Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM

Decoupled Direct Memory Access

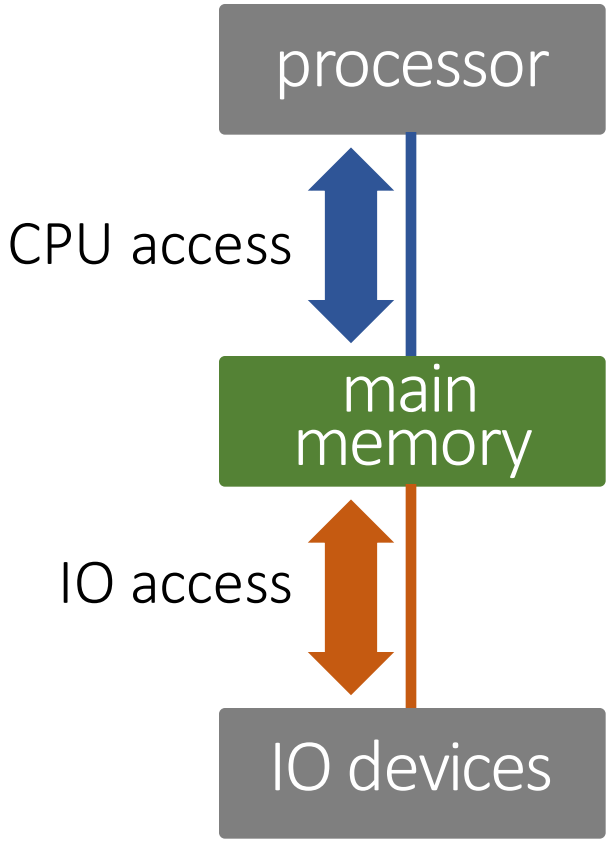
Donghyuk Lee

Lavanya Subramanian, Rachata Ausavarungnirun,
Jongmoo Choi, Onur Mutlu

SAFARI

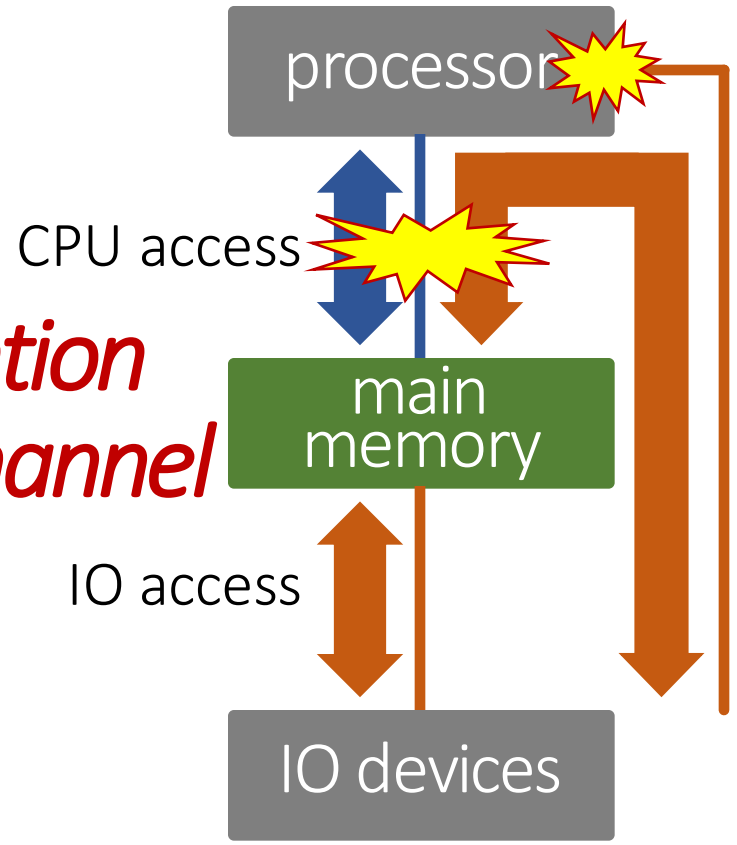
Carnegie Mellon

Logical System Organization



Main memory connects processor and IO devices as an ***intermediate layer***

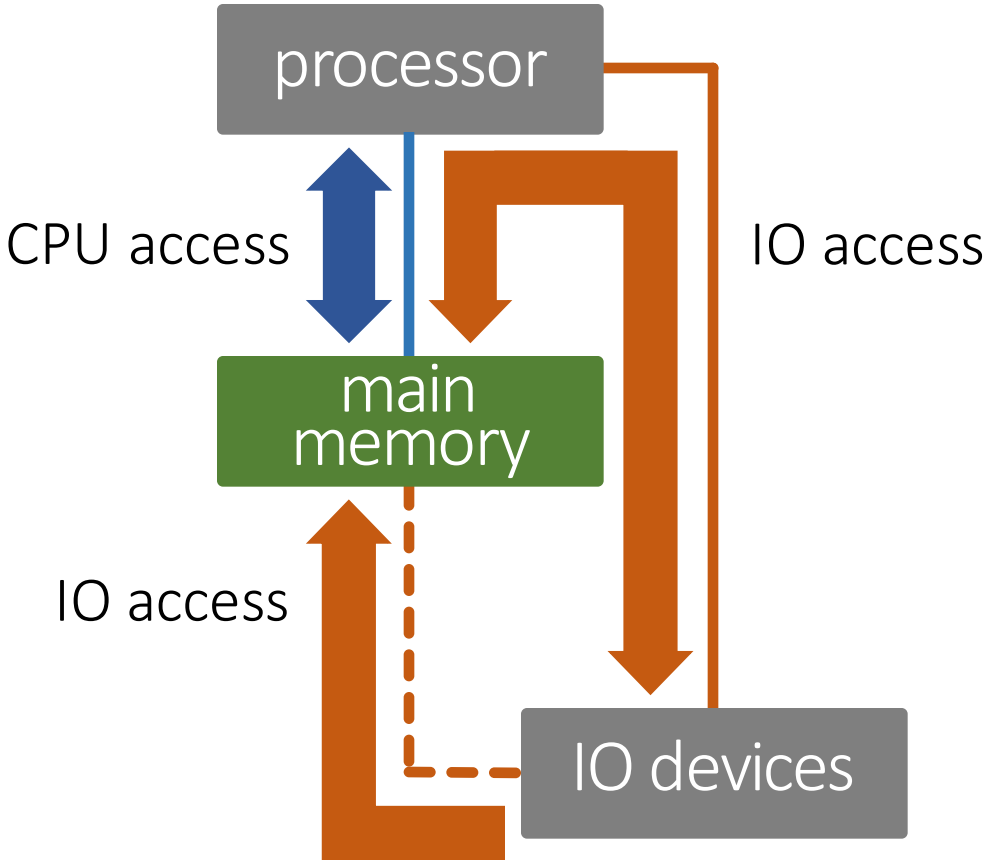
Physical System Implementation



High Pin Cost in Processor

High Contention in Memory Channel

Our Approach



Enabling IO channel,
decoupled & *isolated* from CPU channel

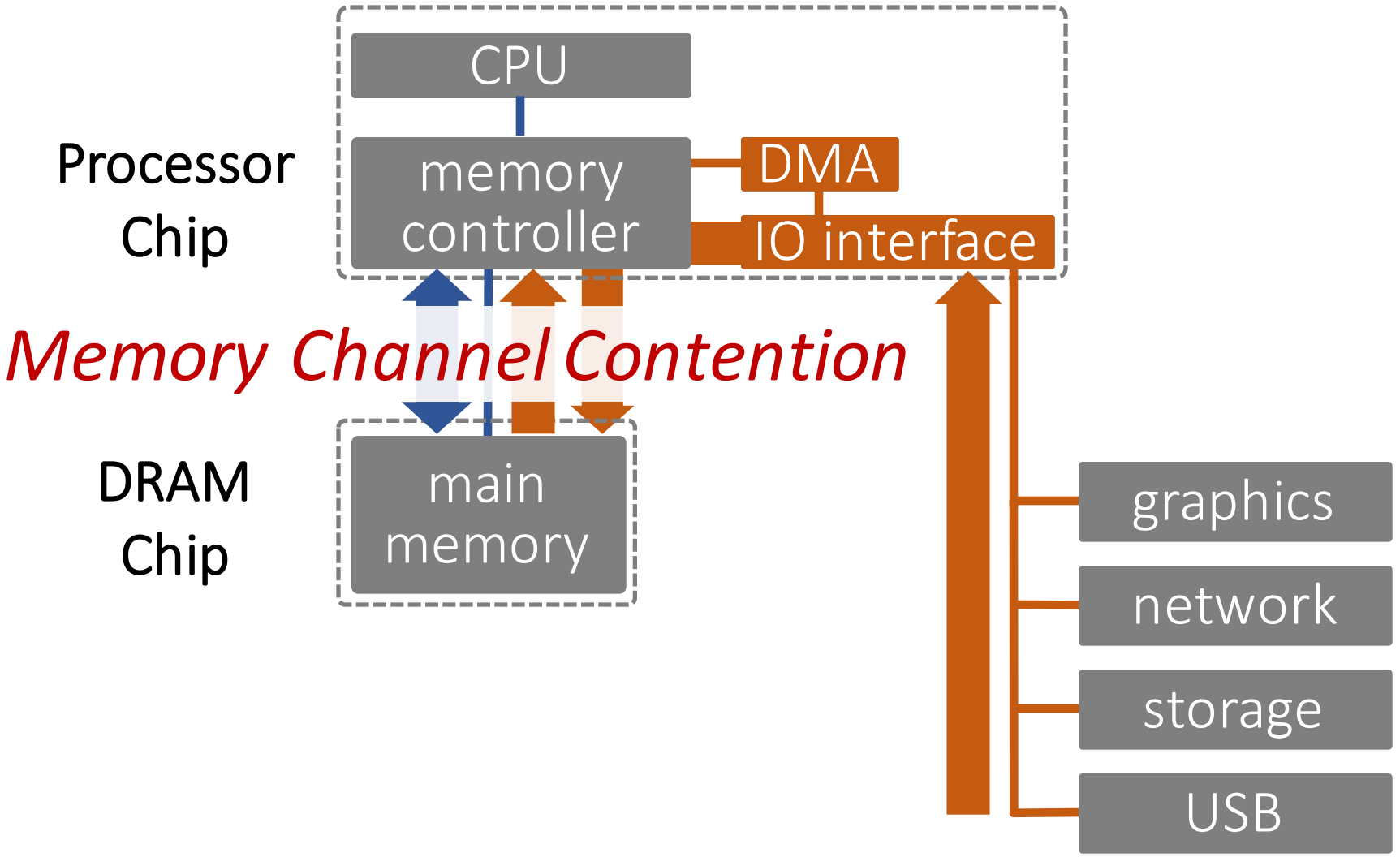
Executive Summary

- **Problem**
  - CPU and IO accesses contend for the shared memory channel
- **Our Approach:** *Decoupled Direct Memory Access (DDMA)*
  - Design new DRAM architecture with two independent data ports
    → *Dual-Data-Port DRAM*
  - Connect one port to CPU and the other port to IO devices
    → *Decouple CPU and IO accesses*
- **Application**
  - Communication between compute units (e.g., CPU – GPU)
  - In-memory communication (e.g., bulk in-memory copy/init.)
  - Memory-storage communication (e.g., page fault, IO prefetch)
- **Result**
  - Significant *performance improvement* (20% in 2 ch. & 2 rank system)
  - *CPU pin count reduction* (4.5%)

