Tiered-Latency DRAM: A Low Latency and A Low Cost DRAM Architecture

Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, Onur Mutlu

Carnegie Mellon

SAFARI

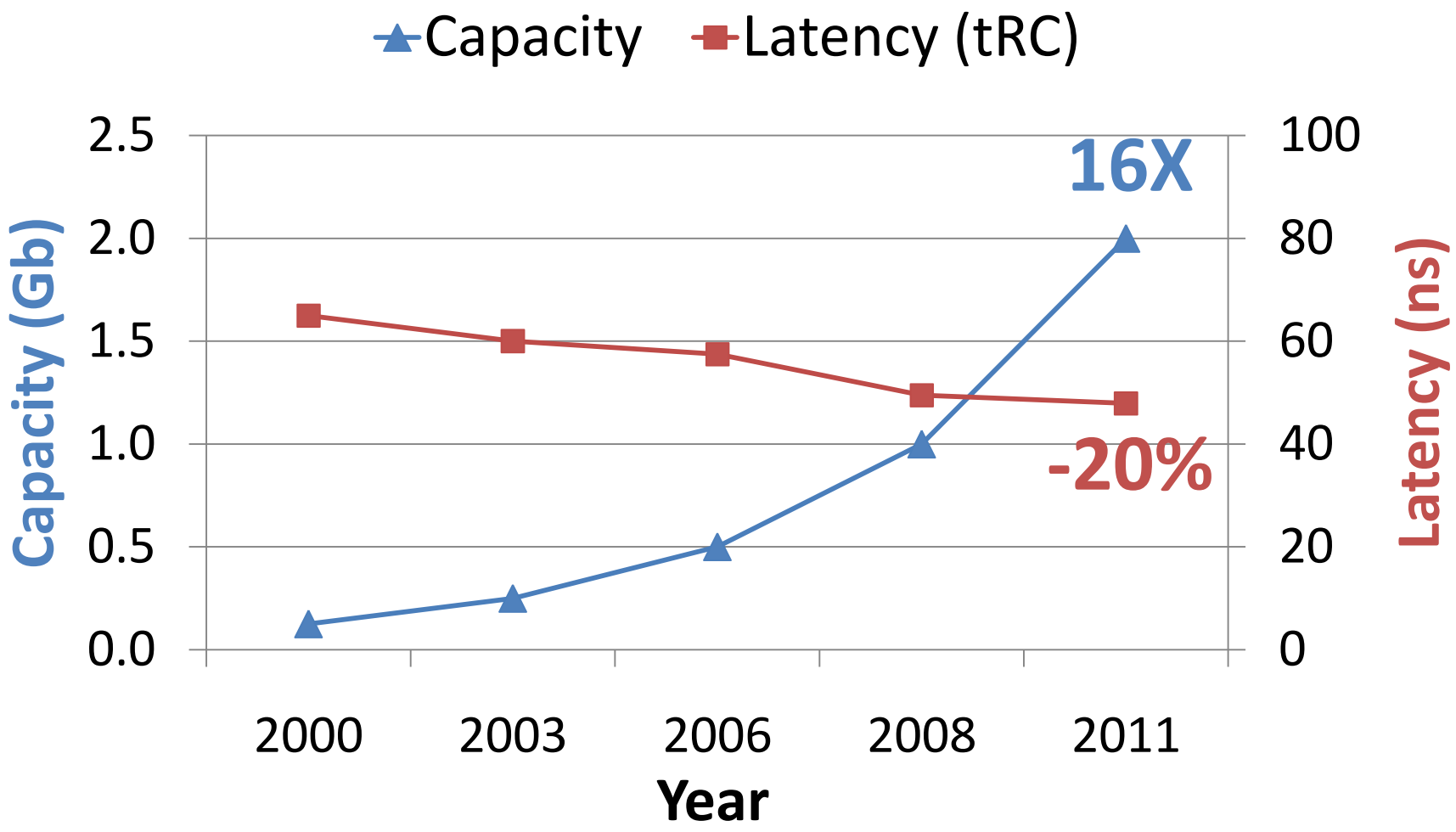
Executive Summary

- **Problem: DRAM latency is a critical performance bottleneck**
- **Our Goal**: Reduce DRAM latency with low area cost
- **Observation**: Long bitlines in DRAM are the dominant source of DRAM latency
- **Key Idea: Divide long bitlines into two shorter segments**
  - **Fast and slow segments**
- **Tiered-latency DRAM: Enables latency heterogeneity in DRAM**
  - **Can leverage this in many ways to improve performance and reduce power consumption**
- **Results**: When the fast segment is used as a cache to the slow segment → Significant performance improvement (>12%) and power reduction (>23%) at low area cost (3%)

