon University
lsubrama@ece.cmu.edu

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Mahmut Kandemir
Pennsylvania State University
kandemir@cse.psu.edu

Thomas Moscibroda
Microsoft Research Asia
moscitho@microsoft.com

Application-Aware Memory Channel Partitioning

Sai Prashanth Muralidhara § Lavanya Subramanian †

Onur Mutlu † Mahmut Kandemir §

Thomas Moscibroda ‡

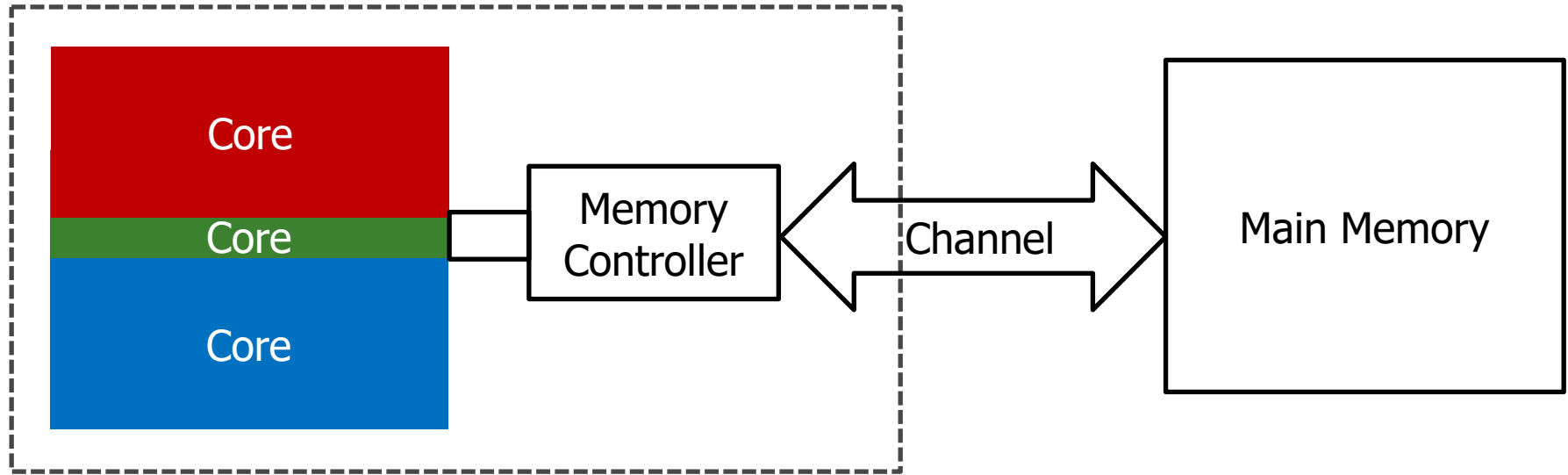
§ **Pennsylvania State University** † **Carnegie Mellon University**

‡ **Microsoft Research**

SAFARI Carnegie Mellon

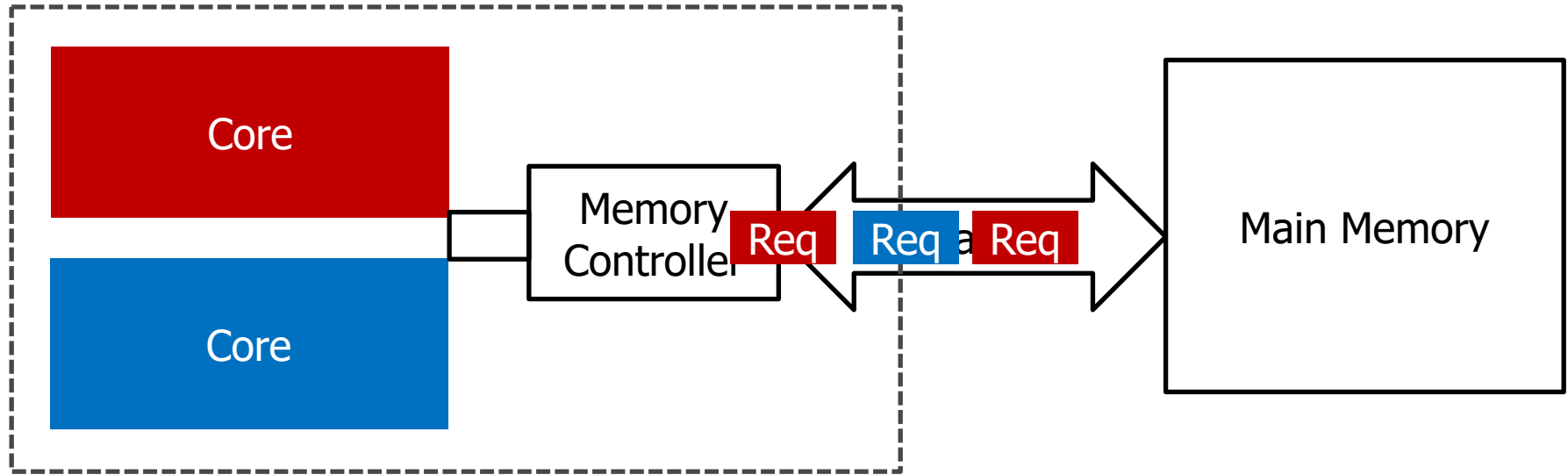
Background, Problem & Goal

Main Memory is a Bottleneck



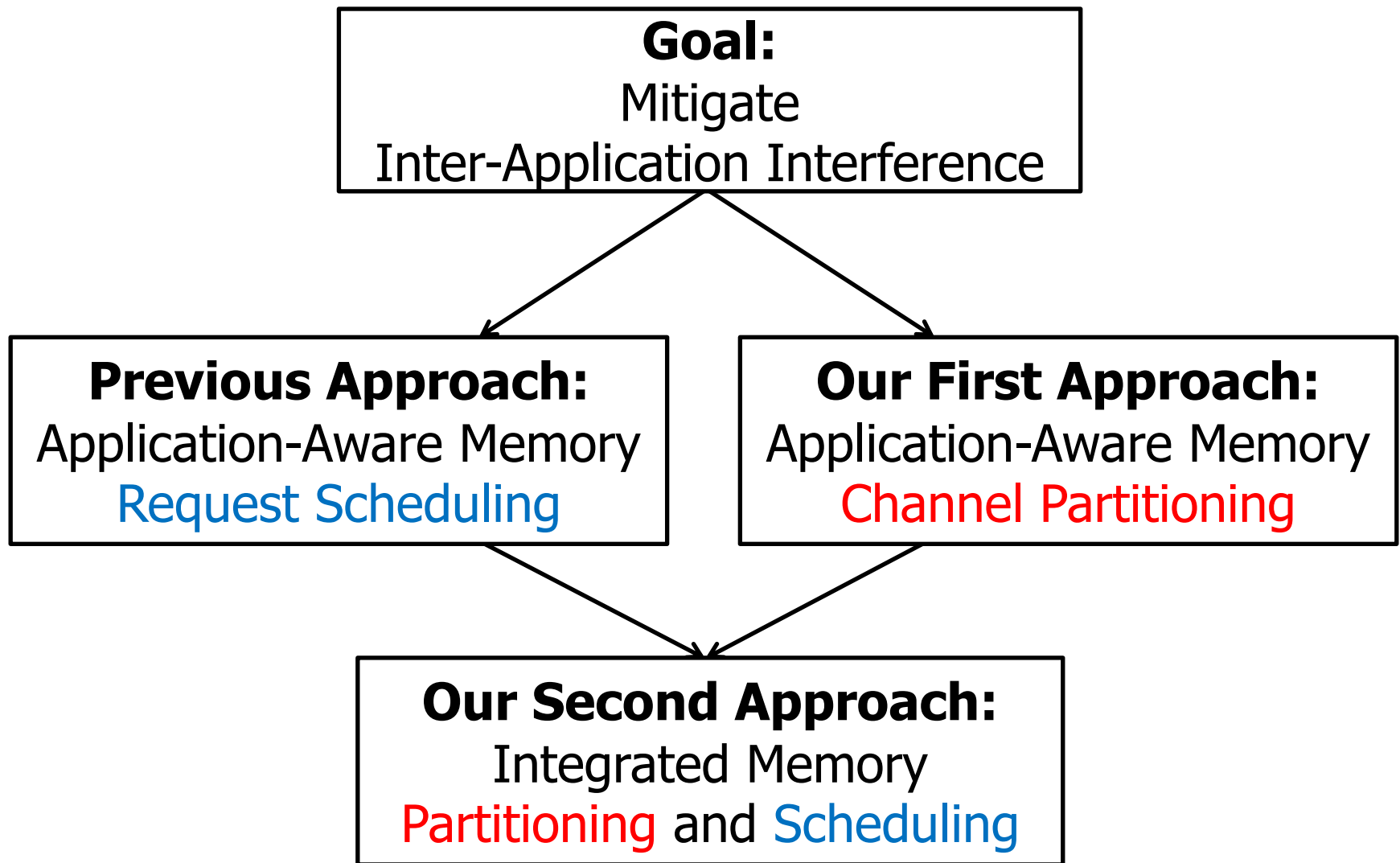
- Main memory latency is long
- Core stalls, performance degrades
- Multiple applications share the main memory