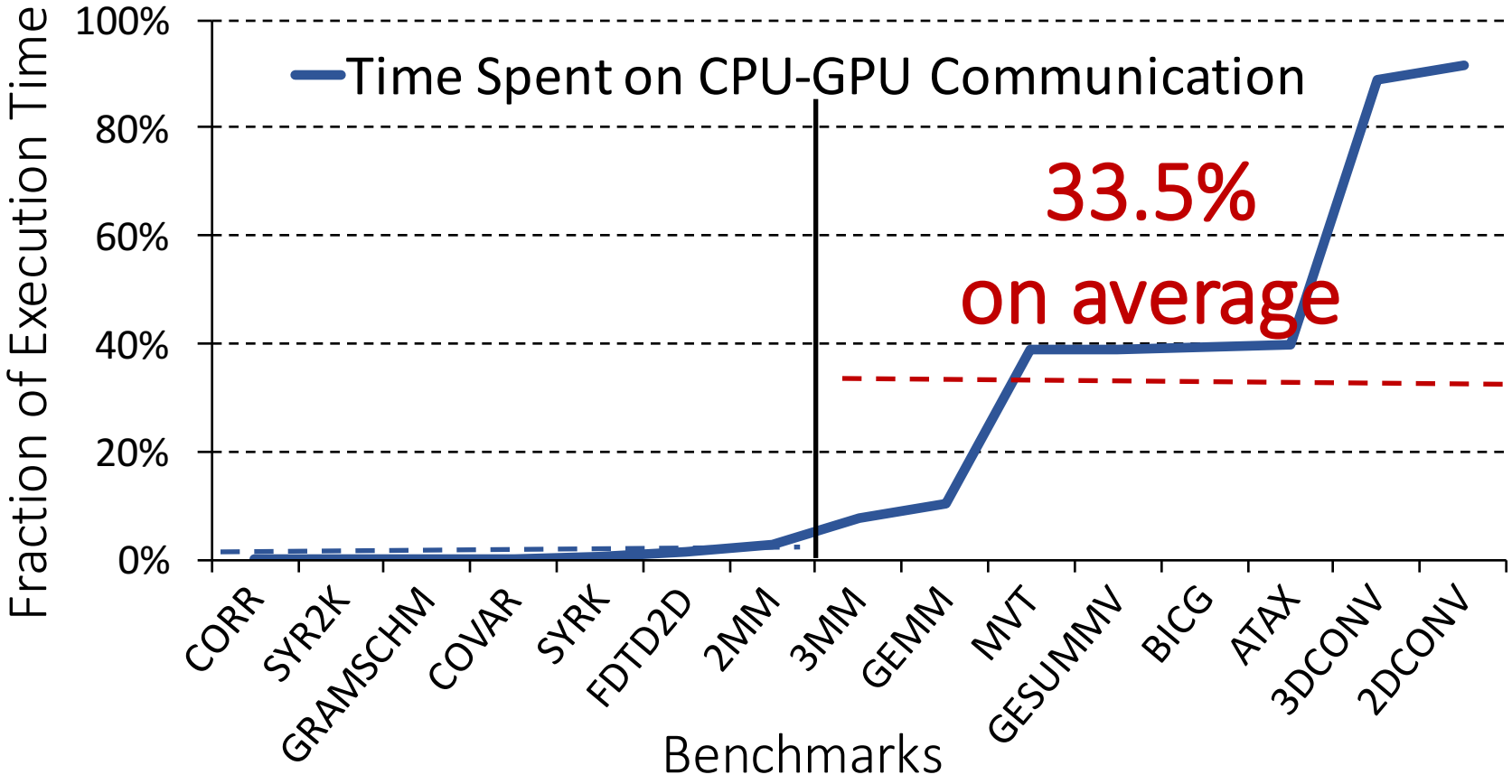
Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Problem 1: Memory Channel Contention

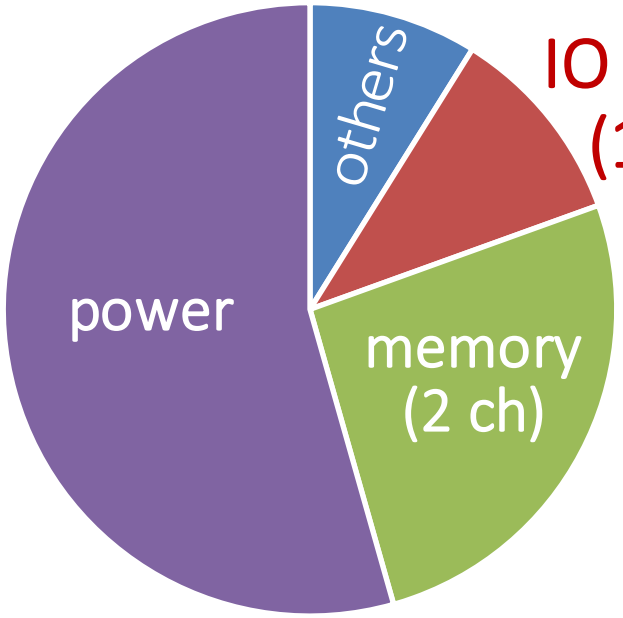


Problem 1: Memory Channel Contention



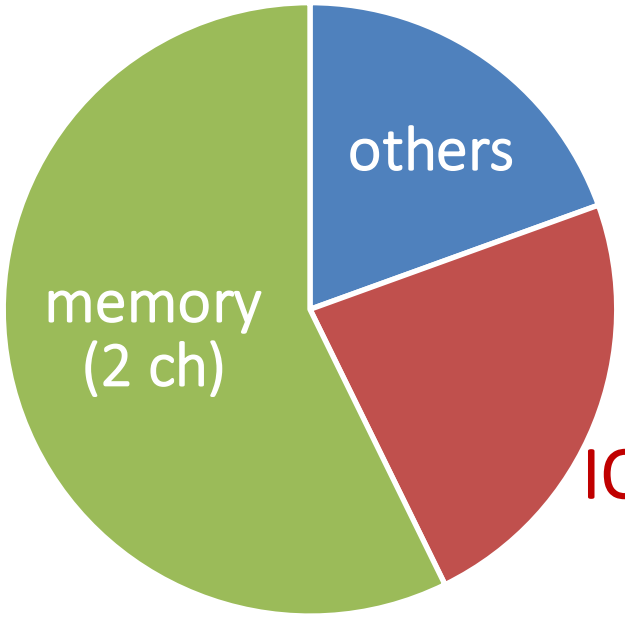
A large fraction of the execution time is spent on IO accesses

Problem 2: High Cost for IO Interfaces



959 pins in total

Processor Pin Count (w/ power pins)



359 pins in total

Processor Pin Count (w/o power pins)

Integrating IO interface on the processor chip leads to ***high area cost***

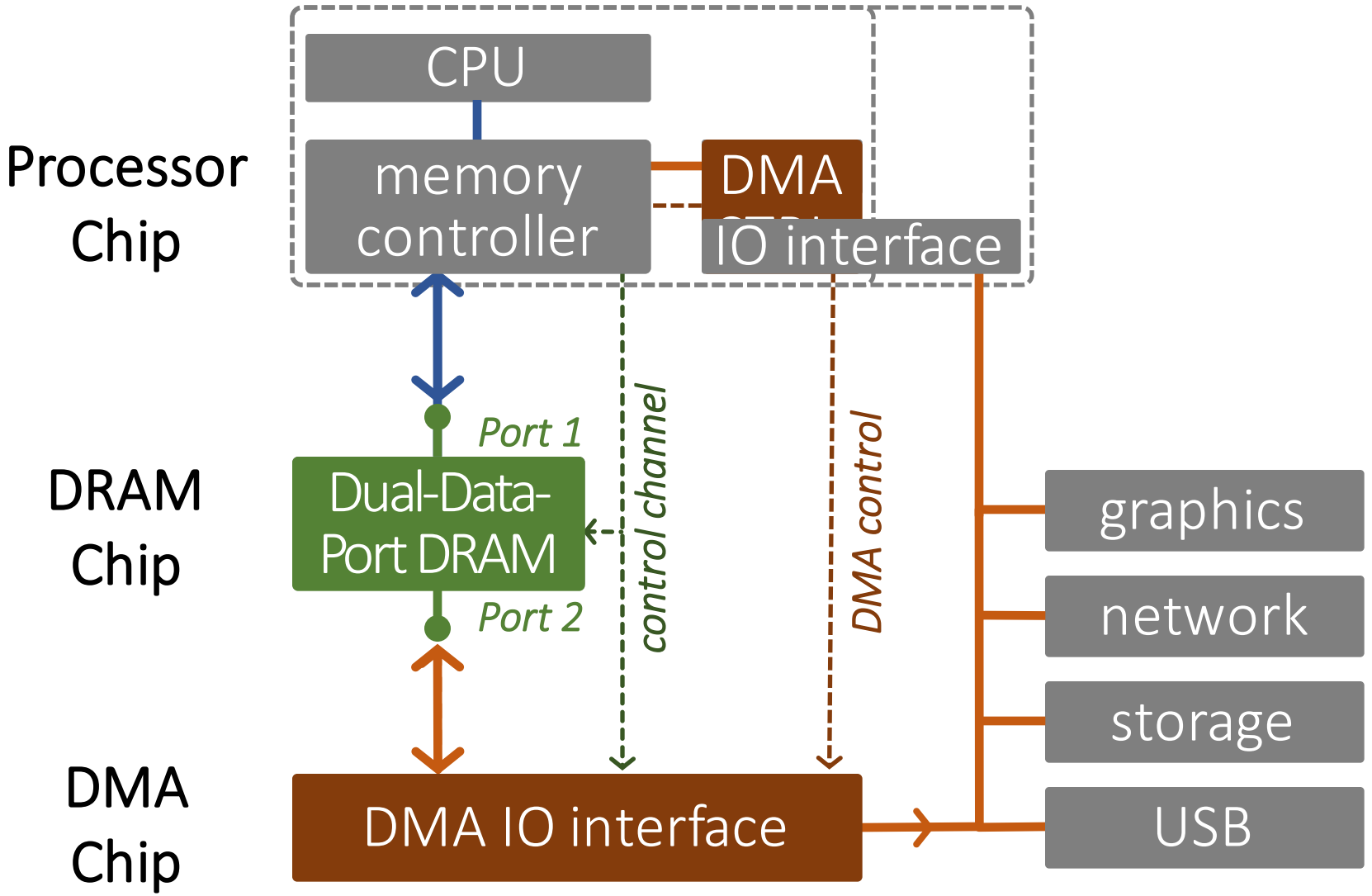
Shared Memory Channel

- Memory channel contention for IO access and CPU access
- High area cost for integrating IO interfaces on processor chip