Outline

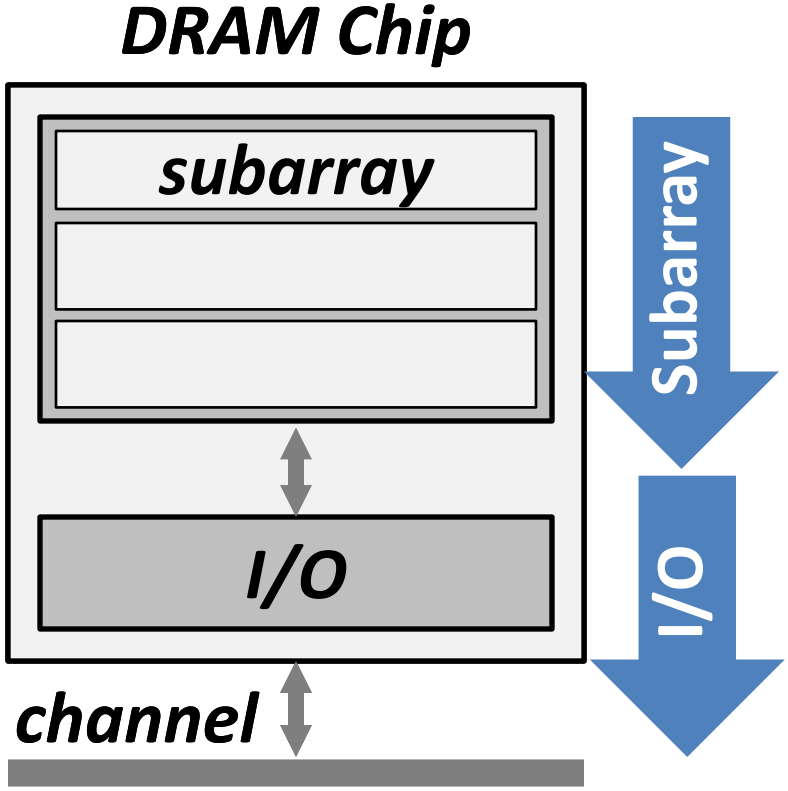
- **Motivation & Key Idea**
- Tiered-Latency DRAM
- Leveraging Tiered-Latency DRAM
- Evaluation Results

Historical DRAM Trend



DRAM latency continues to be a critical bottleneck

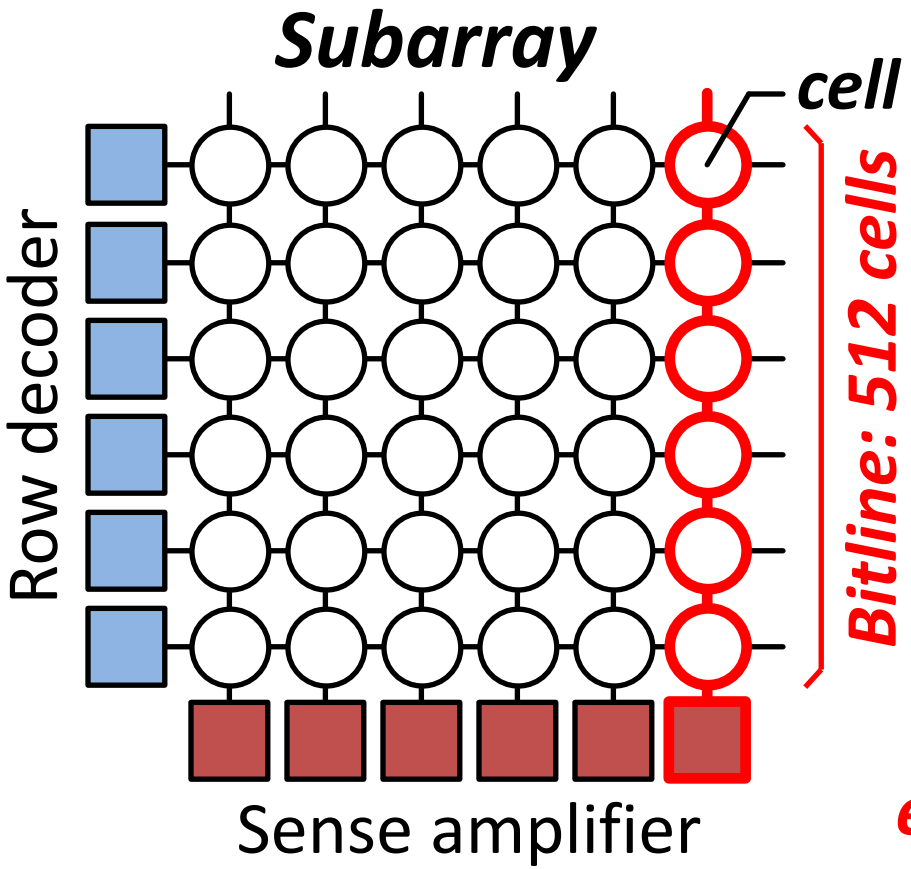
What Causes the Long Latency?

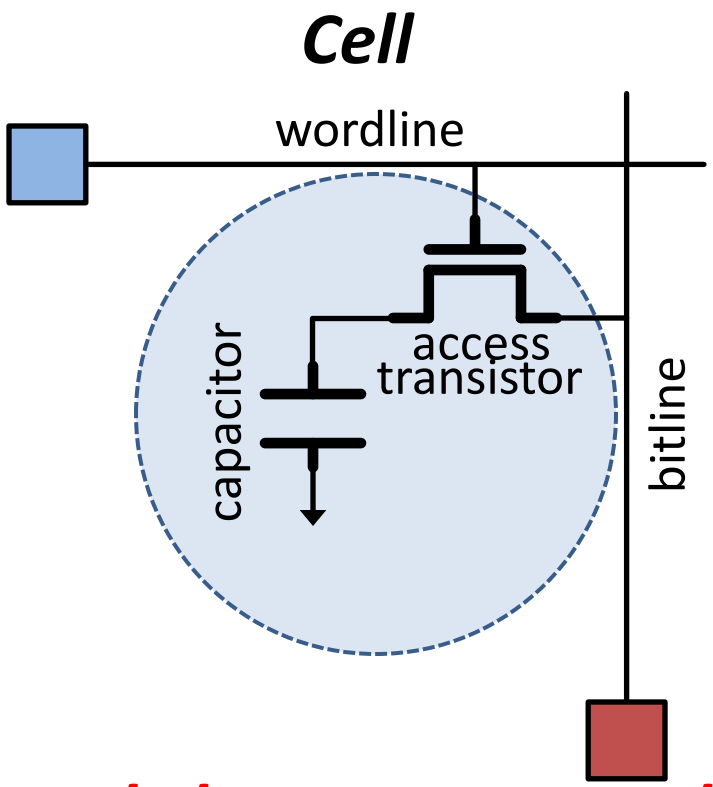


DRAM Latency = Subarray Latency + I/O Latency

Dominant

Why is the Subarray So Slow?