Problem of Inter-Application Interference

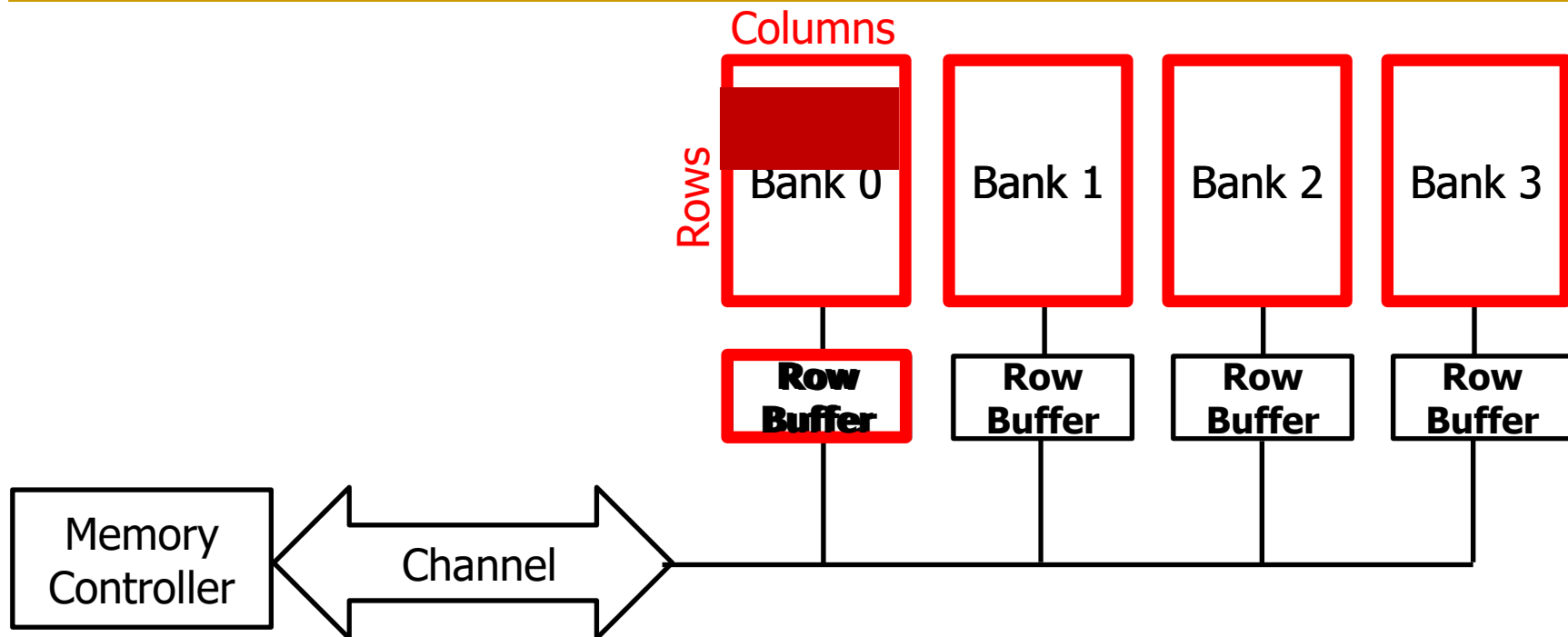


- Applications' requests interfere at the main memory
- This **inter-application interference** degrades system performance
- Problem further exacerbated due to
 - Increasing number of cores
 - Limited off-chip pin bandwidth

Outline



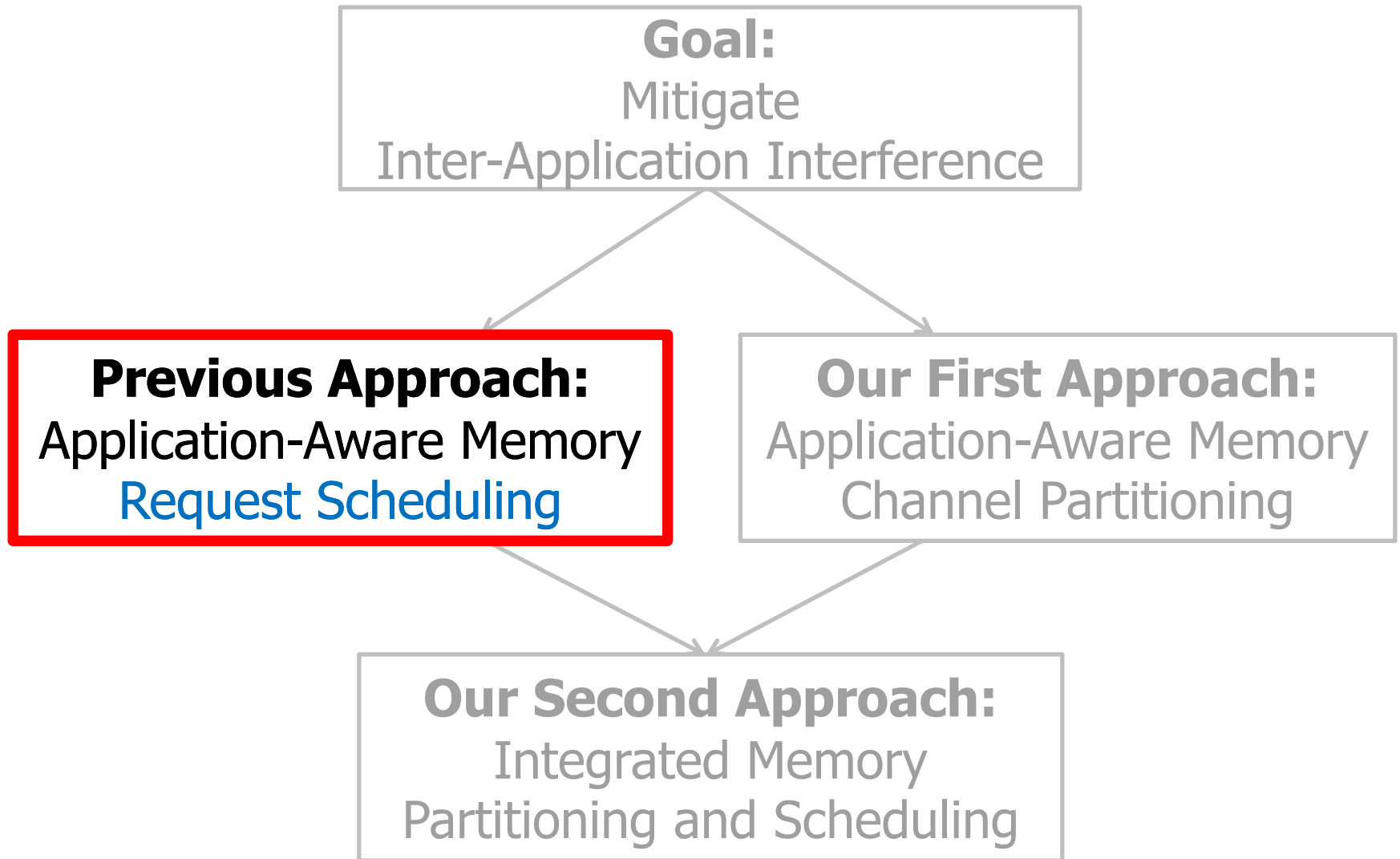
Background: Main Memory



- FR-FCFS memory scheduling policy [Zuravleff et al., US Patent '97; Rixner et al., ISCA '00]
 - Row-buffer hit first
 - Oldest request first
- Unaware of inter-application interference

Novelty

Previous Approach



Application-Aware Memory Request Scheduling

- **Monitor** application memory access characteristics
- **Rank** applications based on memory access characteristics
- **Prioritize** requests at the memory controller, based on ranking