Outline

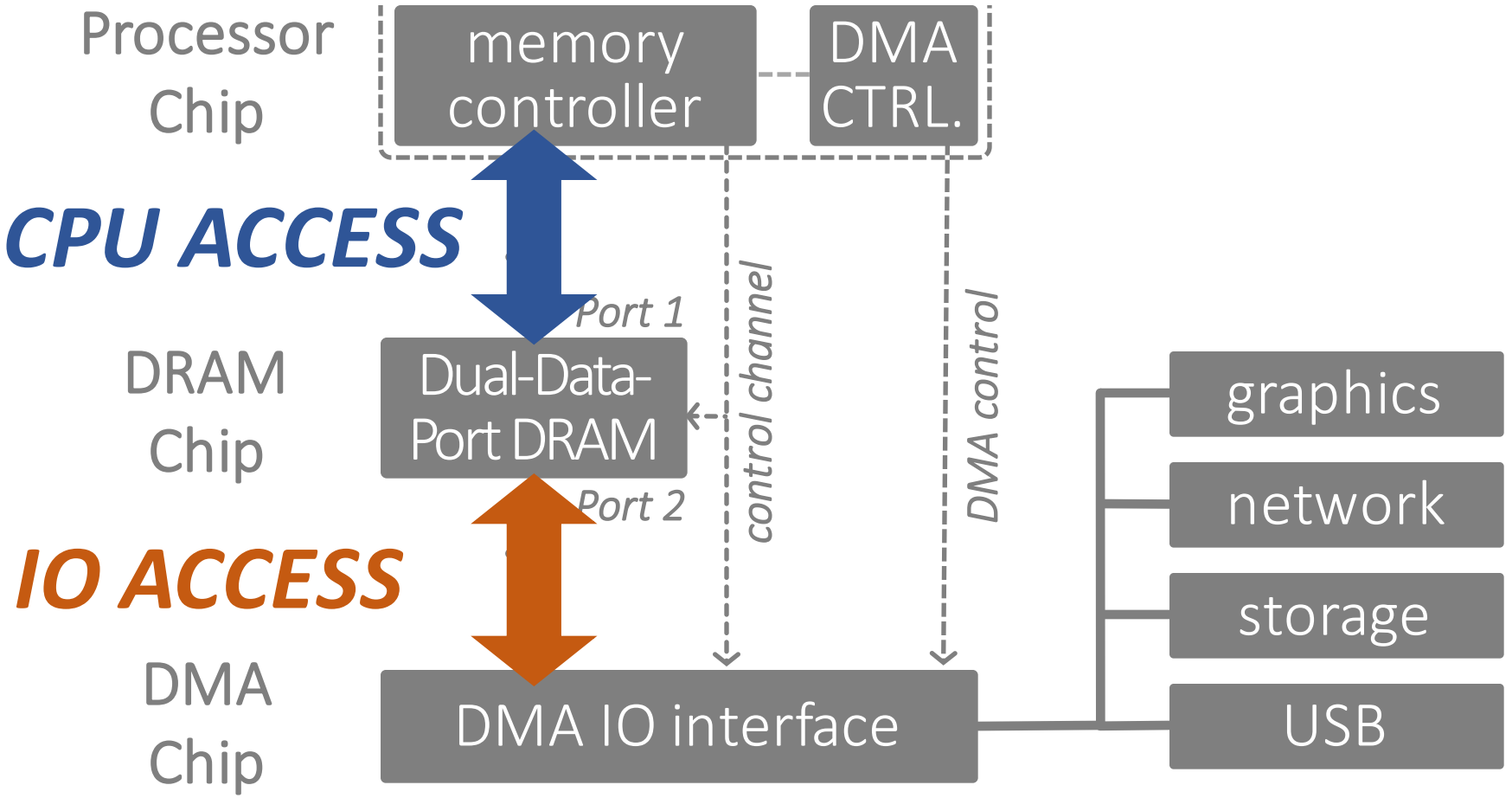
1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Our Approach



Our Approach

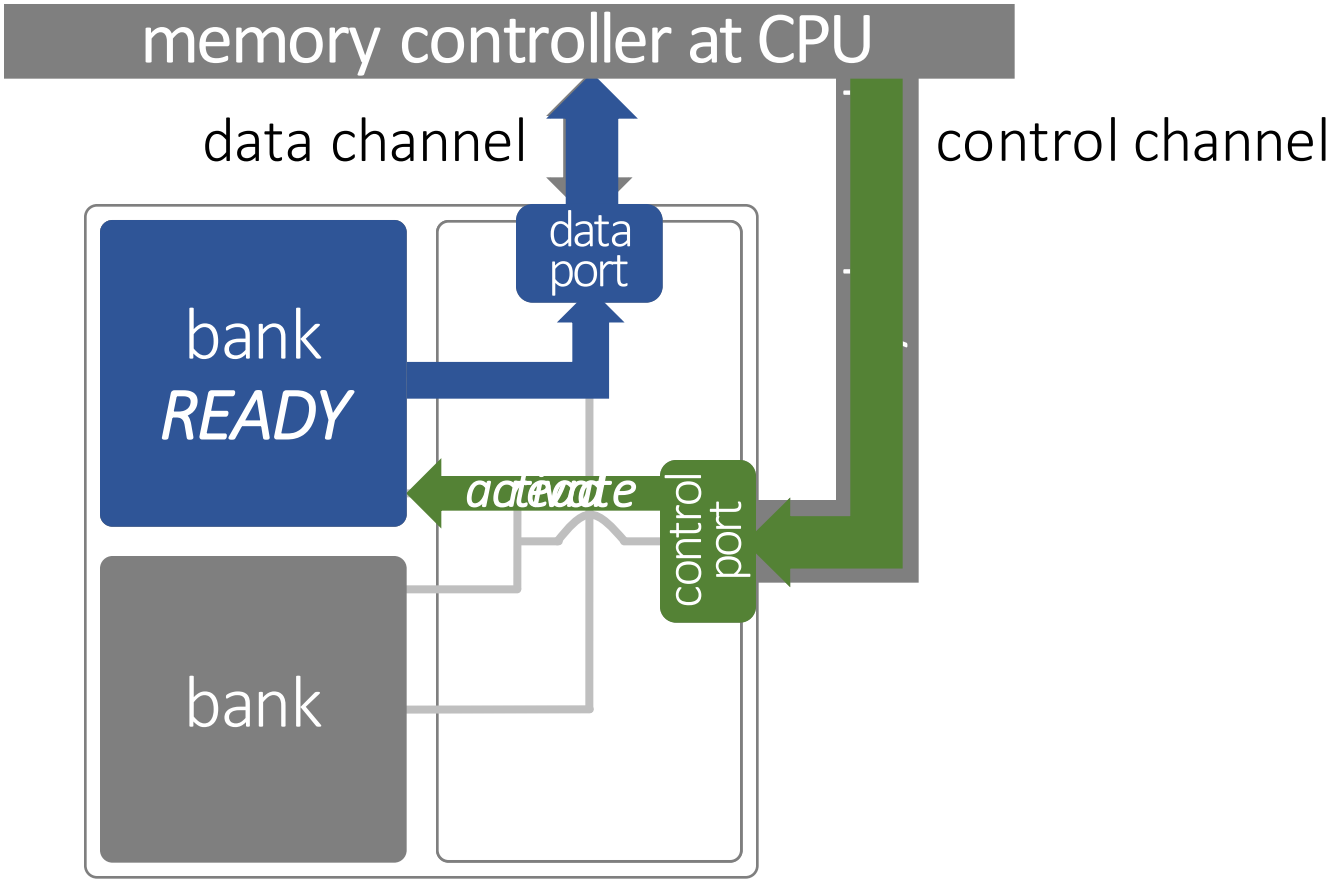
Decoupled Direct Memory Access



Outline

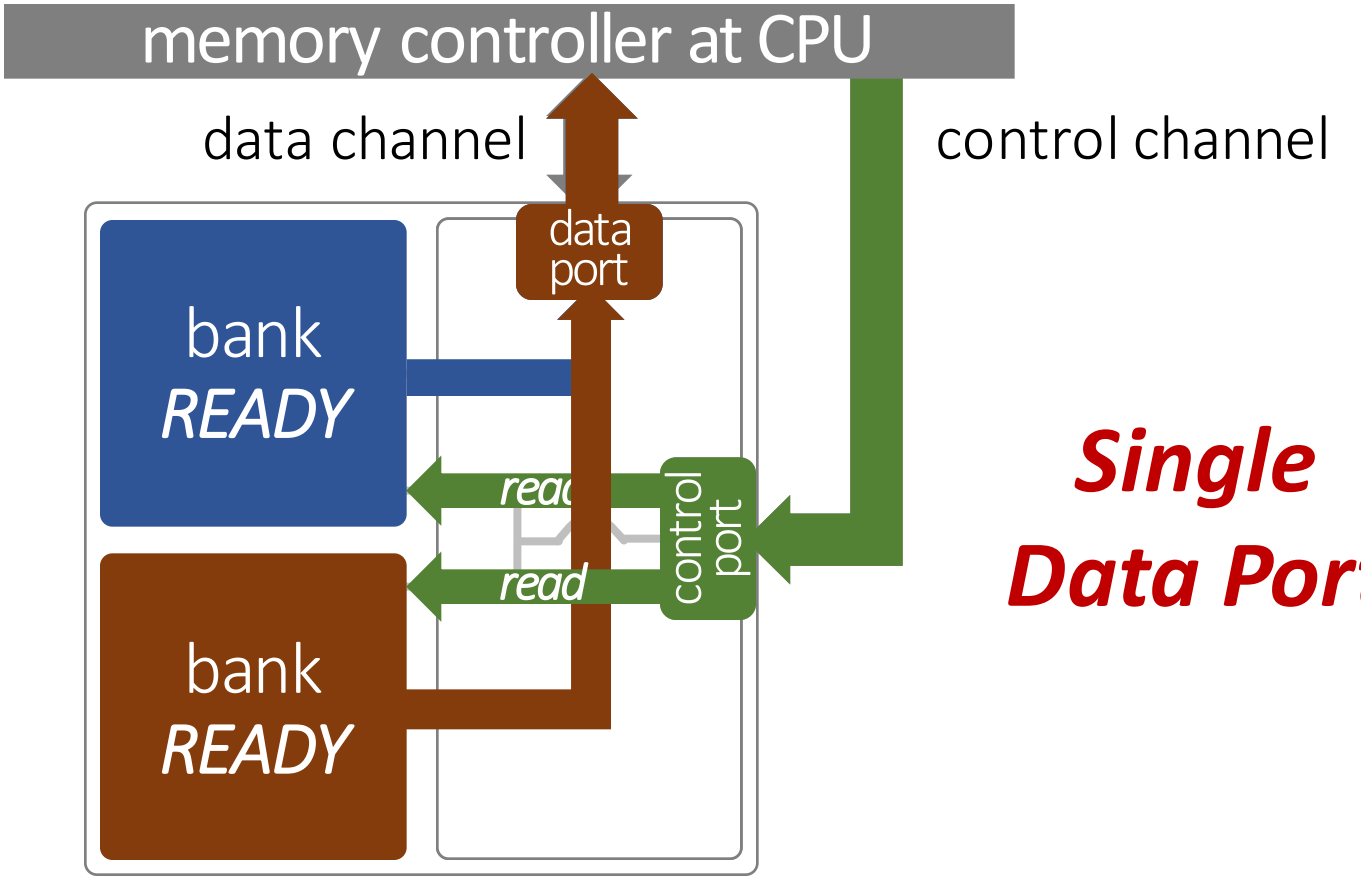
1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Background: DRAM Operation