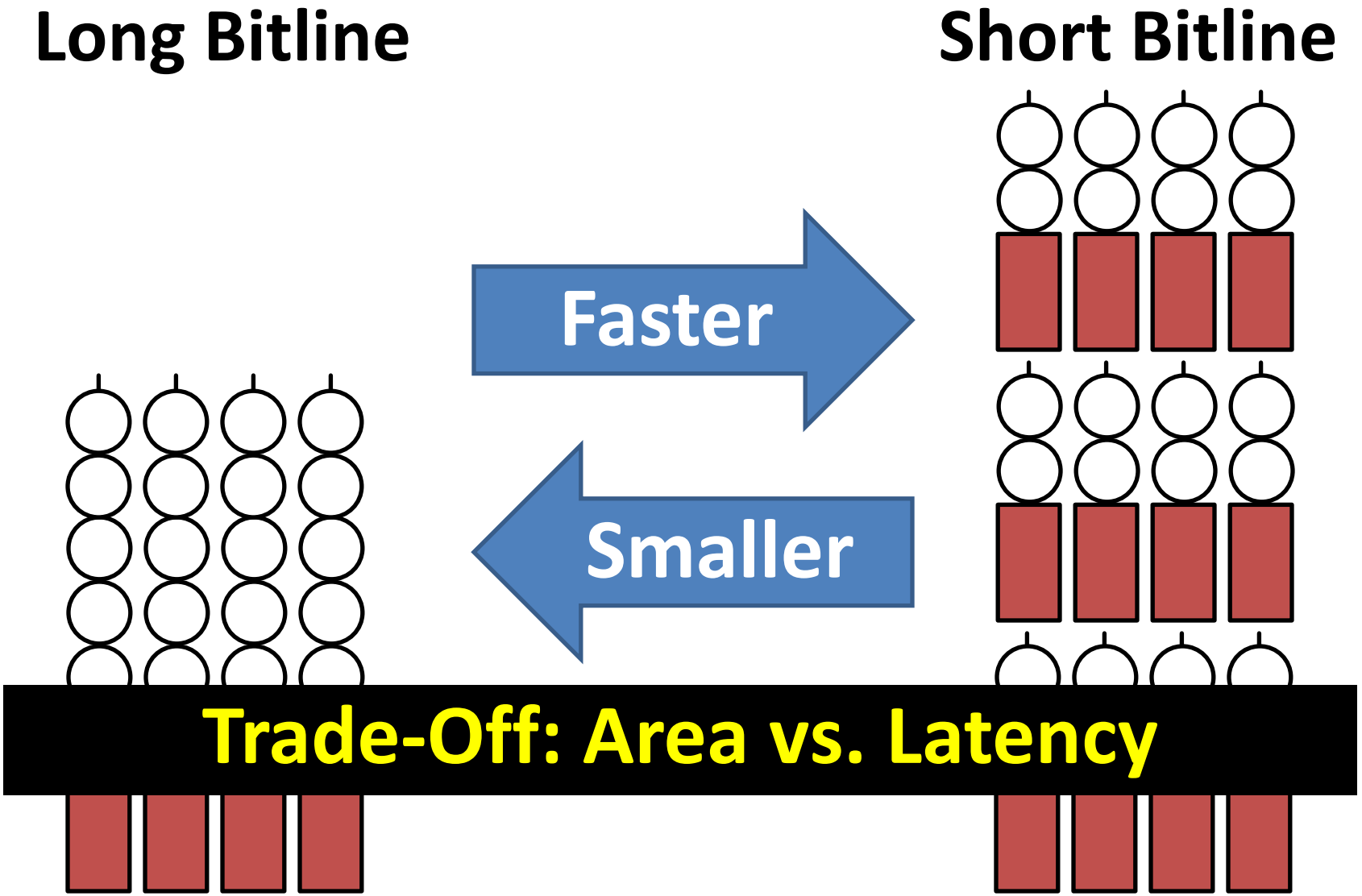


extremely large sense amplifier (≈100X the cell size)