An Example: Thread Cluster Memory Scheduling

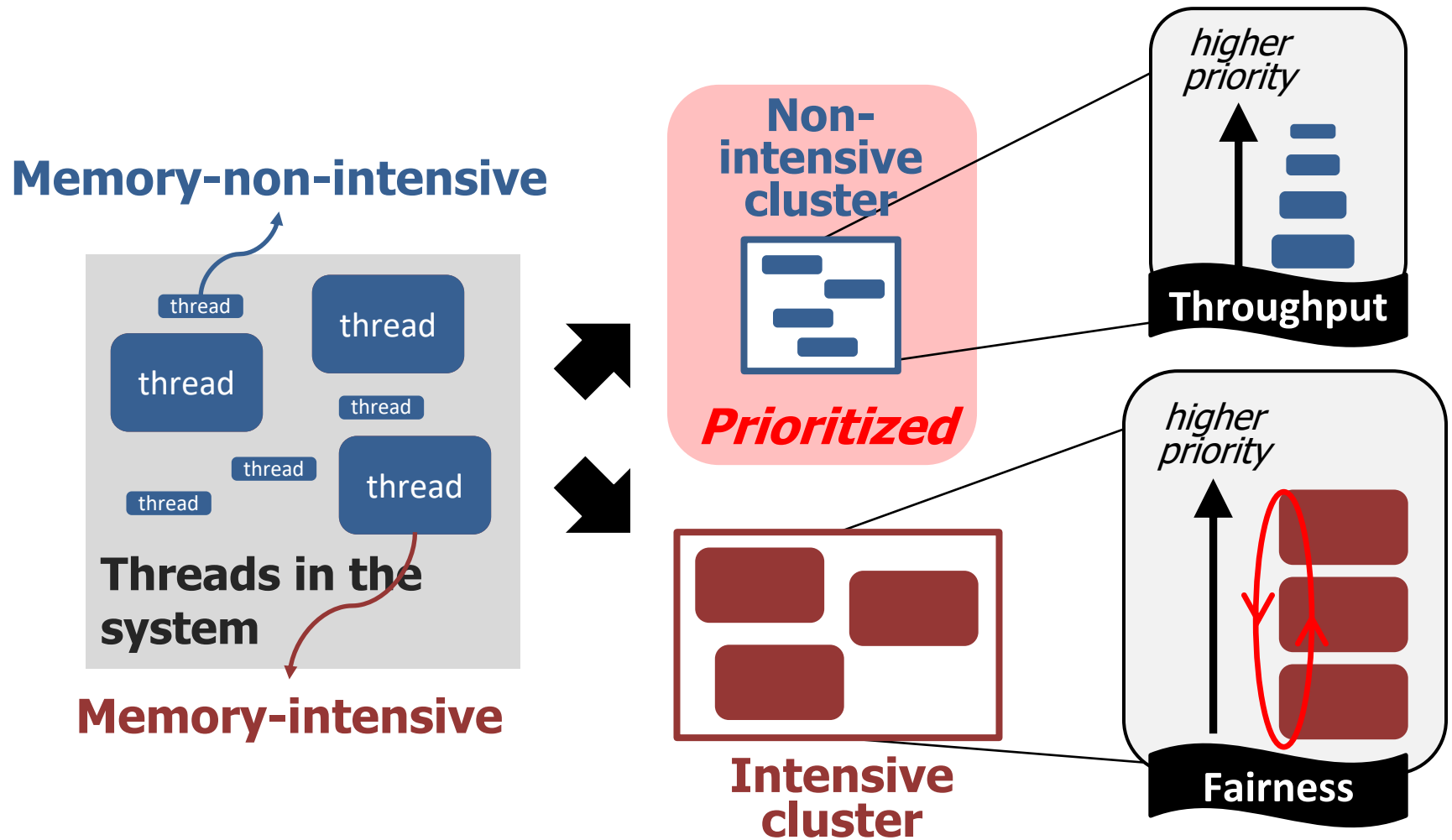


Figure: Kim et al., MICRO 2010

Application-Aware Memory Request Scheduling

Advantages

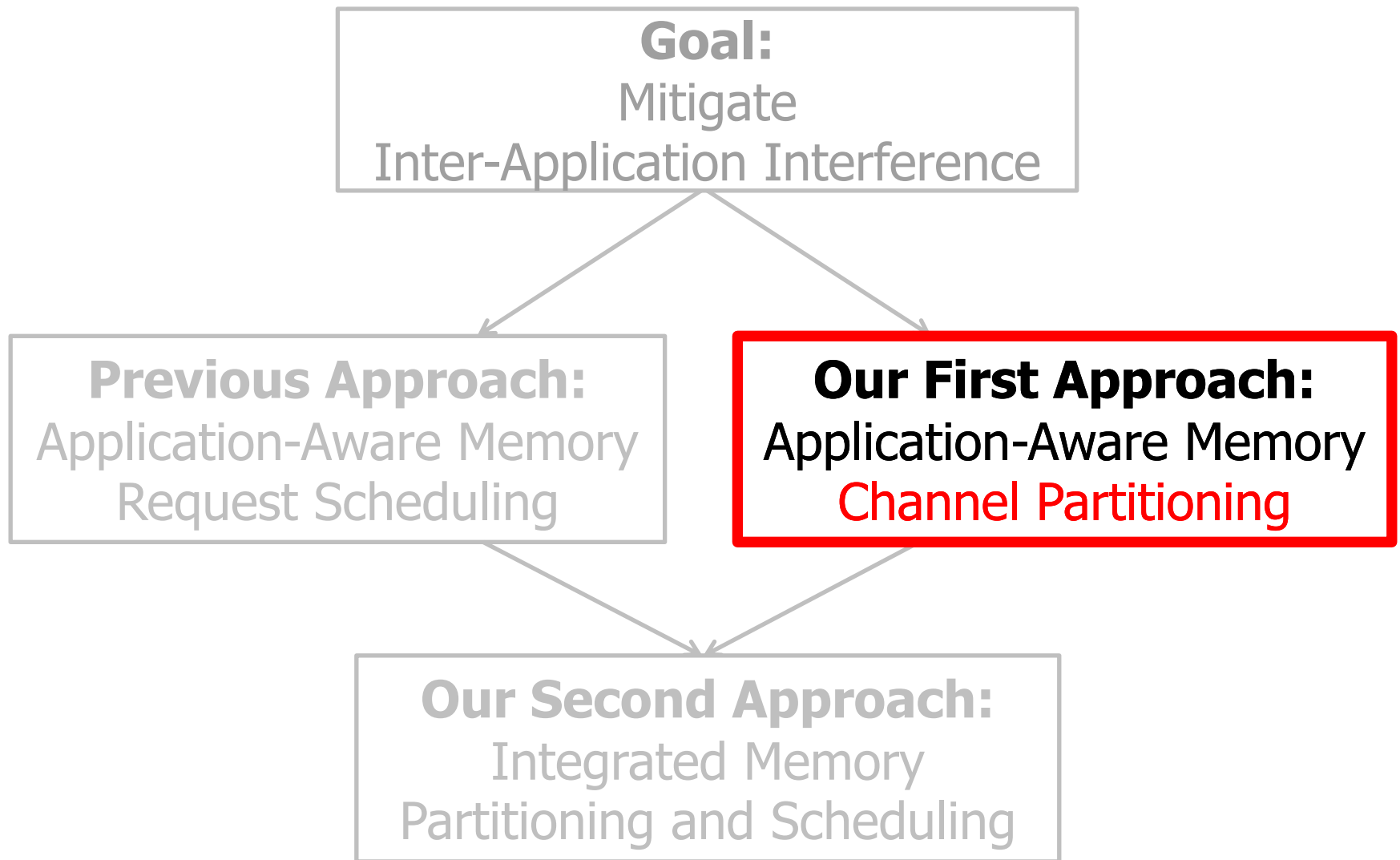
- Reduces interference between applications by request reordering
- Improves system performance

Disadvantages

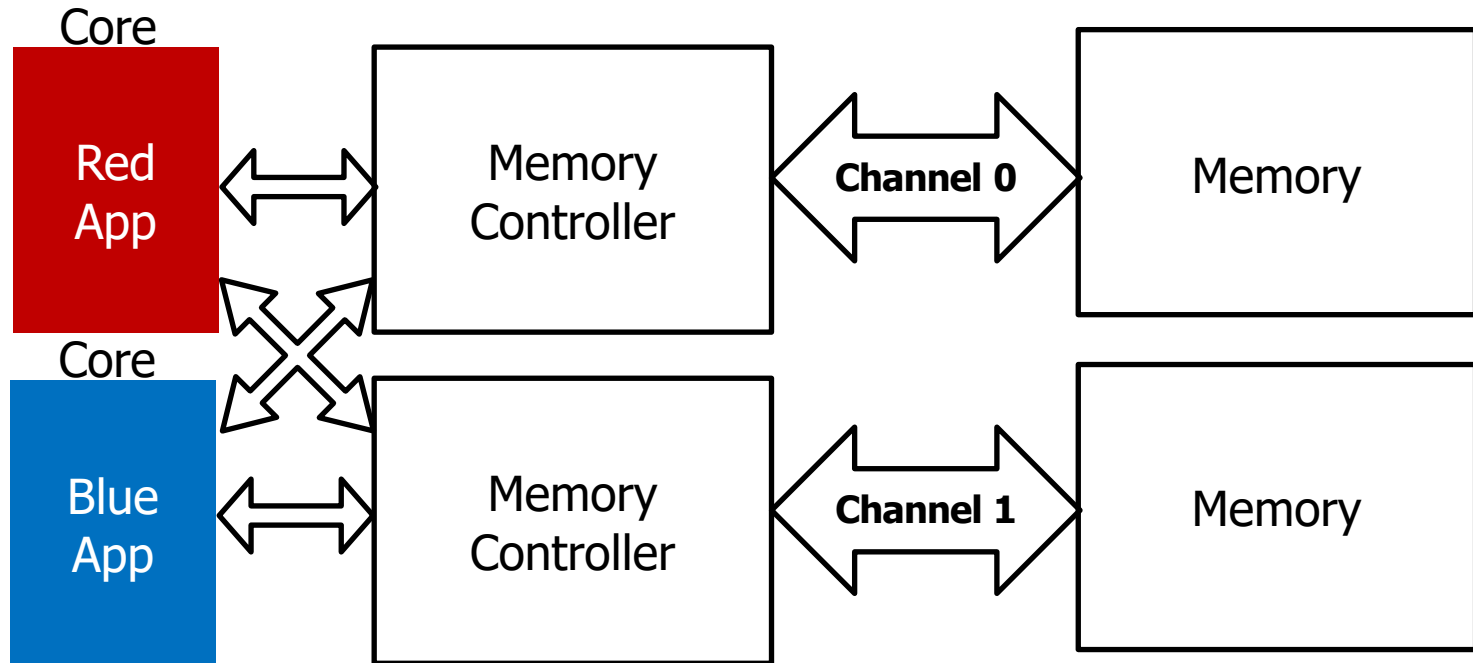
- Requires modifications to memory scheduling logic for
 - Ranking
 - Prioritization
- Cannot completely eliminate interference by request reordering