DRAM peripheral logic: *i) controls banks*, and *ii) transfers data* over memory channel

Problem: Single Data Port



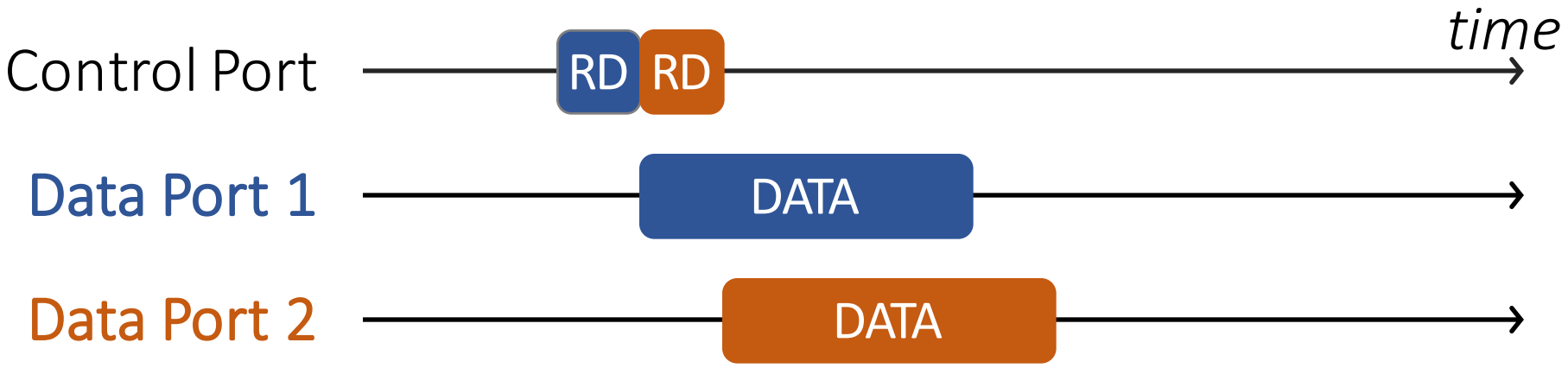
Many Banks

Single Data Port

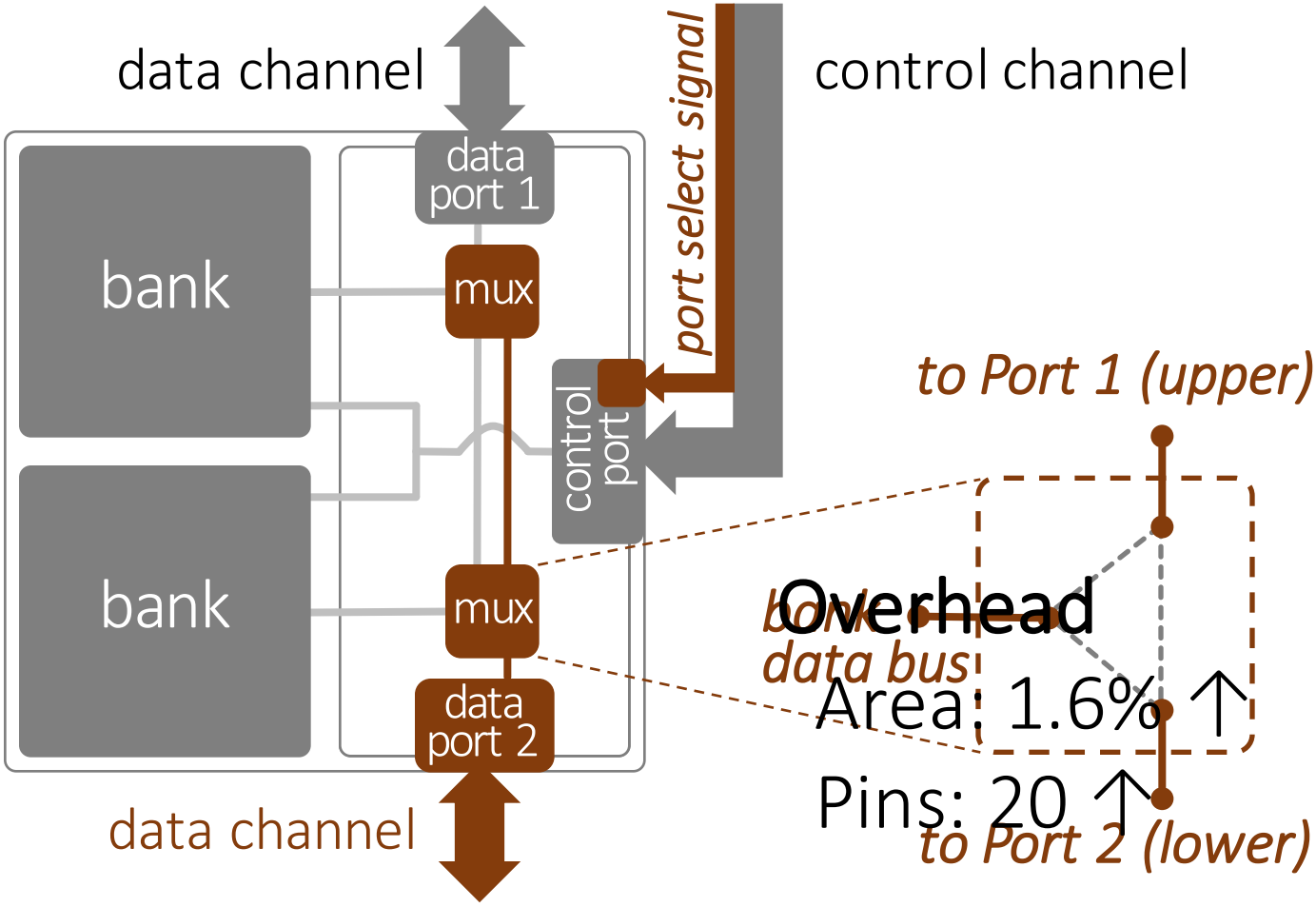
Requests are served *serially* due to ***single data port***

Problem: Single Data Port

What about a DRAM with **two data ports**?