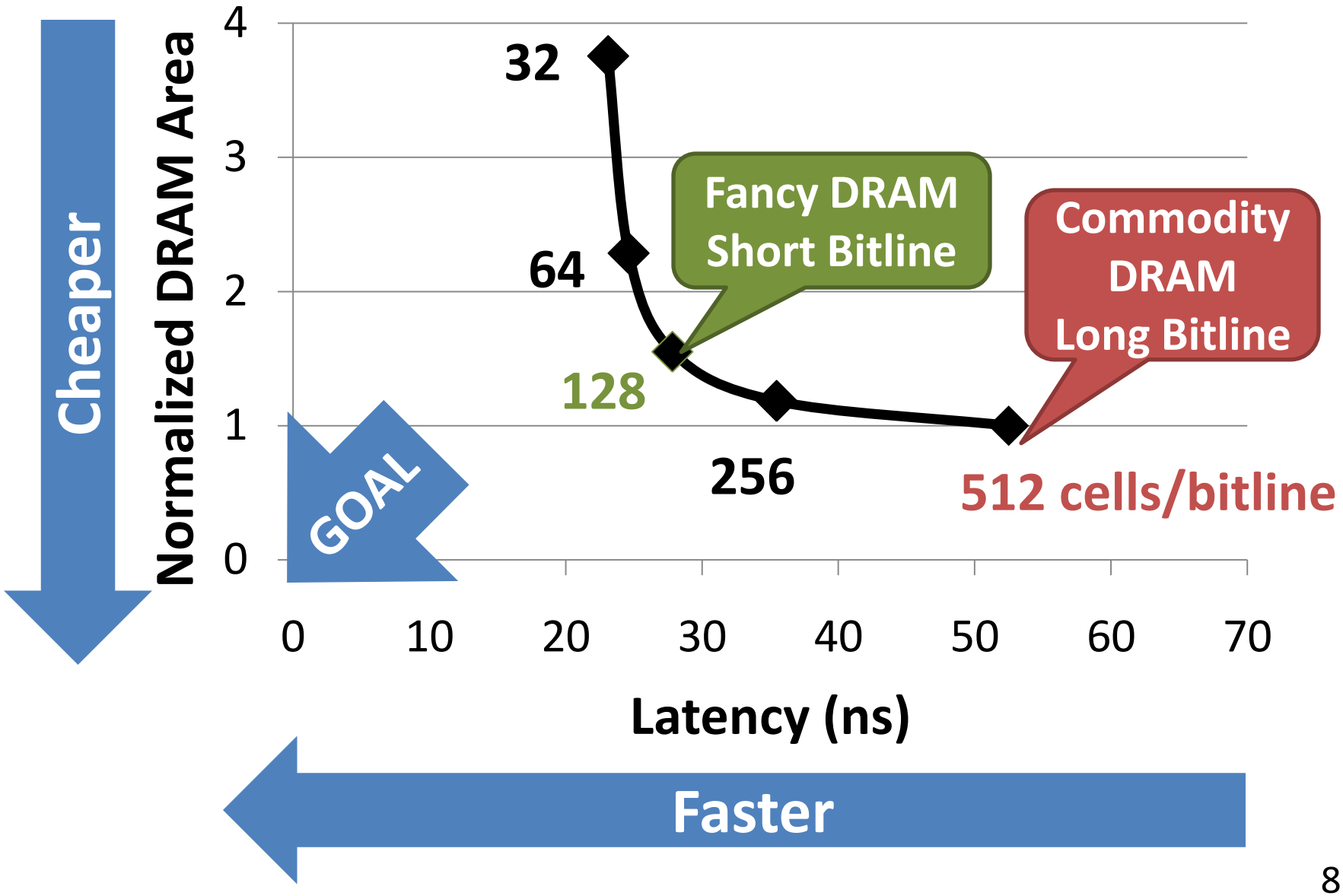
Long Bitline: Amortize sense amplifier → Small area