Key Approach and Ideas

The Paper's Approach

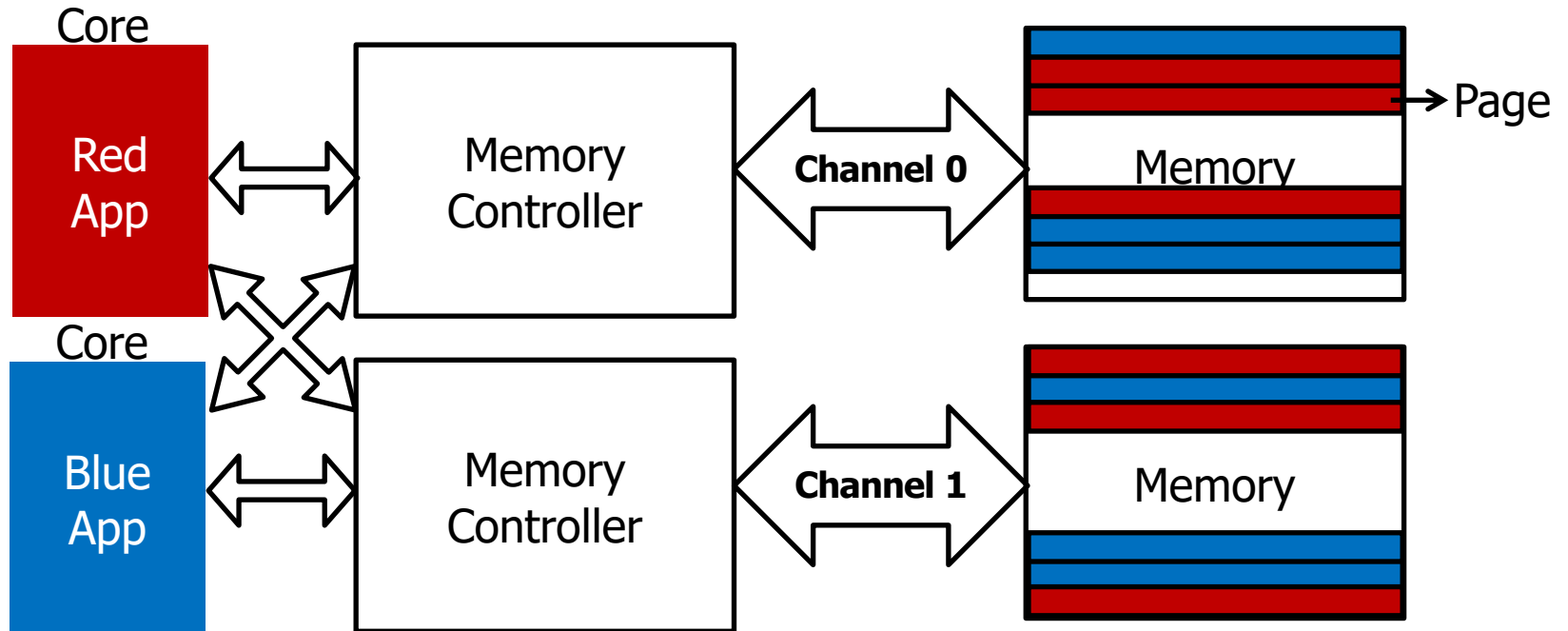


Observation: Modern Systems Have Multiple Channels



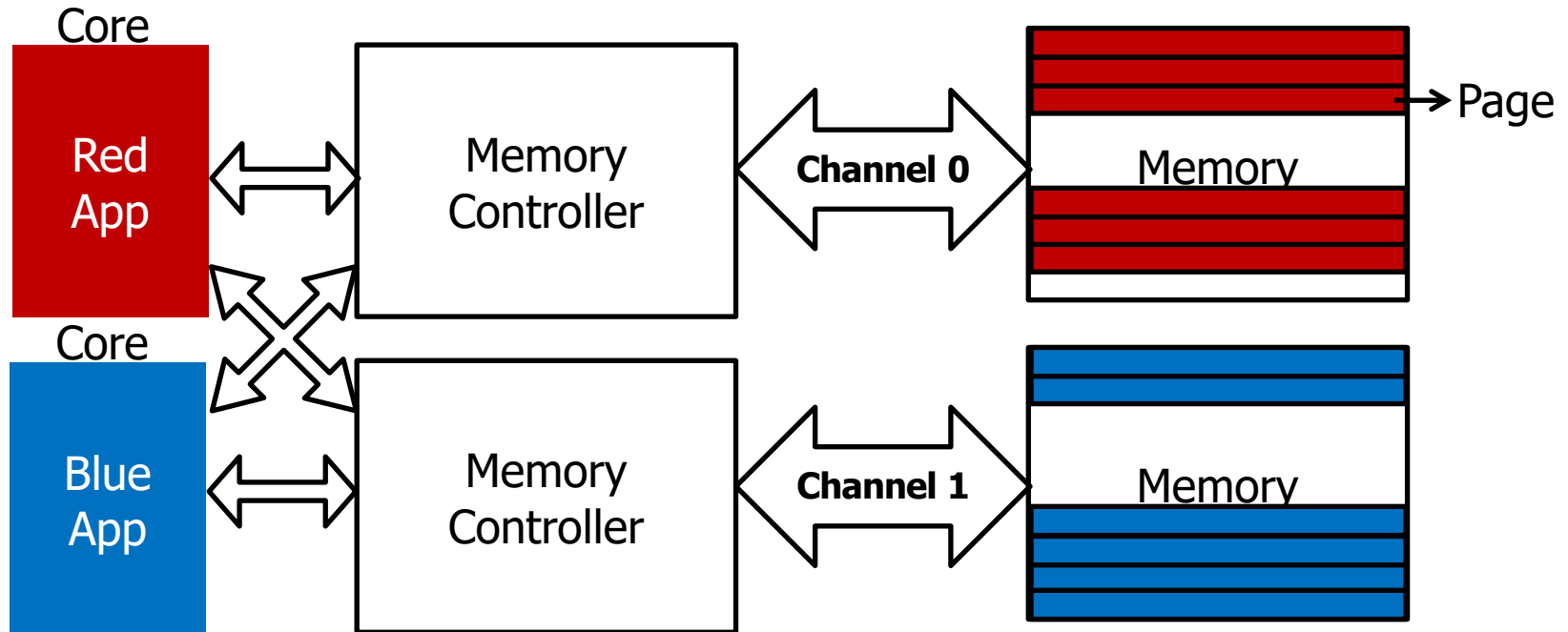
A new degree of freedom
Mapping data across multiple channels

Data Mapping in Current Systems



Causes interference between applications' requests

Partitioning Channels Between Applications



Eliminates interference between applications' requests

Overview: Memory Channel Partitioning (MCP)

■ Goal

- Eliminate harmful interference between applications

■ Basic Idea

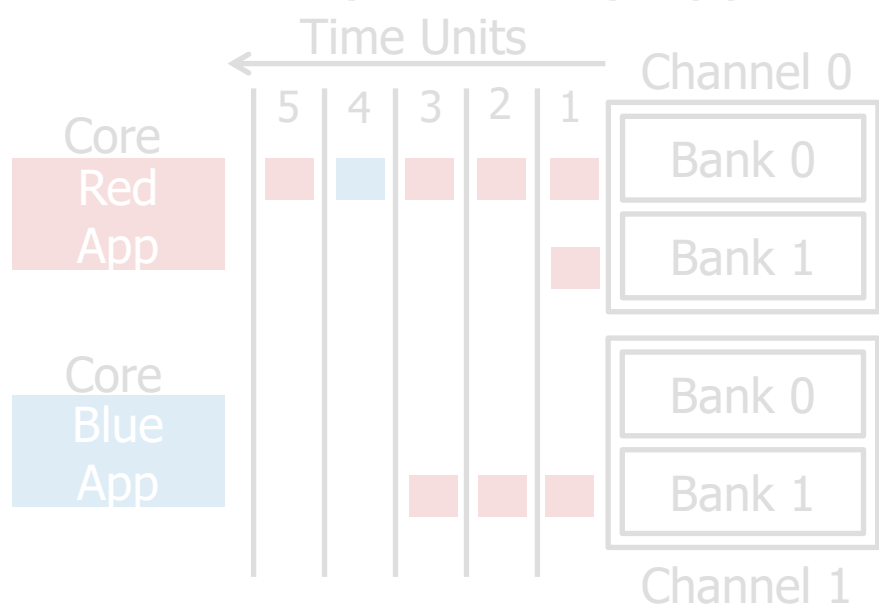
- Map the data of **badly-interfering applications** to different channels

■ Key Principles

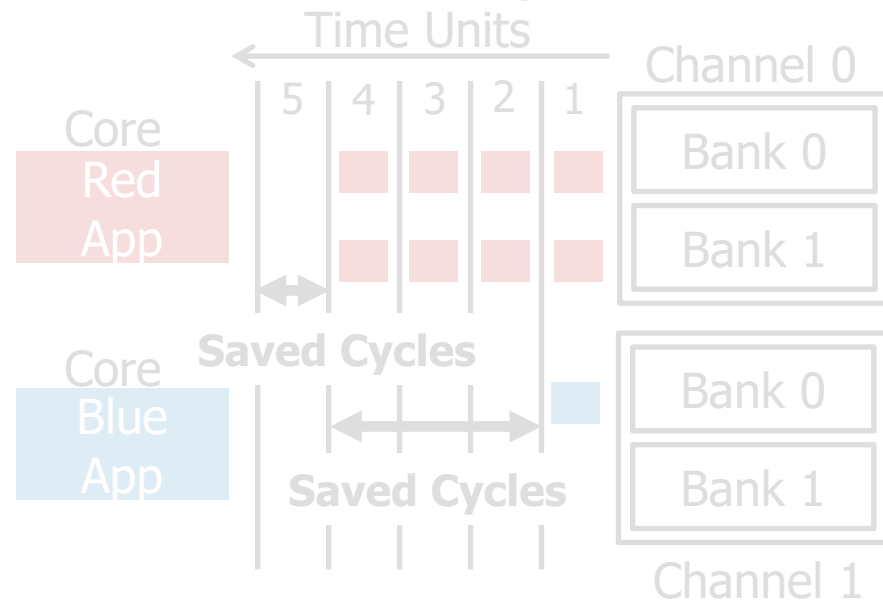
- Separate **low and high memory-intensity applications**
- Separate **low and high row-buffer locality applications**

Key Insight 1: Separate by Memory Intensity

High memory-intensity applications interfere with low memory-intensity applications in shared memory channels