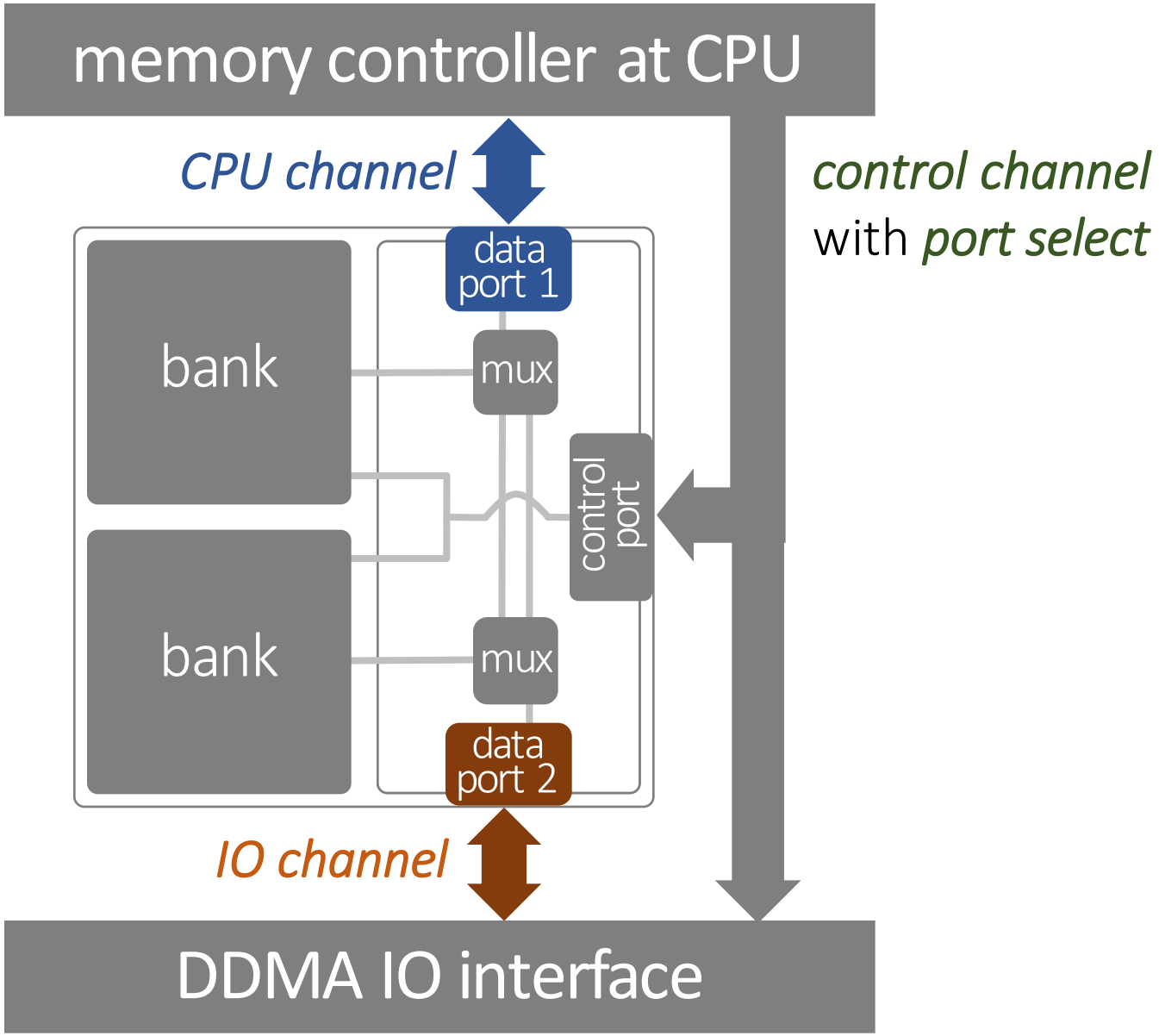


Dual-Data-Port DRAM



twice the bandwidth & *independent data ports with low overhead*

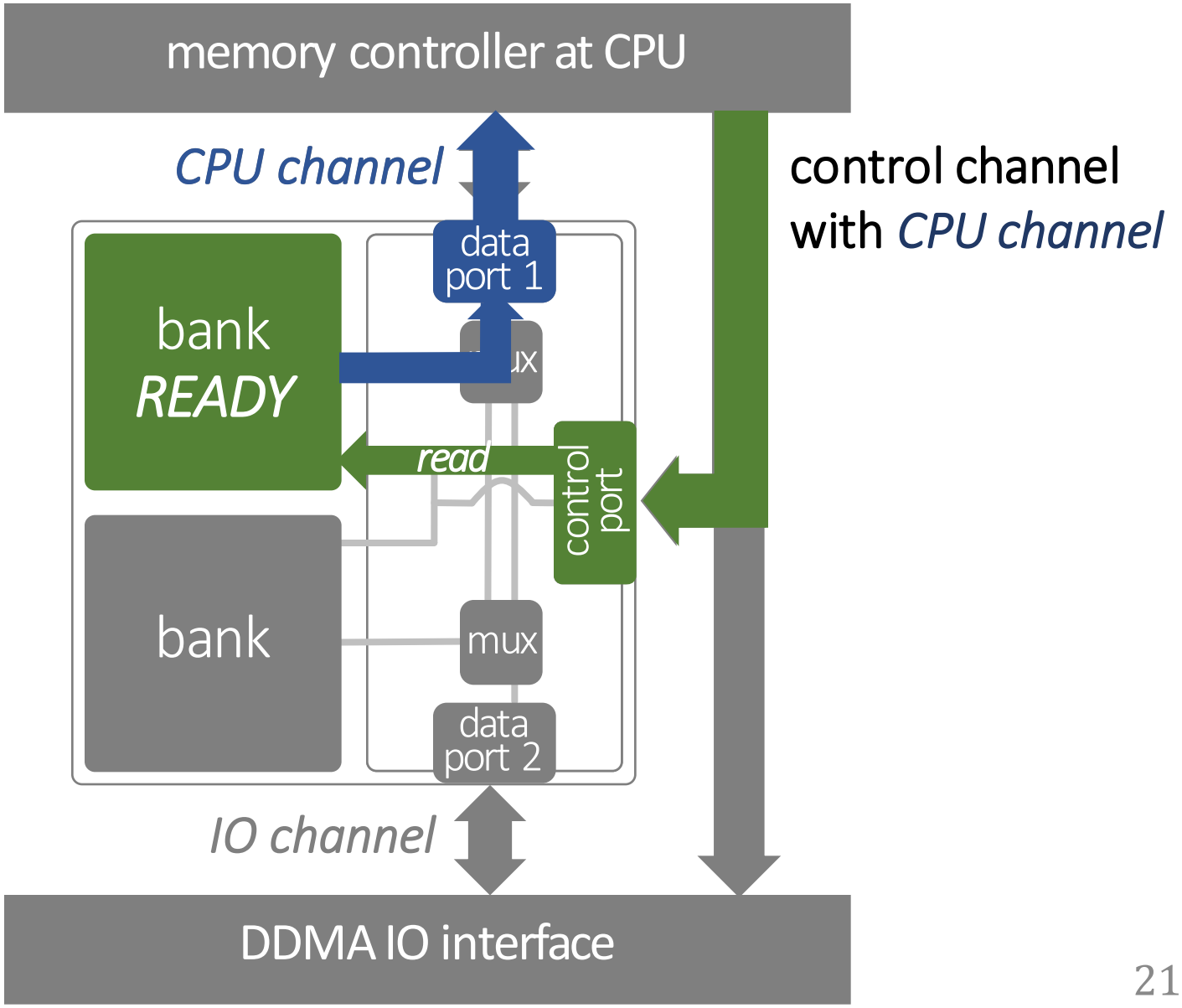
DDP-DRAM Memory System



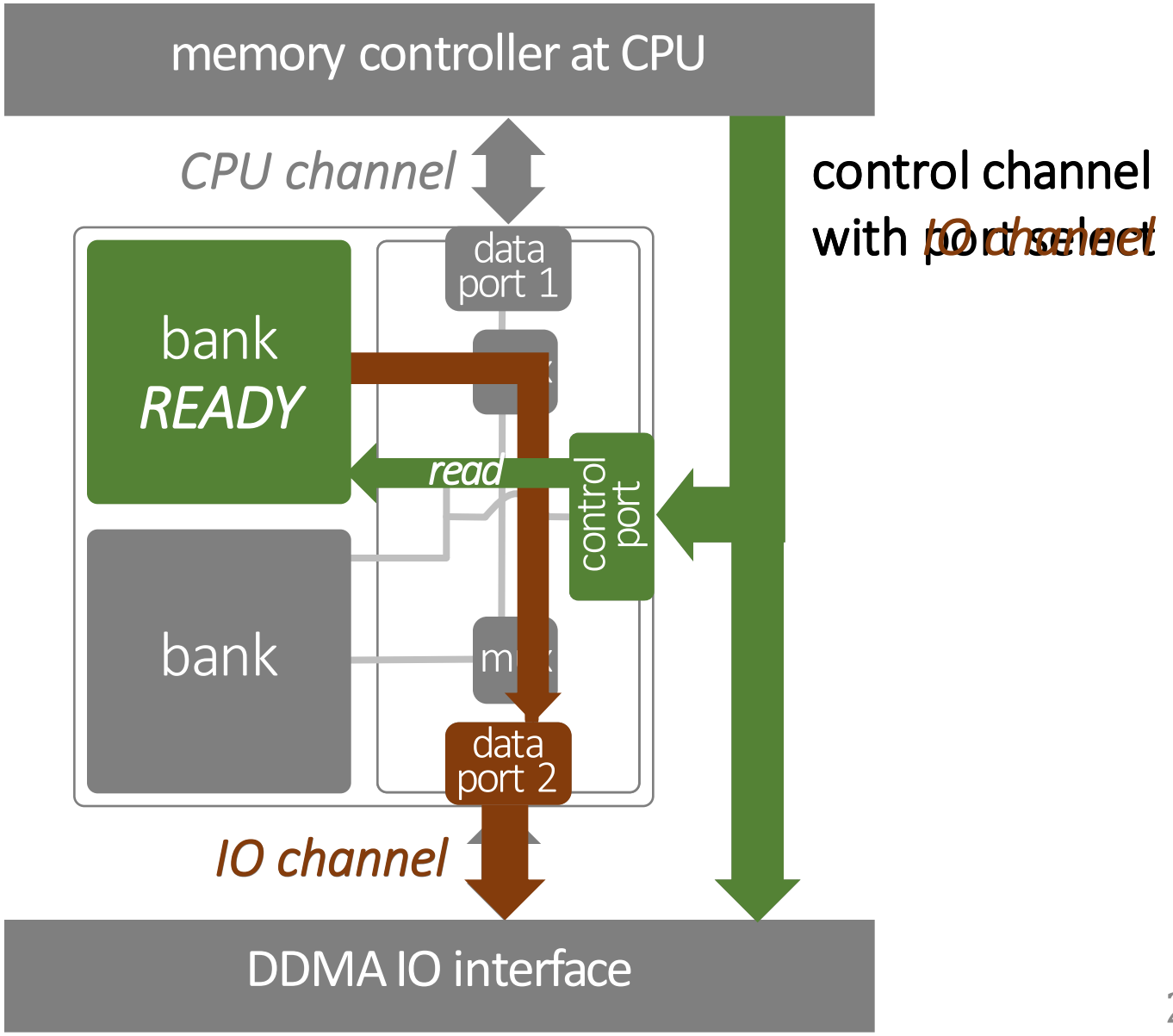
Three Data Transfer Modes

- **CPU Access**: Access through CPU channel
  - DRAM read/write with CPU port selection
- **IO Access**: Access through IO channel
  - DRAM read/write with IO port selection
- **Port Bypass**: Direct transfer between channels
  - DRAM access with port bypass selection