Long Bitline: Large bitline cap. → High latency

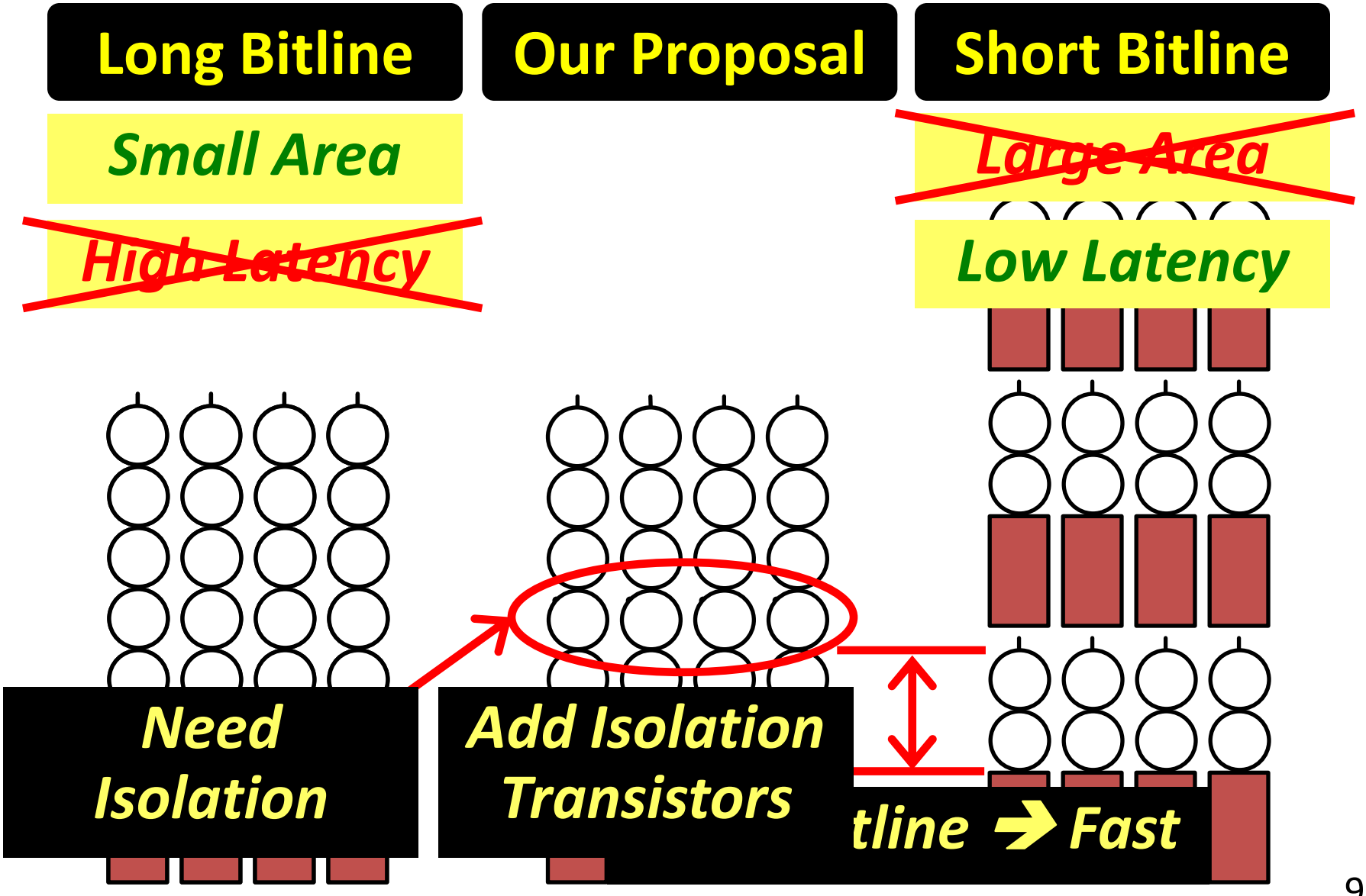
Trade-Off: Area (Die Size) vs. Latency



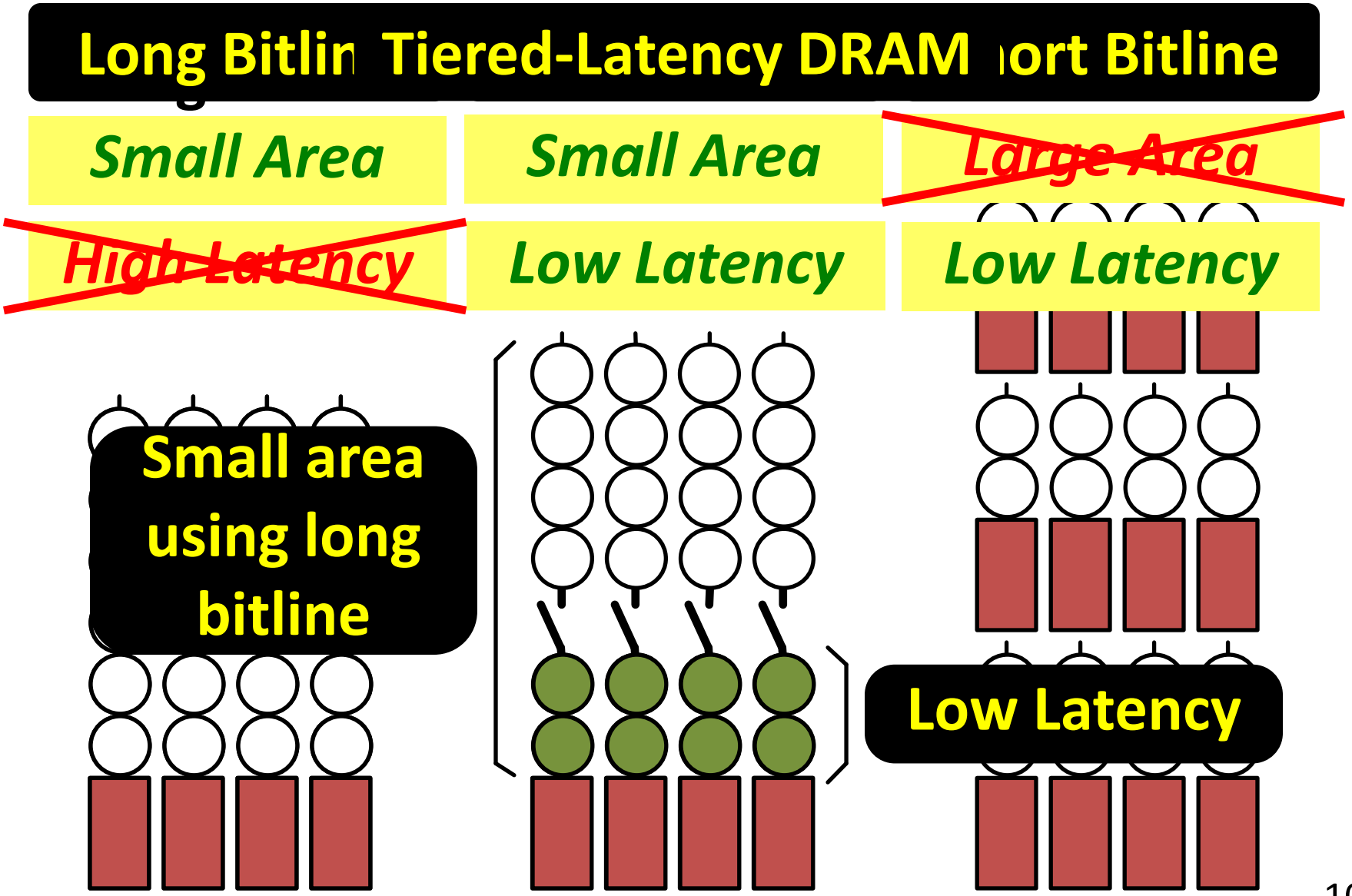
Trade-Off: Area (Die Size) vs. Latency