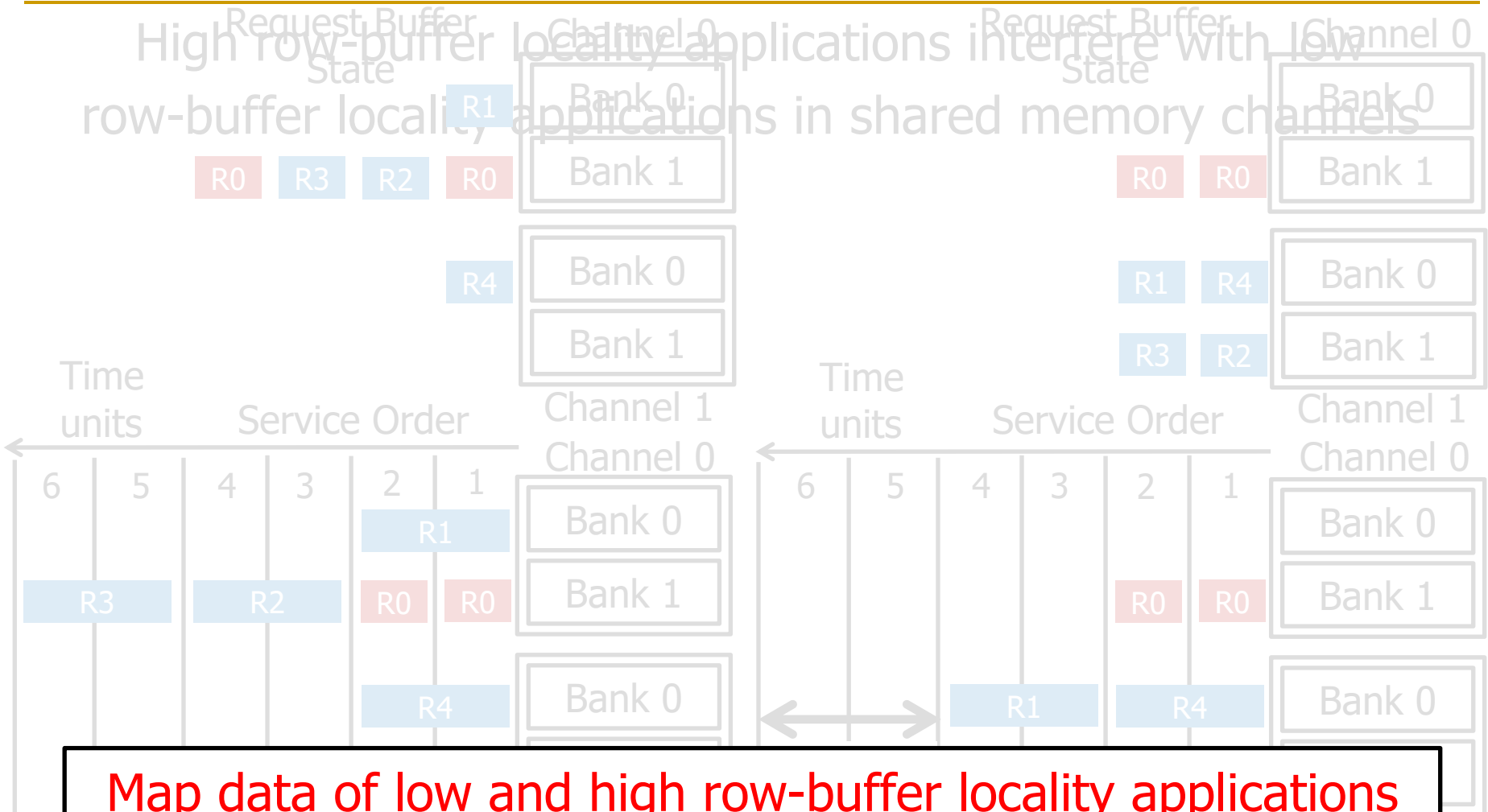
Conventional Page Mapping



Channel Partitioning

Map data of low and high memory-intensity applications to different channels

Key Insight 2: Separate by Row-Buffer Locality



Map data of low and high row-buffer locality applications to different channels

Mechanisms (in some detail)

Memory Channel Partitioning (MCP) Mechanism

Hardware

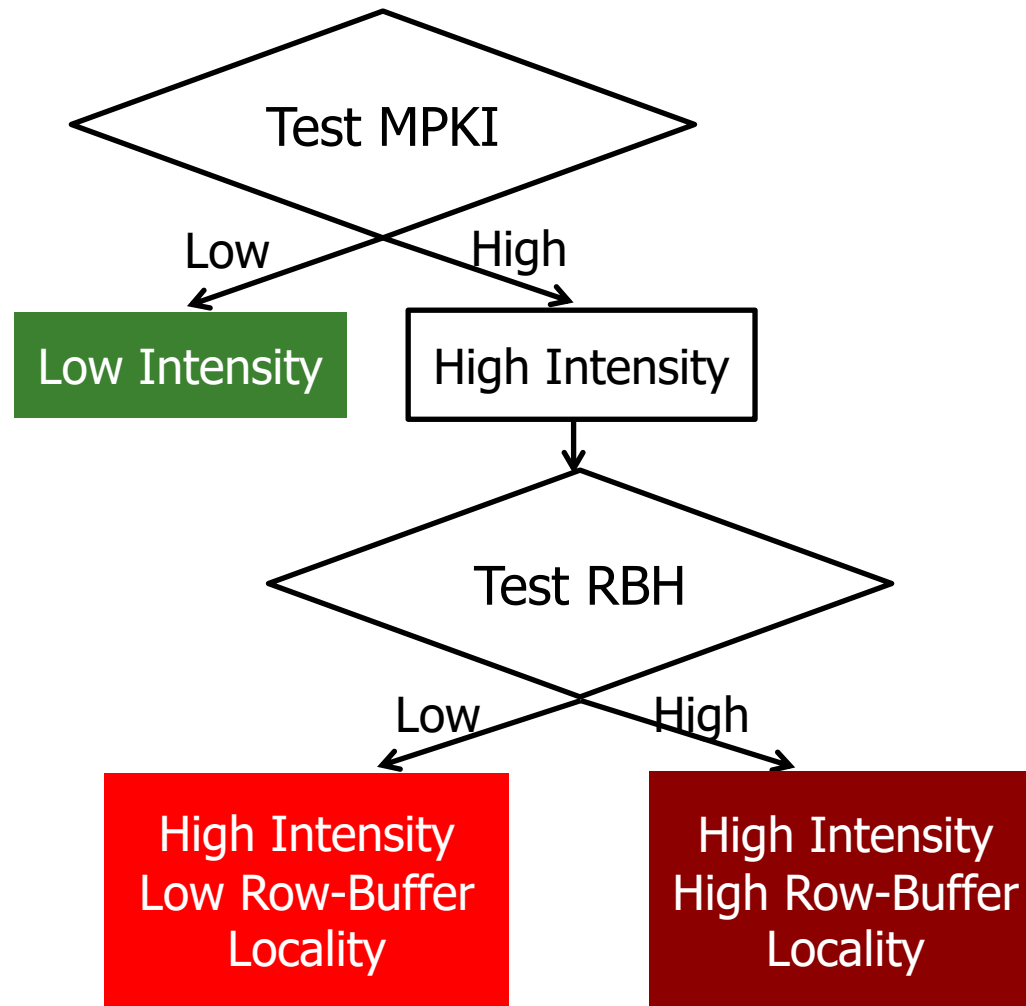
1. Profile applications
2. Classify applications into groups
3. Partition channels between application groups
4. Assign a preferred channel to each application
5. Allocate application pages to preferred channel

**System
Software**

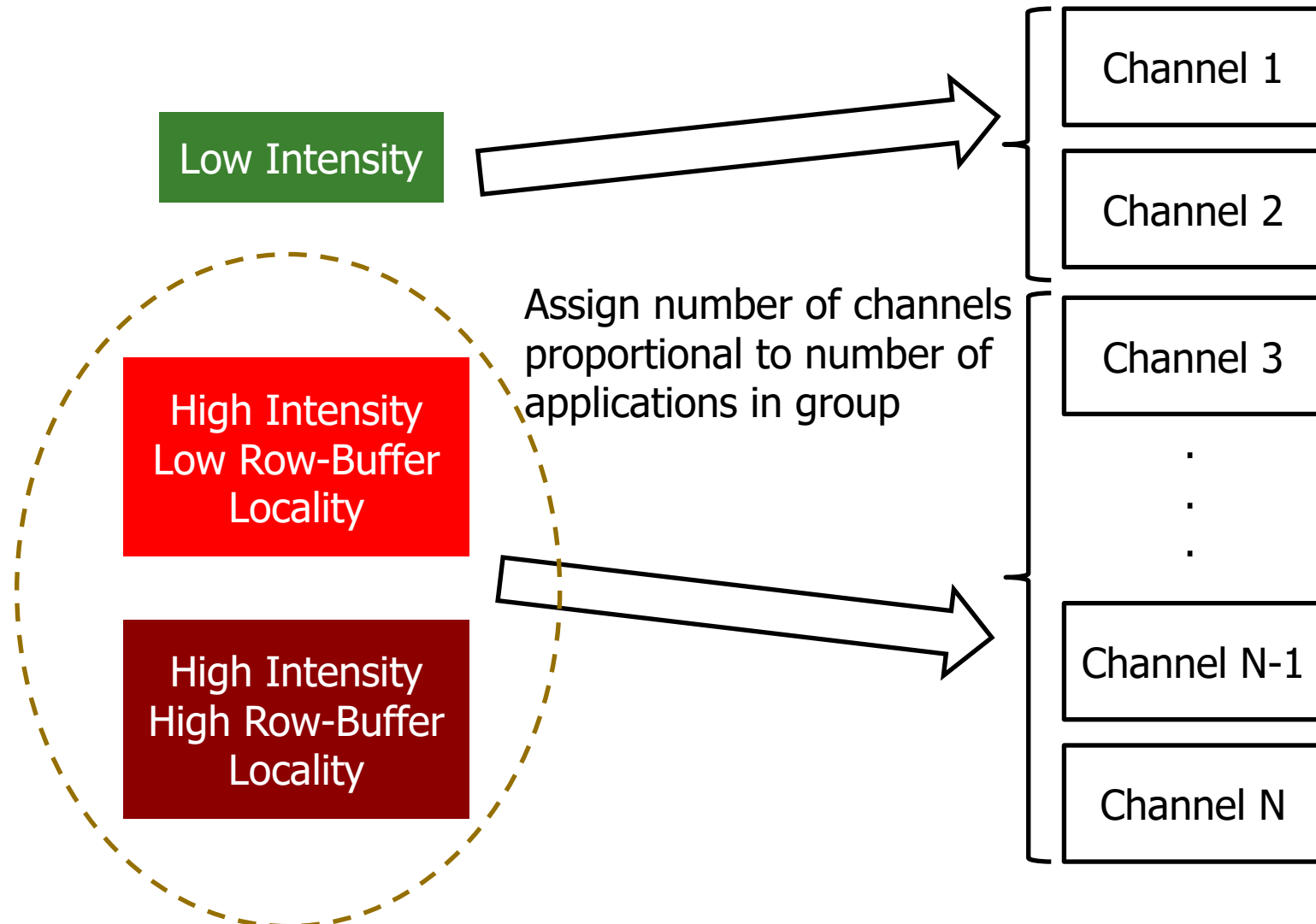
1. Profile Applications

- Hardware counters collect application memory access characteristics
- Memory access characteristics
 - **Memory intensity:**
 - Last level cache **Misses Per Kilo Instruction (MPKI)**
 - **Row-buffer locality:**
 - Row-buffer Hit Rate (RBH)** - percentage of accesses that hit in the row buffer