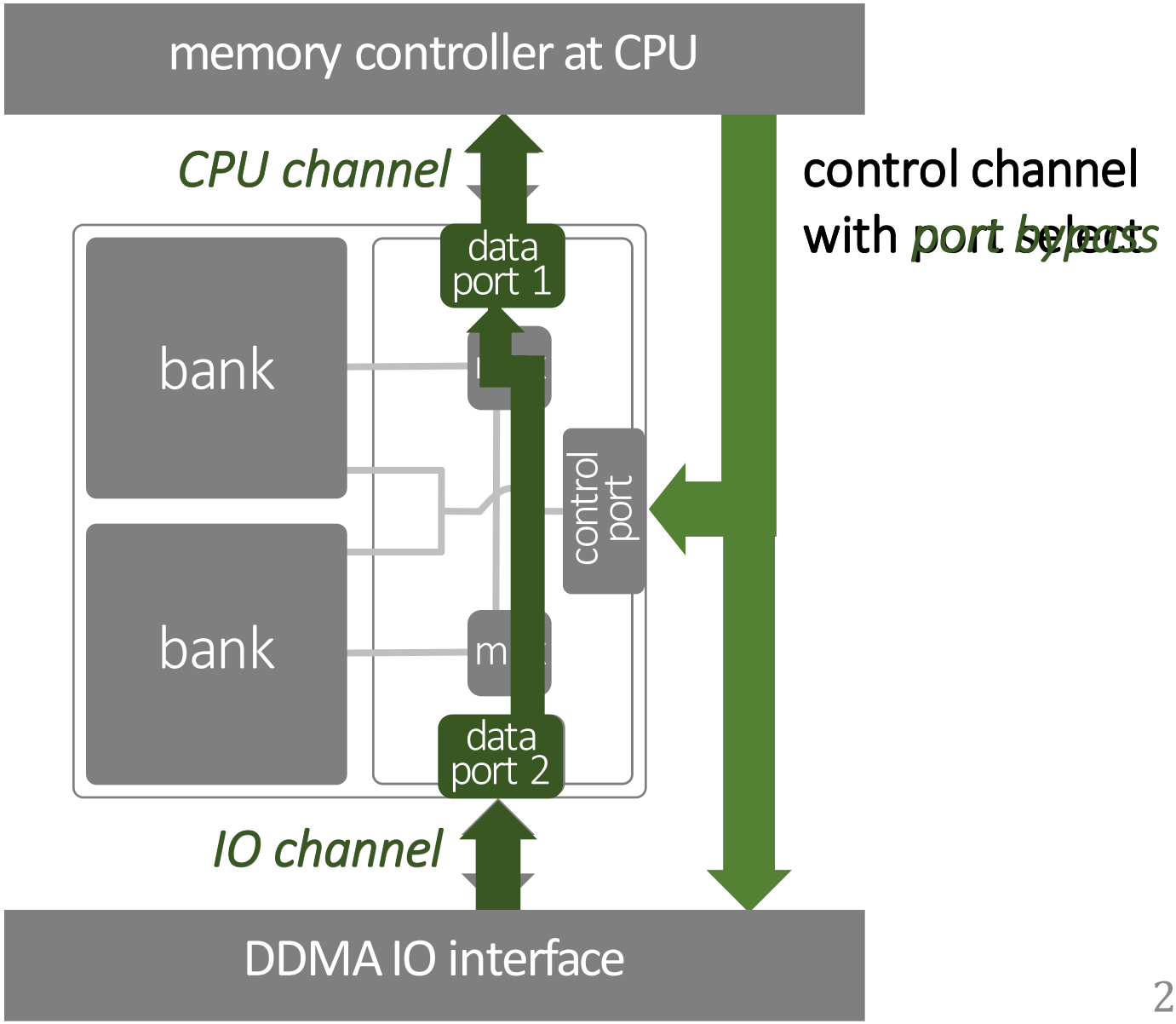
1. CPU Access Mode



2. IO Access Mode



3. Port Bypass Mode



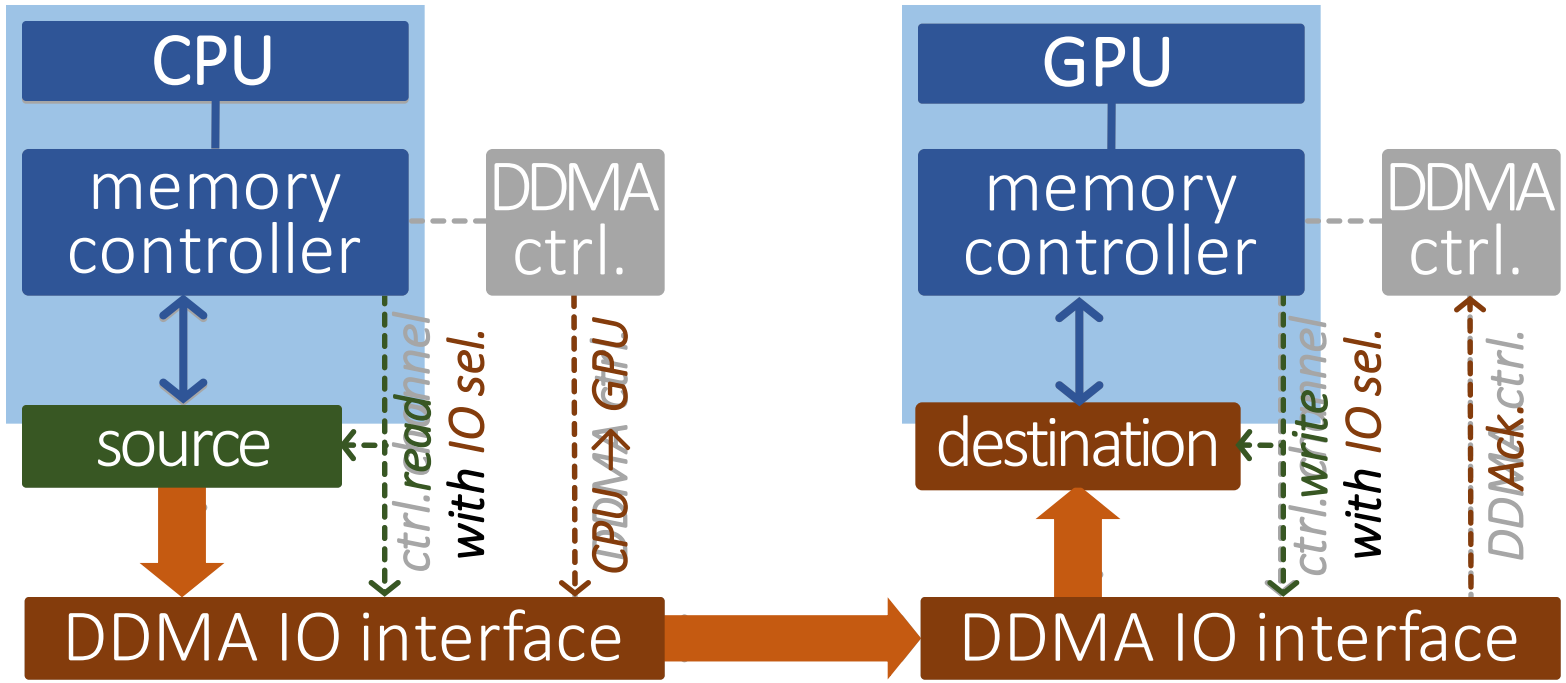
Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Three Applications for DDMA

- **Communication b/w Compute Units**
  - CPU-GPU communication
- **In-Memory Communication and Initialization**
  - Bulk page copy/initialization
- **Communication b/w Memory and Storage**
  - Serving page fault/file read & write

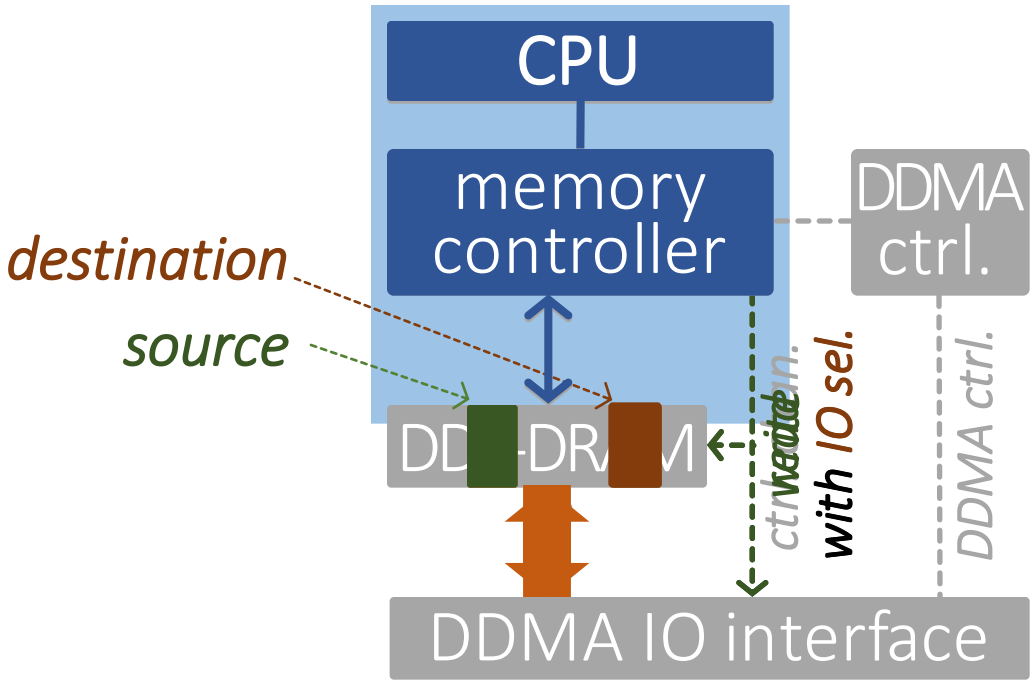
1. Compute Unit ↔ Compute Unit



Transfer data through DDMA

without interfering w/ CPU/GPU memory accesses

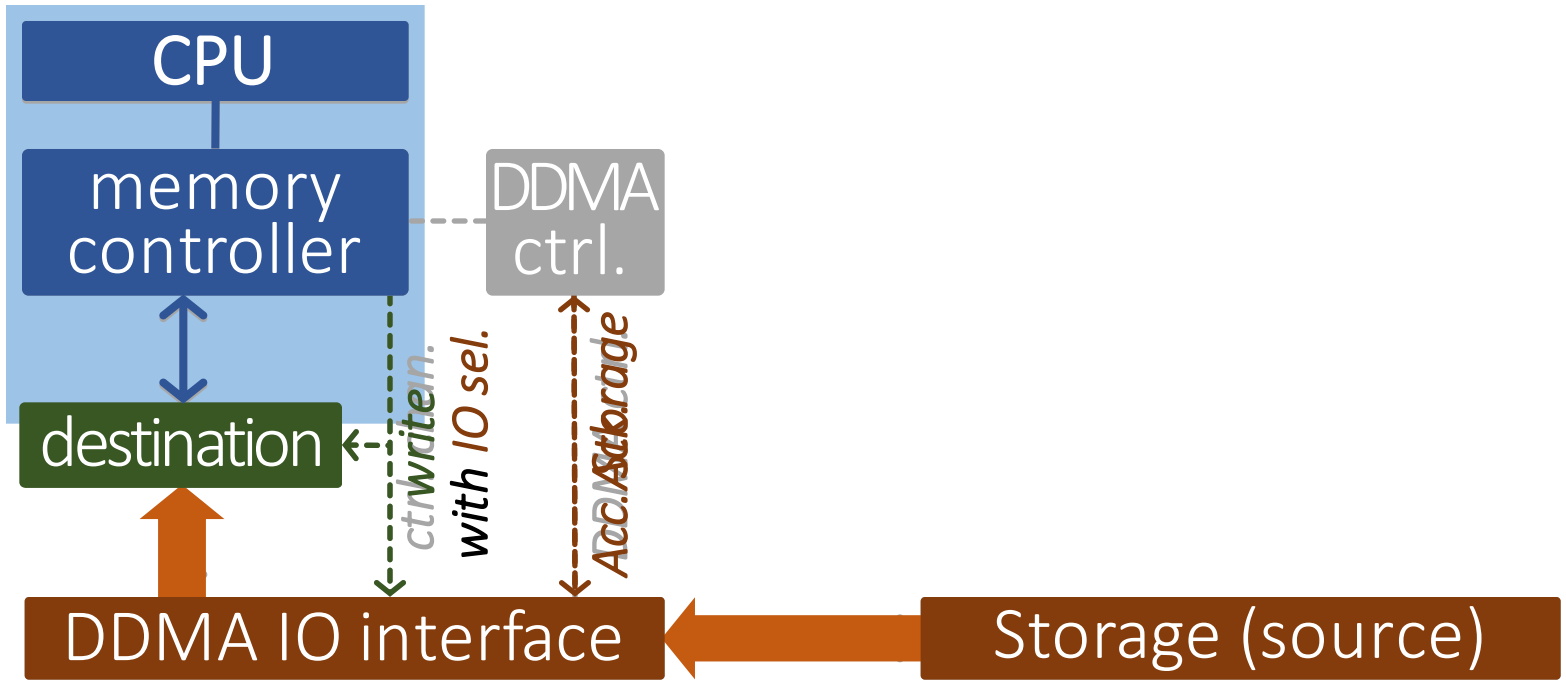
2. In-Memory Communication



Transfer data in DRAM through DDAM

without interfering with CPU memory accesses

3. Memory ↔ Storage



Transfer data from storage through DDMA

without interfering with CPU memory accesses

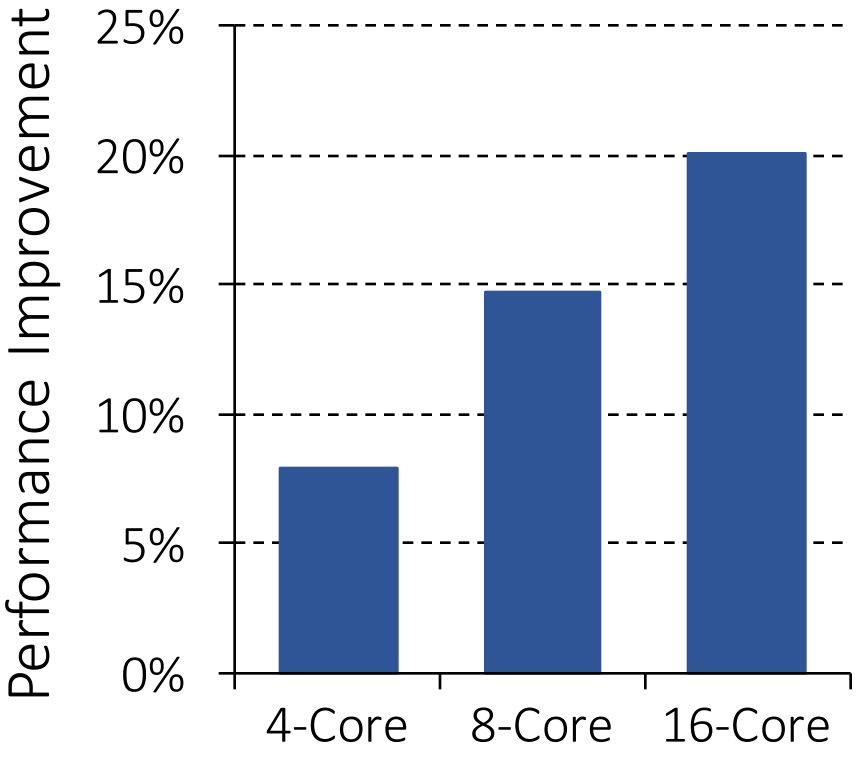
Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