Approximating the Best of Both Worlds



Approximating the Best of Both Worlds

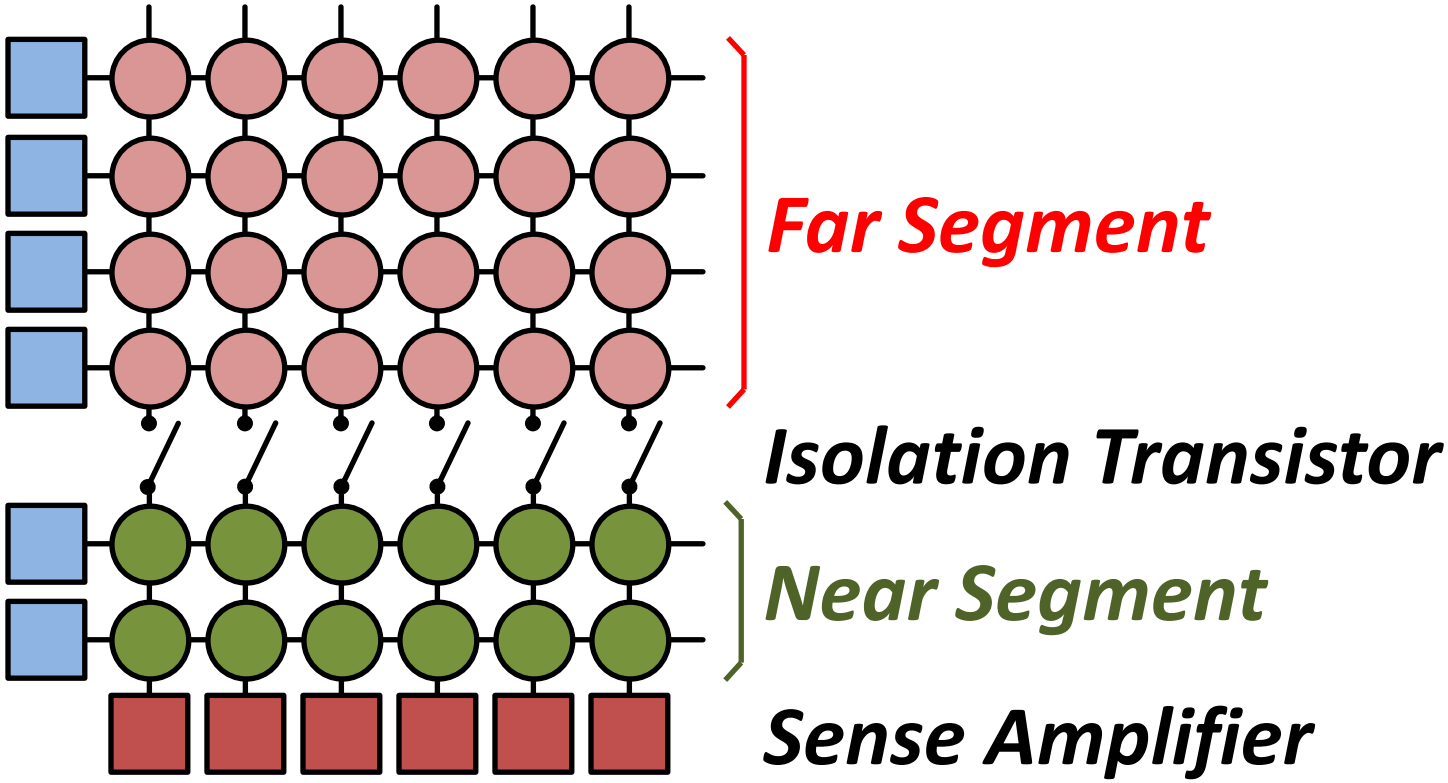


Outline

- Motivation & Key Idea
- **Tiered-Latency DRAM**
- Leveraging Tiered-Latency DRAM
- Evaluation Results