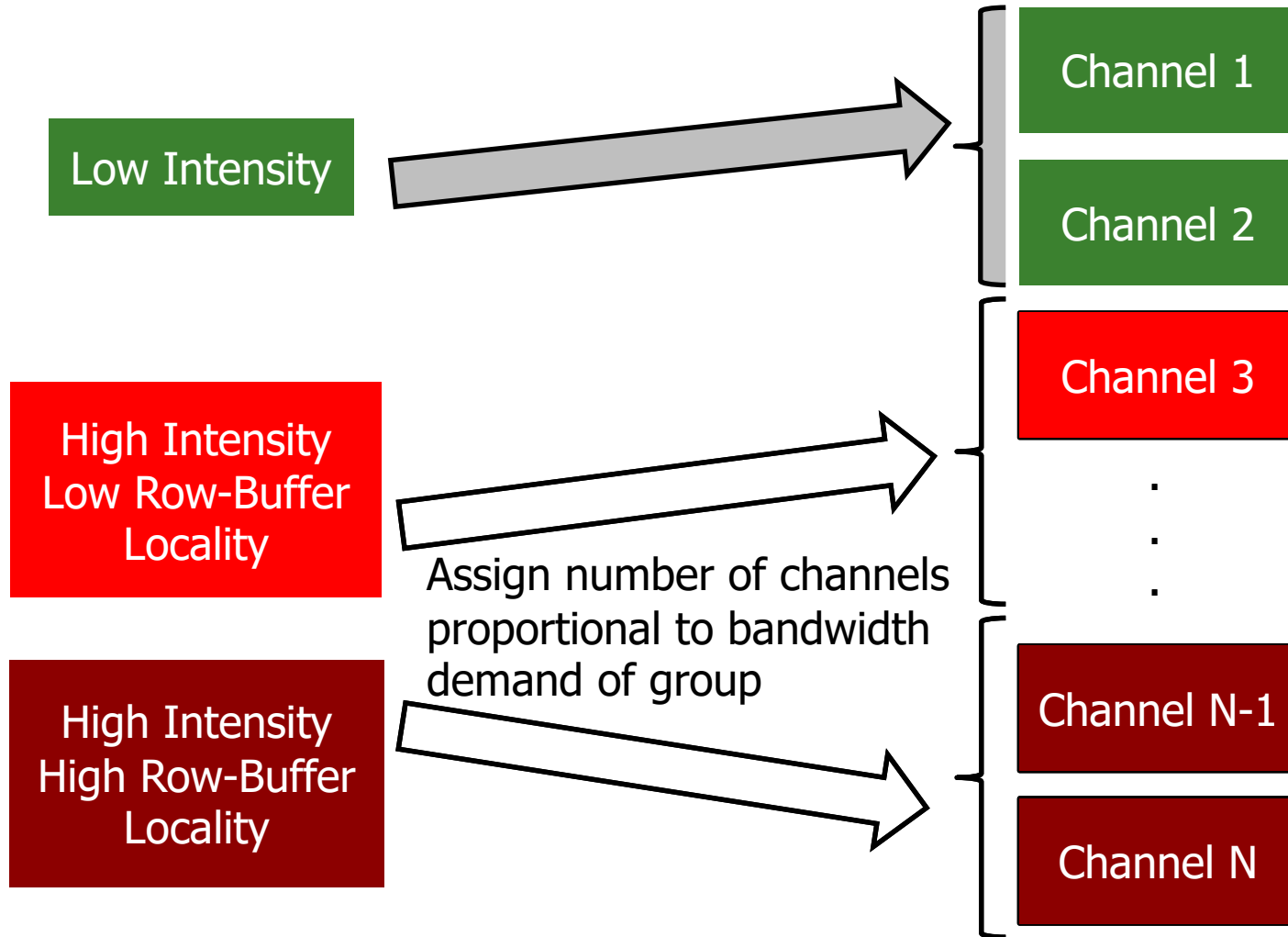
2. Classify Applications



3. Partition Channels Among Groups: Step 1

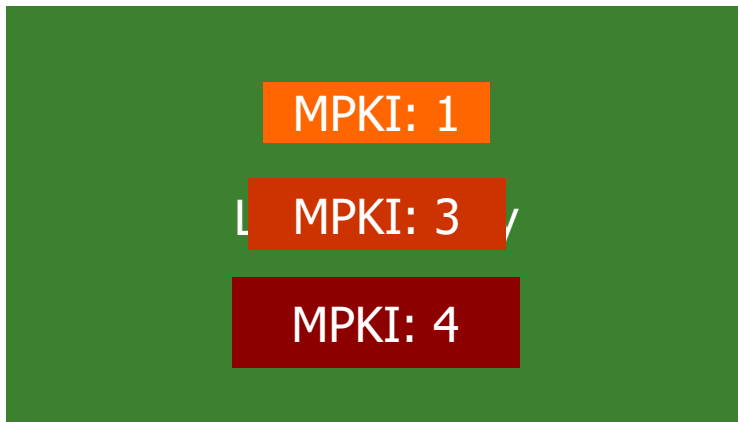


3. Partition Channels Among Groups: Step 2



4. Assign Preferred Channel to Application

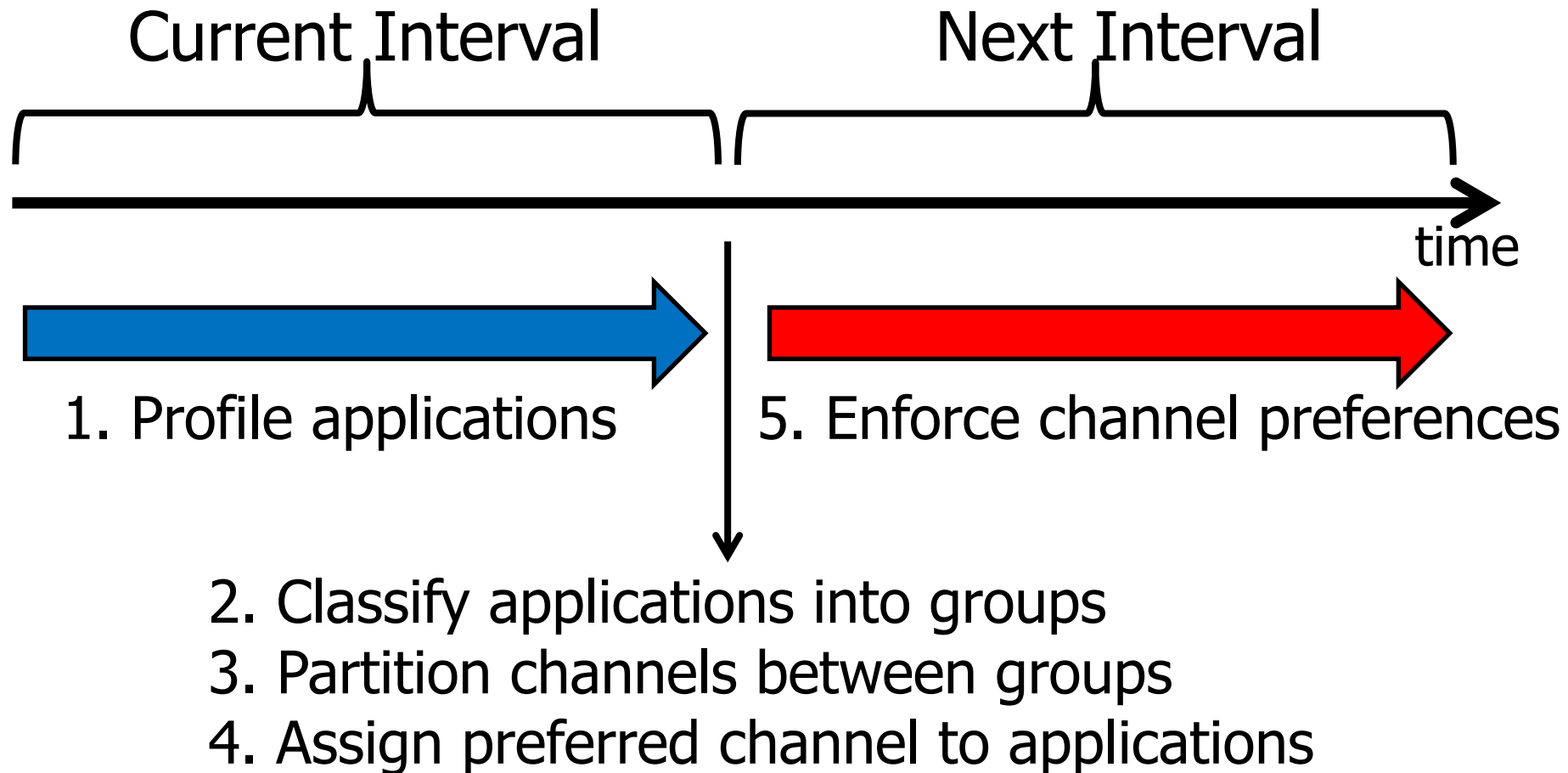
- Assign **each application a preferred channel** from its group's allocated channels
- Distribute applications to channels such that **group's bandwidth demand is balanced** across its channels