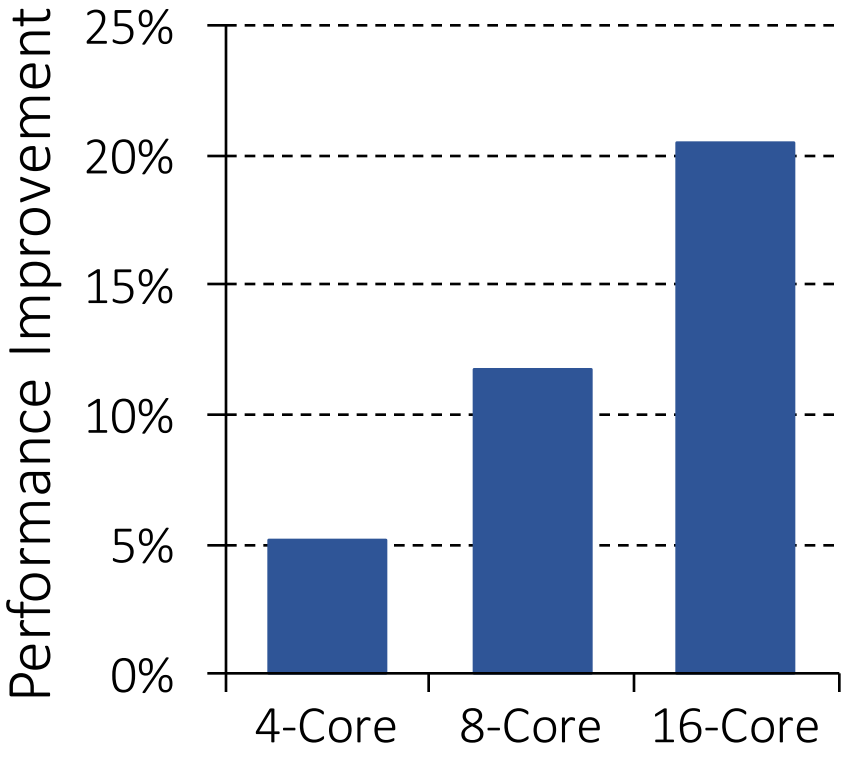
Evaluation Methods

- System
  - Processor: 4 – 16 cores
  - LLC: 16-way associative, 512KB private cache-slice/core
  - Memory: 1 – 4 ranks and 1 – 4 channels
- Workloads
  - **Memory intensive**:
    SPEC CPU2006, TPC, stream (31 benchmarks)
  - **CPU-GPU communication intensive**:
    polybench (8 benchmarks)
  - **In-memory communication intensive**:
    apache, bootup, compiler, filecopy, mysql, fork, shell, memcached (8 in total)

Performance (2 Channel, 2 Rank)