Tiered-Latency DRAM

- Divide a bitline into two segments with an **isolation transistor**



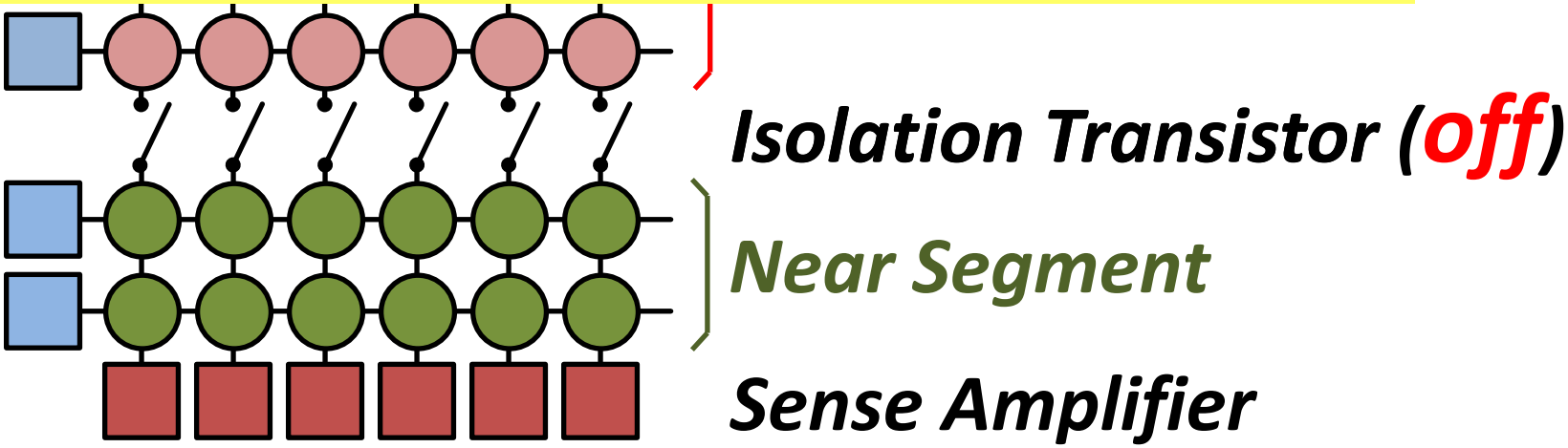
Near Segment Access

- Turn *off* the isolation transistor

Reduced bitline length

Reduced bitline capacitance

➔ Low latency & low power



Far Segment Access

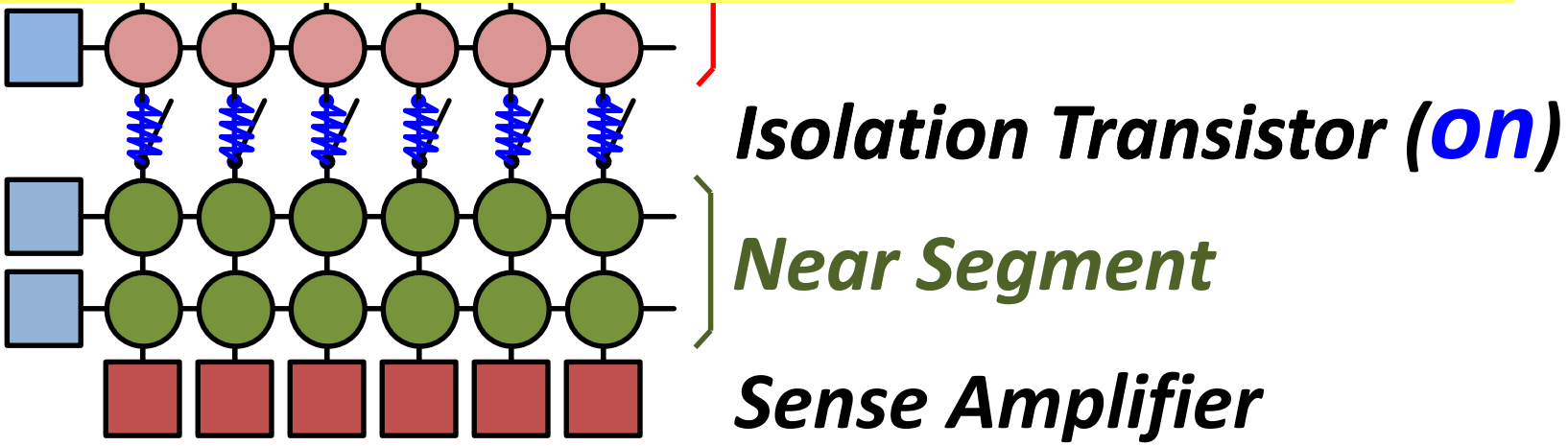
- Turn ***on*** the isolation transistor