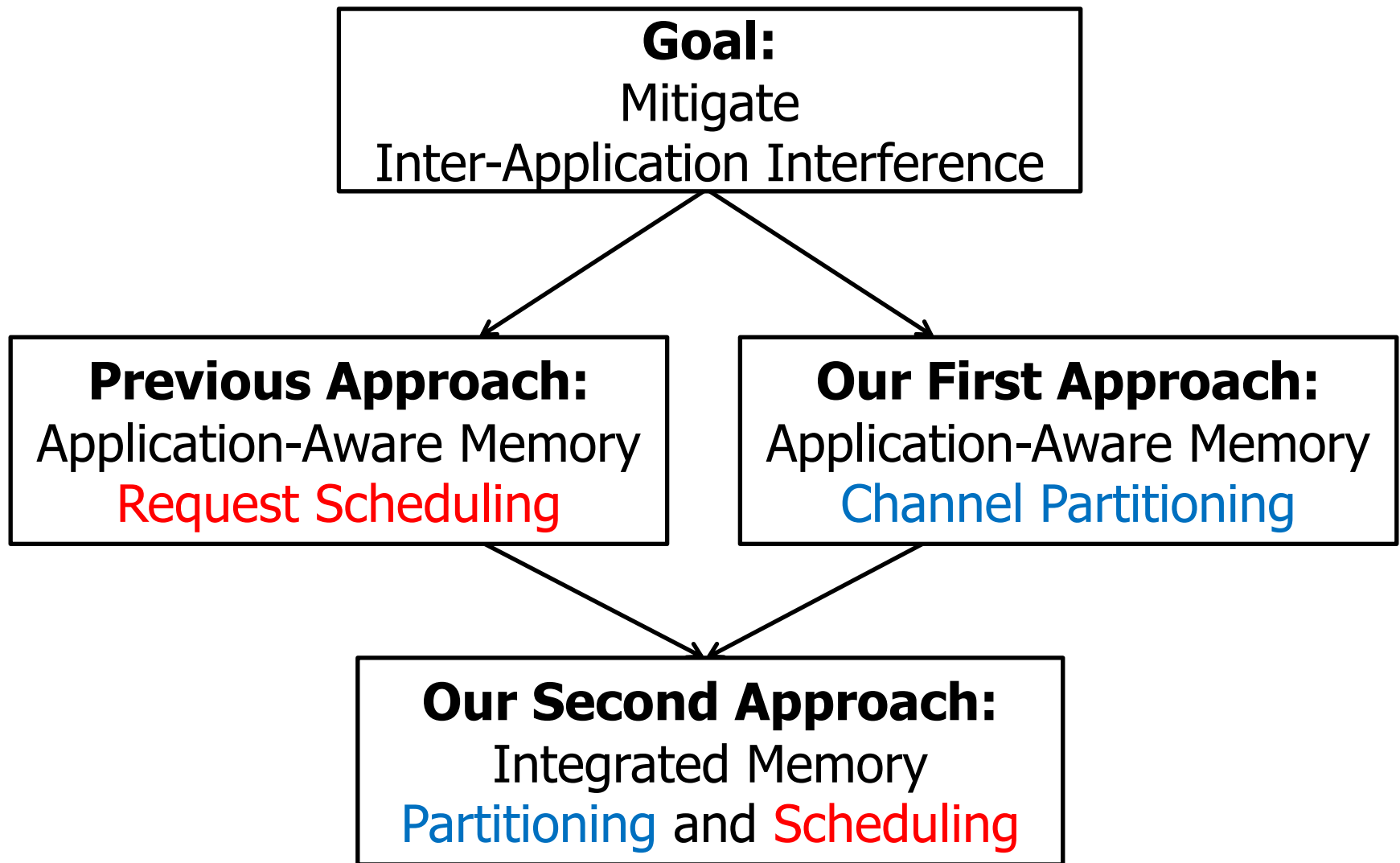
5. Allocate Page to Preferred Channel

- **Enforce channel preferences**
computed in the previous step
- On a page fault, the operating system
 - allocates page to preferred channel **if free page available** in preferred channel
 - **if free page not available**, replacement policy tries to allocate page to preferred channel
 - **if it fails**, allocate page to another channel

Interval Based Operation



Integrating Partitioning and Scheduling



Observations

- Applications with very low memory-intensity rarely access memory
 - Dedicating channels to them results in precious memory bandwidth waste
- They have the most potential to keep their cores busy
 - We would really like to prioritize them
- They interfere minimally with other applications
 - Prioritizing them does not hurt others

Integrated Memory Partitioning and Scheduling (IMPS)

- Always prioritize very low memory-intensity applications in the memory scheduler
- Use memory channel partitioning to mitigate interference between other applications

Key Results:

Methodology and Evaluation

Hardware Cost

- **Memory Channel Partitioning (MCP)**
 - ❑ Only profiling counters in hardware
 - ❑ No modifications to memory scheduling logic
 - ❑ 1.5 KB storage cost for a 24-core, 4-channel system
- **Integrated Memory Partitioning and Scheduling (IMPS)**
 - ❑ A single bit per request
 - ❑ Scheduler prioritizes based on this single bit

Methodology

■ Simulation Model

- 24 cores, 4 channels, 4 banks/channel
- Core Model
 - Out-of-order, 128-entry instruction window
 - 512 KB L2 cache/core
- Memory Model – DDR2

■ Workloads

- 240 SPEC CPU 2006 multiprogrammed workloads (categorized based on memory intensity)

■ Metrics

- System Performance $Weighted\ Speedup = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}}$

Previous Work on Memory Scheduling

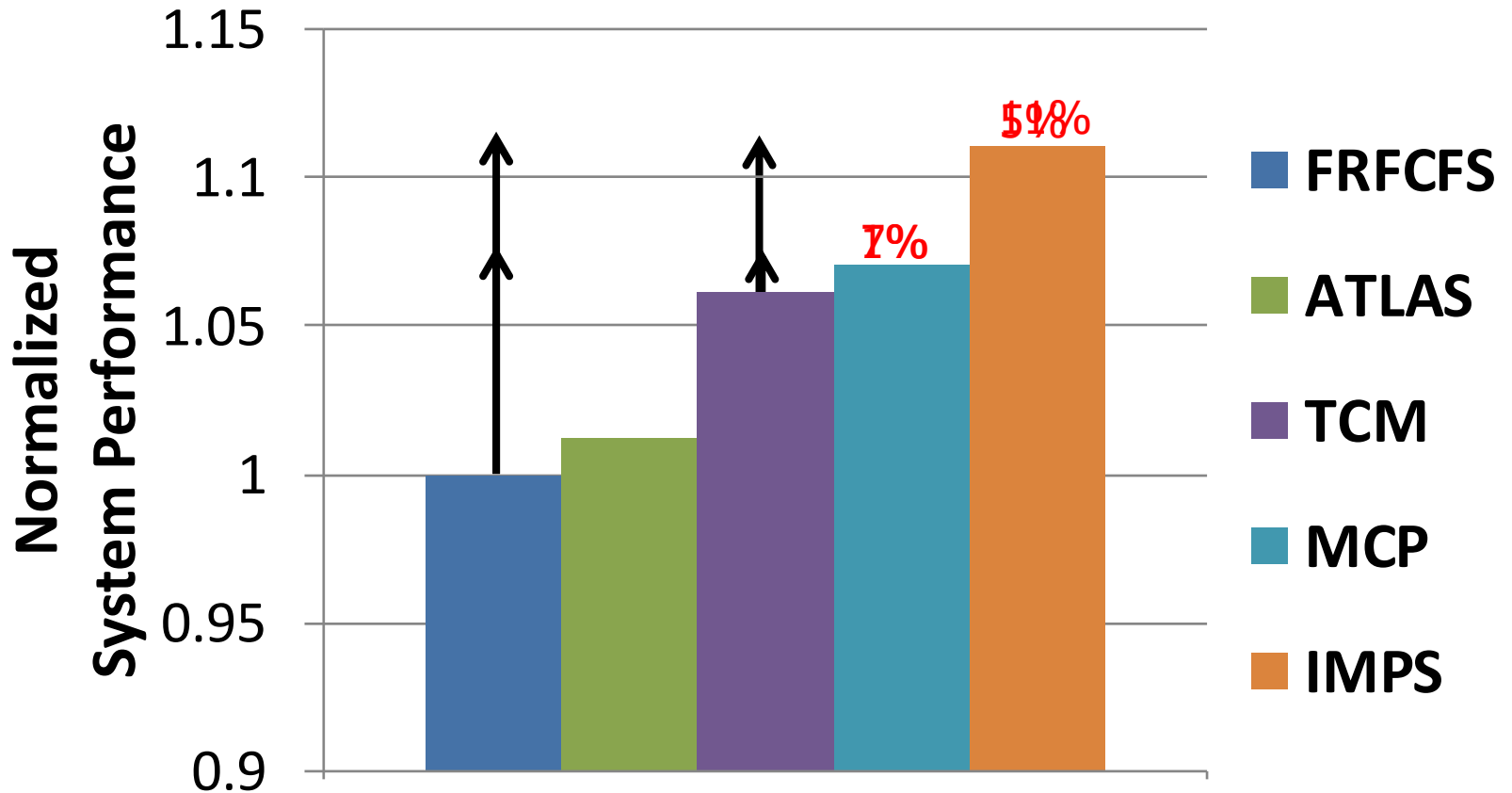
- **FR-FCFS** [Zuravleff et al., US Patent 1997, Rixner et al., ISCA 2000]
 - Prioritizes row-buffer hits and older requests
 - Application-unaware

- **ATLAS** [Kim et al., HPCA 2010]
 - Prioritizes applications with low memory-intensity

- **TCM** [Kim et al., MICRO 2010]
 - Always prioritizes low memory-intensity applications
 - Shuffles request priorities of high memory-intensity applications