CPU-GPU Comm.-Intensive

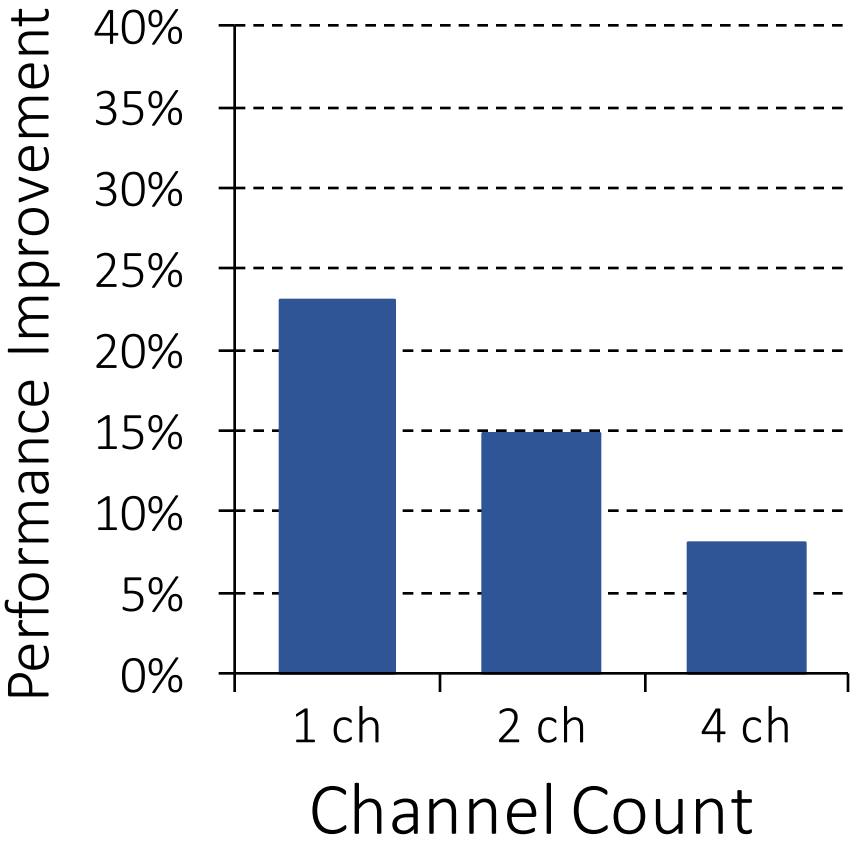


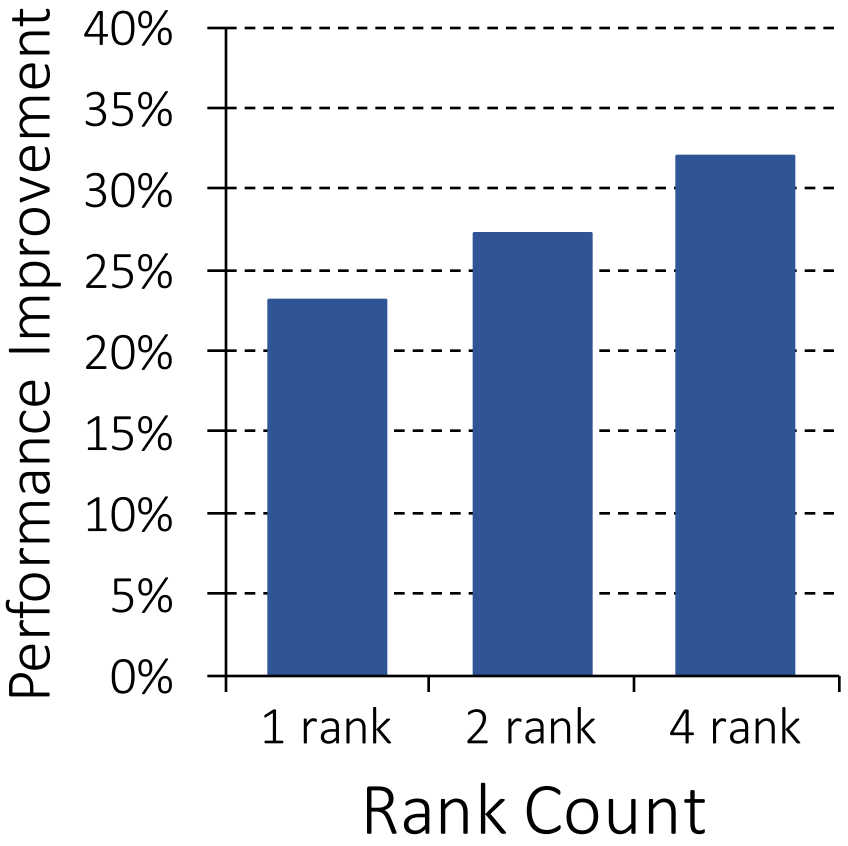
In-Memory Comm.-Intensive

High performance improvement

More performance improvement at *higher core count*

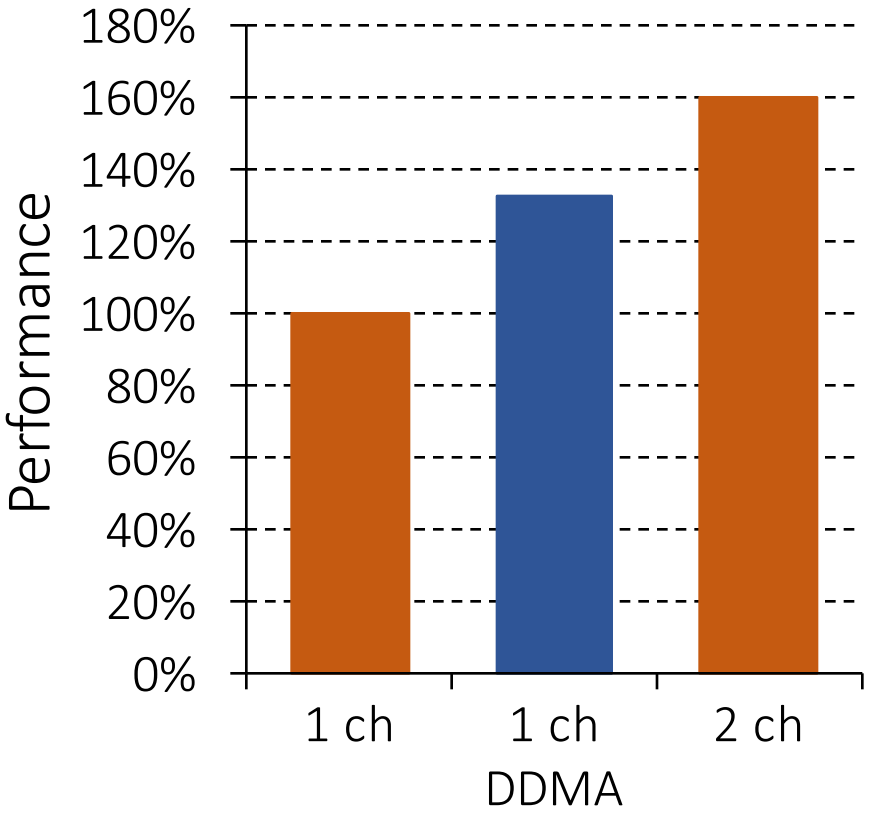
Performance on Various Systems

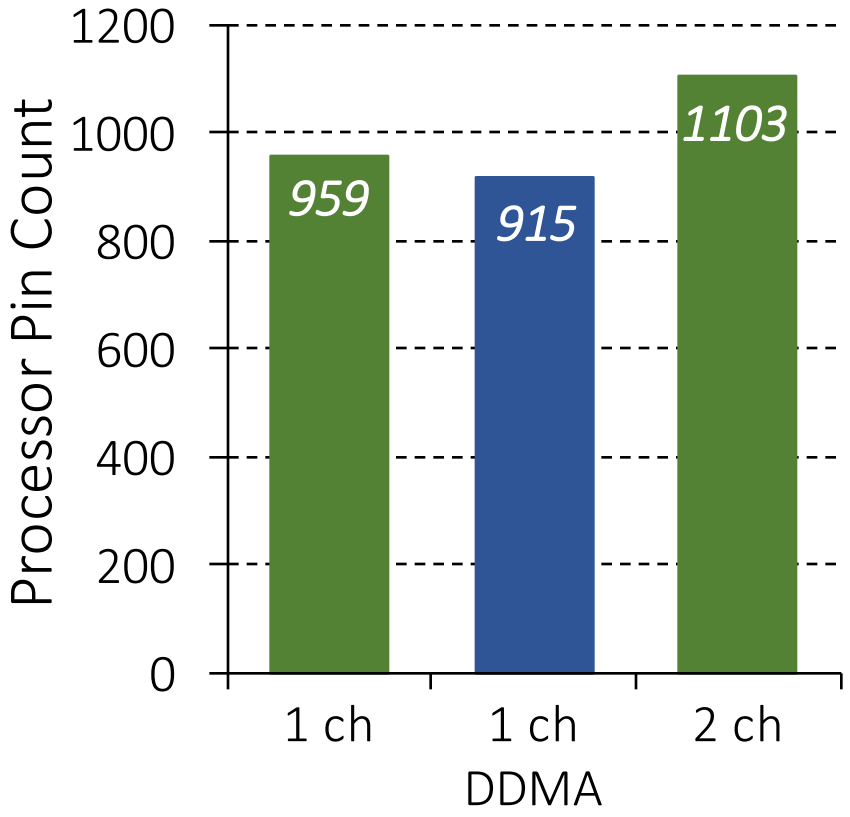




Performance increases with rank count

DDMA vs. Doubling Channel





DDMA achieves *higher performance* at *lower processor pin count*

Conclusion

- **Problem**
  - CPU and IO accesses contend for the shared memory channel
- **Our Approach:** *Decoupled Direct Memory Access (DDMA)*
  - Design new DRAM architecture with two independent data ports
    → *Dual-Data-Port DRAM*
  - Connect one port to CPU and the other port to IO devices
    → *Decouple CPU and IO accesses*
- **Application**
  - Communication between compute units (e.g., CPU – GPU)
  - In-memory communication (e.g., bulk in-memory copy/init.)
  - Memory-storage communication (e.g., page fault, IO prefetch)
- **Result**
  - Significant *performance improvement* (20% in 2 ch. & 2 rank system)
  - *CPU pin count reduction* (4.5%)

Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM

Decoupled Direct Memory Access

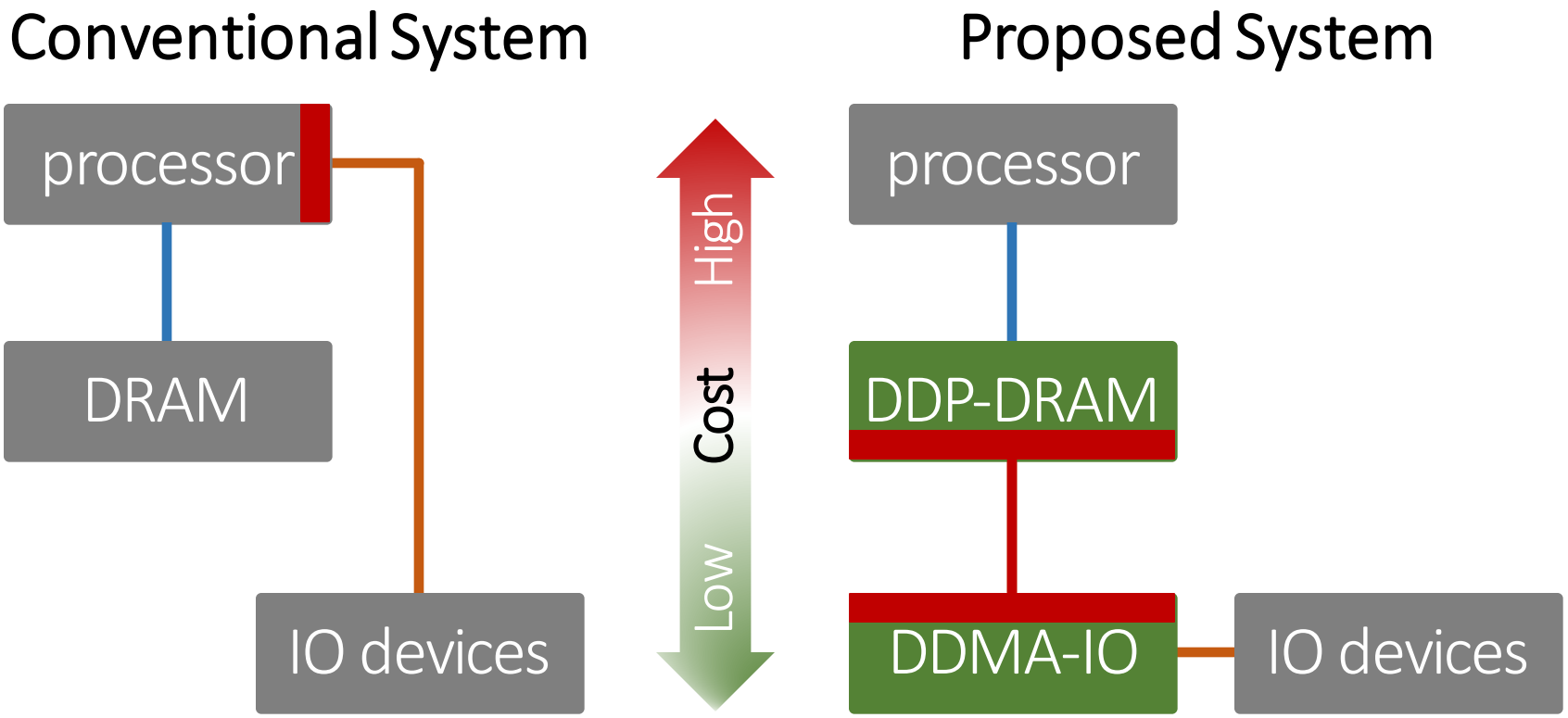
Donghyuk Lee

Lavanya Subramanian, Rachata Ausavarungnirun,
Jongmoo Choi, Onur Mutlu

SAFARI

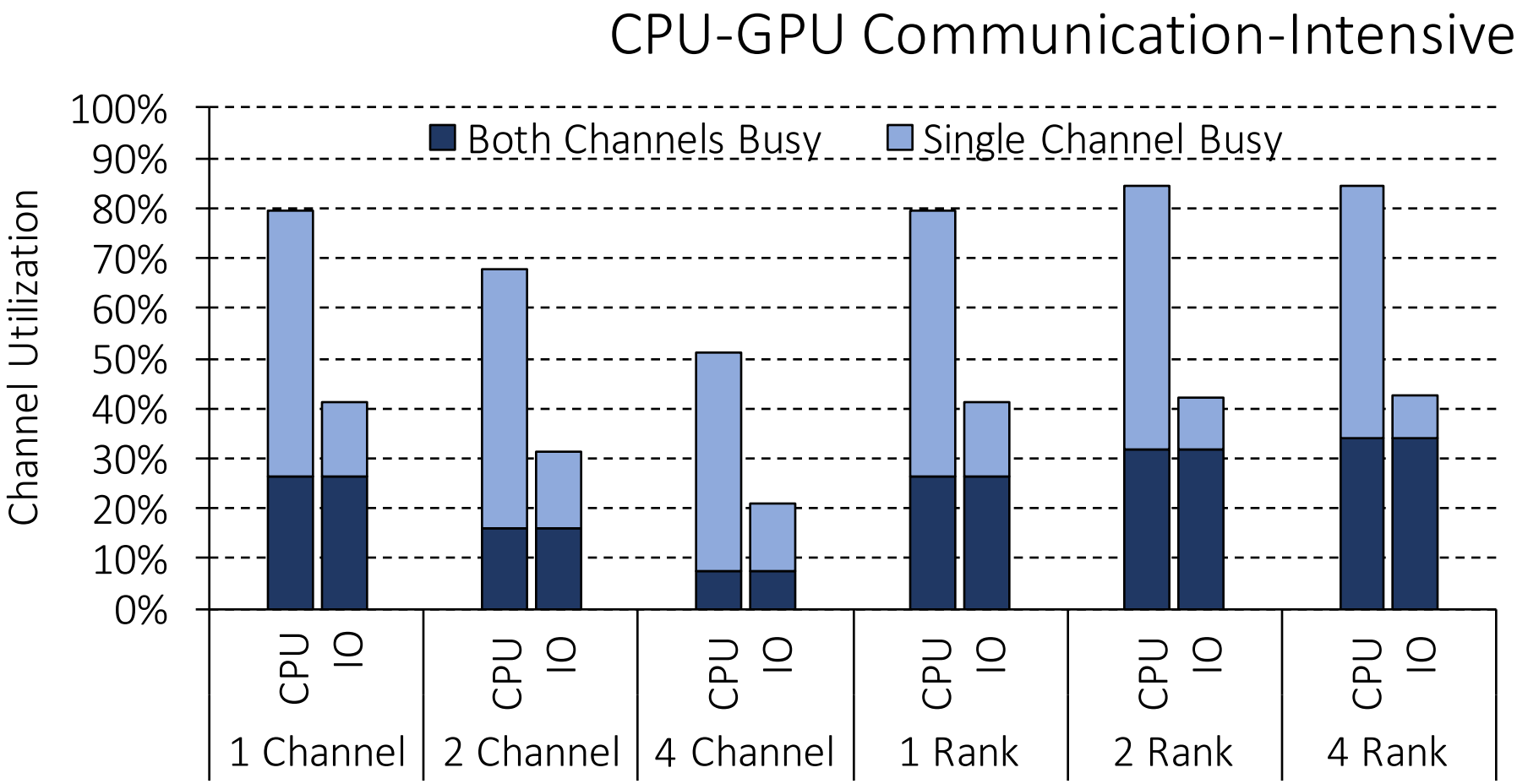
Carnegie Mellon

System Overhead



DDMA reduces more expensive on-chip area, while *increasing less expensive off-chip area*

Channel Utilization Analysis



Simultaneous Channel Utilization → Performance Improvement