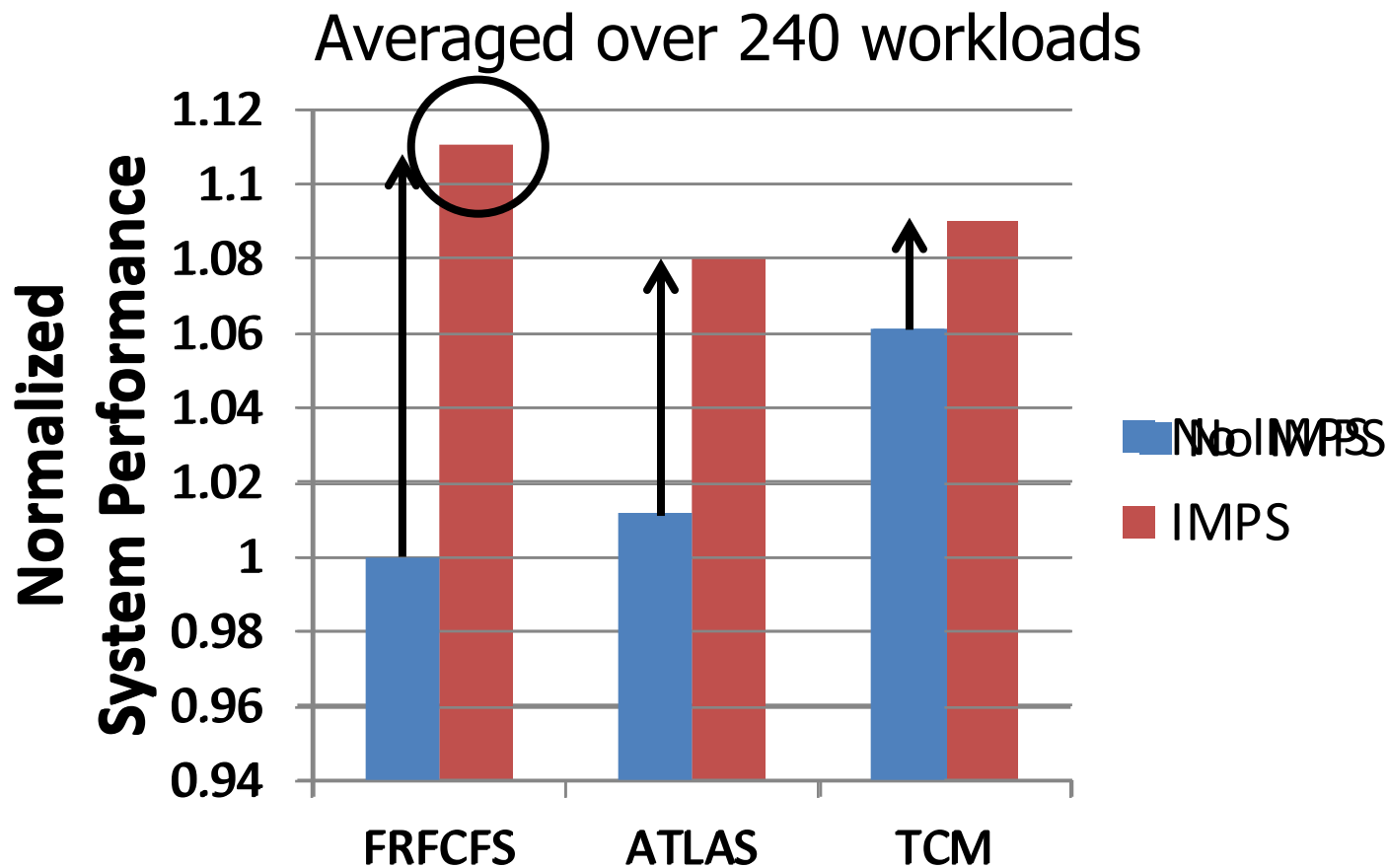
Comparison to Previous Scheduling Policies

Averaged over 240 workloads



Better system performance than the best previous scheduler
Significant performance improvement over baseline FRFCFS
at lower hardware cost

Interaction with Memory Scheduling



IMPS improves performance regardless of scheduling policy
Highest improvement over FRFCFS as IMPS designed for FRFCFS

Summary

Summary

- Uncontrolled inter-application interference in main memory degrades system performance
- Application-aware memory channel partitioning (MCP)
 - Separates the data of badly-interfering applications to different channels, eliminating interference
- Integrated memory partitioning and scheduling (IMPS)
 - Prioritizes very low memory-intensity applications in scheduler
 - Handles other applications' interference by partitioning
- MCP/IMPS provide better performance than application-aware memory request scheduling at lower hardware cost

Strengths

Strengths of the Paper

- Novel solution to a key problem in multi-core systems, memory interference; the importance of problem will increase over time
- Keeps the memory scheduling hardware simple
- Combines multiple interference reduction techniques
- Can provide performance isolation across applications mapped to different channels
- General idea of partitioning can be extended to smaller granularities in the memory hierarchy: banks, subarrays, etc.

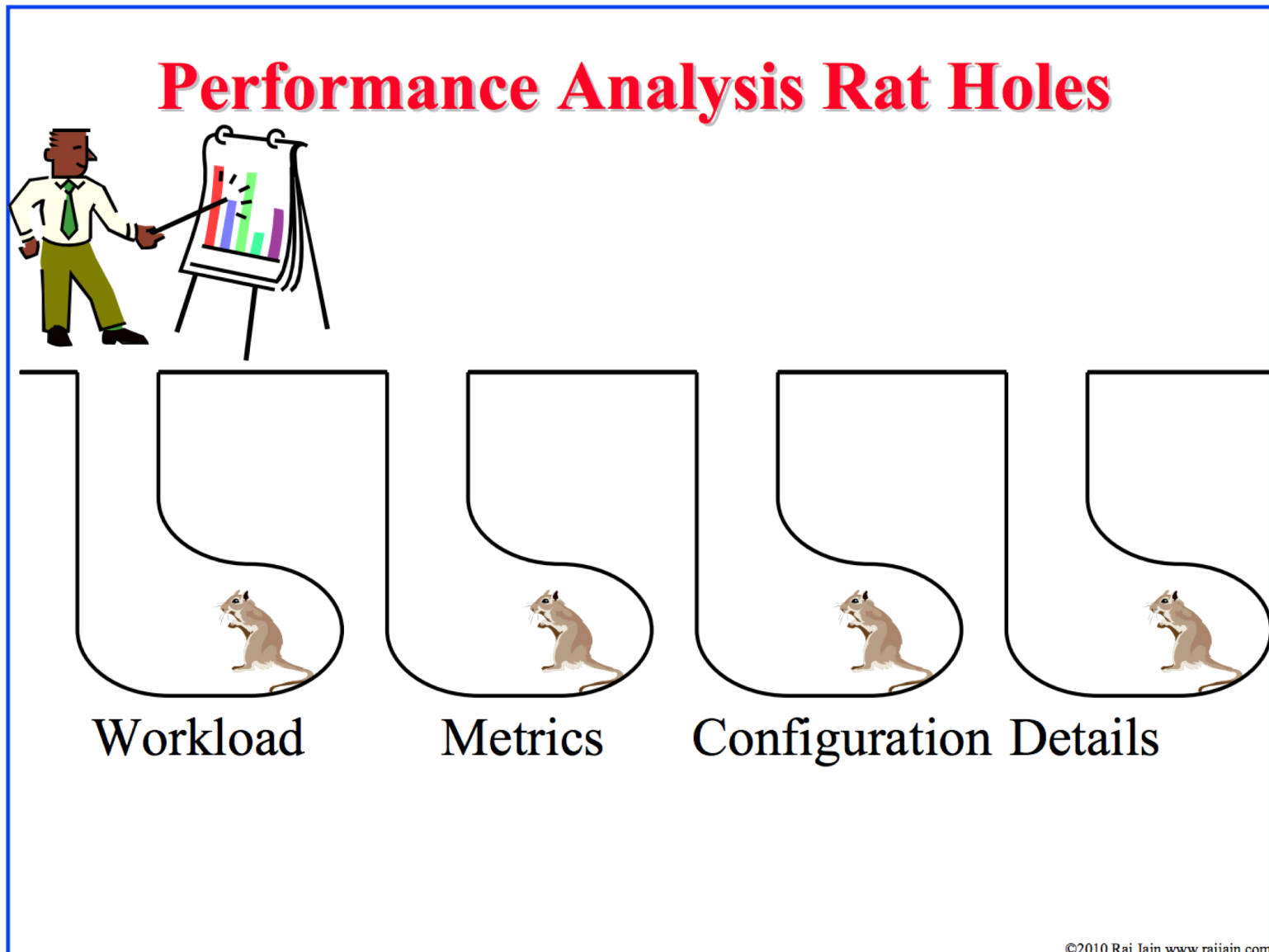
- Well-written paper
- Thorough simulation-based evaluation

Weaknesses

Weaknesses/Limitations of the Paper

- Mechanism may not work effectively if workload changes behavior after profiling
- Overhead of moving pages between channels restricts mechanism's benefits
- Small number of memory channels reduces the scope of partitioning
- Load imbalance across channels can reduce performance
 - The paper addresses this and compares to another mechanism
- Software-hardware cooperative solution might not always be easy to adopt
- Evaluation is done solely in simulation
- Evaluation does not consider multi-chip systems
- Are these the best workloads to evaluate?

Recall: Try to Avoid Rat Holes



Thoughts and Ideas

Extensions

- Can this idea be extended to different granularities in memory?
 - Partition banks, subarrays, mats across workloads
- Can this idea be extended to provide performance predictability and performance isolation? How?
- How can MCP be combined effectively with other interference reduction techniques?
 - E.g., source throttling methods [Ebrahimi+, ASPLOS 2010]
 - E.g., thread scheduling methods
- Can this idea be evaluated on a real system? How?

Takeaways

Key Takeaways

- A novel method to reduce memory interference
- Simple and effective
- Hardware/software cooperative
- Good potential for work building on it to extend it
 - To different structures
 - To different metrics
 - Multiple works have already built on the paper (see bank partitioning works in PACT 2012, HPCA 2012)
- Easy to read and understand paper

Open Discussion

Discussion Starters

- Thoughts on the previous ideas?
- How practical is this?
- Will the problem become bigger and more important over time?
- Will the solution become more important over time?
- Are other solutions better?
- Is this solution clearly advantageous in some cases?

Application-Aware Memory Channel Partitioning

Sai Prashanth Muralidhara § Lavanya Subramanian †

Onur Mutlu † Mahmut Kandemir §

Thomas Moscibroda ‡

§ Pennsylvania State University † Carnegie Mellon University

‡ Microsoft Research

SAFARI Carnegie Mellon