Long bitline length

Large bitline capacitance

Additional resistance of isolation transistor

➔ High latency & high power

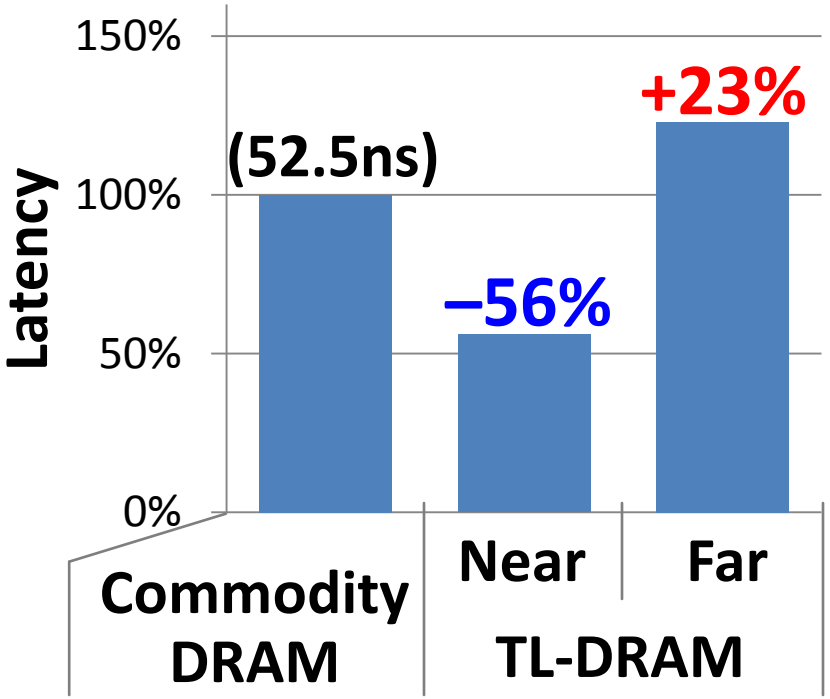


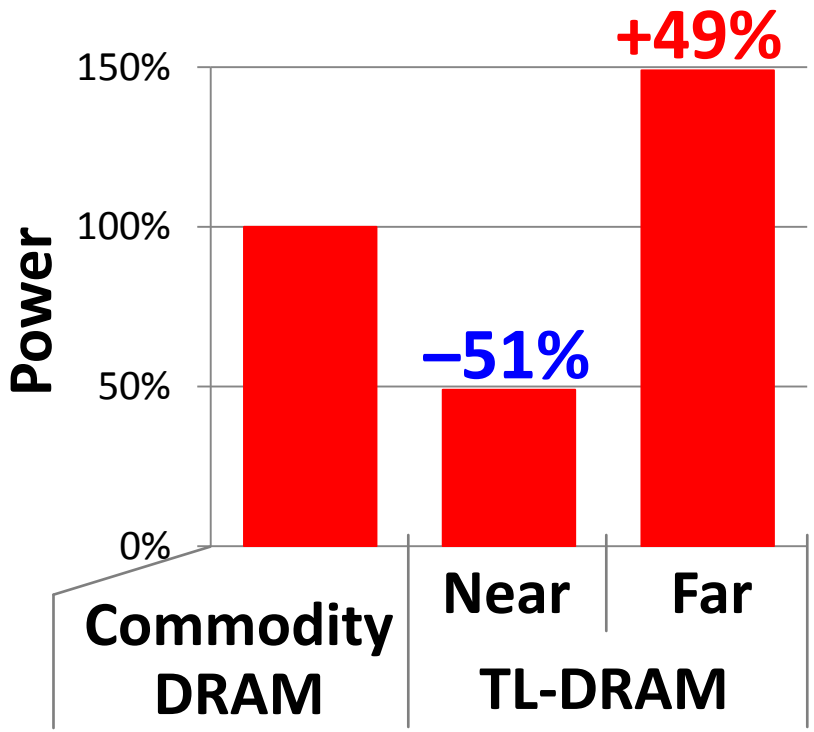
Latency, Power, and Area Evaluation

- **Commodity DRAM:** 512 cells/bitline
- **TL-DRAM:** 512 cells/bitline
  - Near segment: 32 cells
  - Far segment: 480 cells
- **Latency Evaluation**
  - SPICE simulation using circuit-level DRAM model
- **Power and Area Evaluation**
  - DRAM area/power simulator from Rambus
  - DDR3 energy calculator from Micron

Commodity DRAM vs. TL-DRAM

- **DRAM Latency** (tRC)
- **DRAM Power**

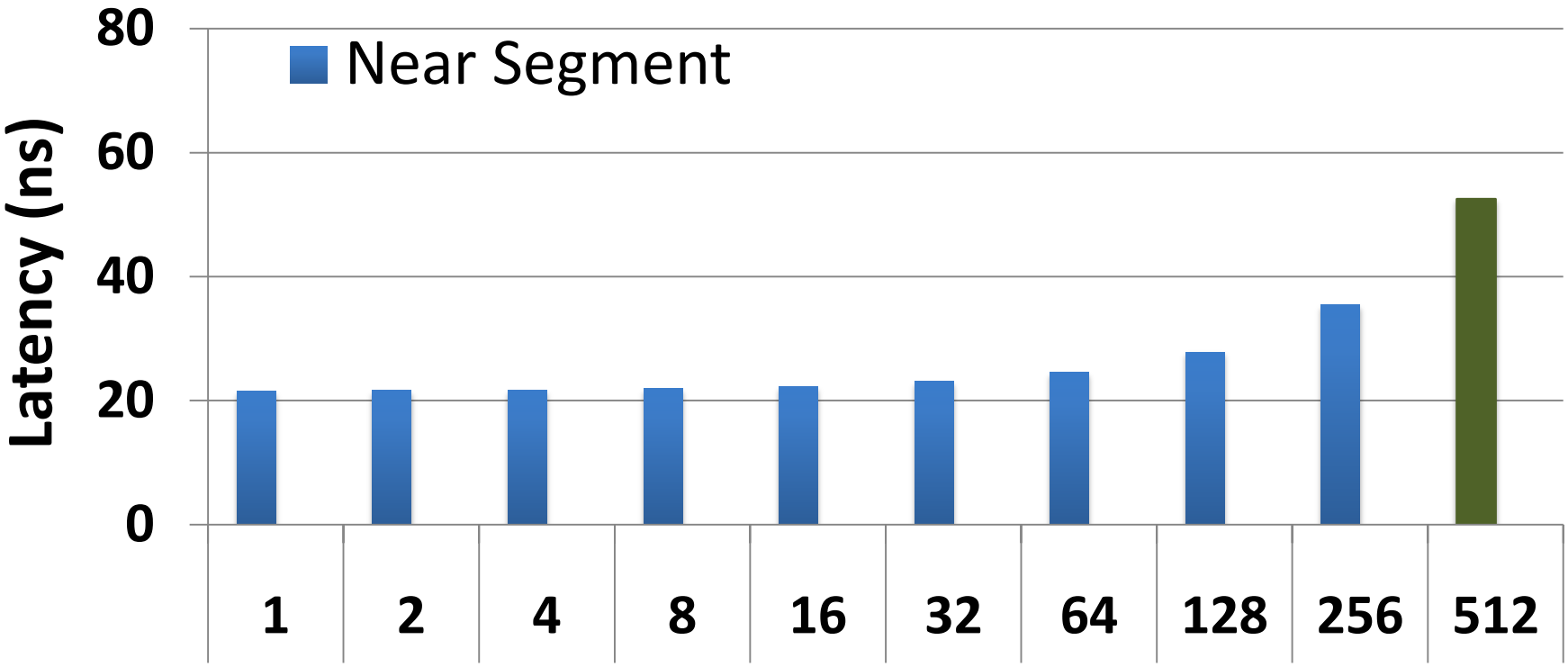




- **DRAM Area Overhead**

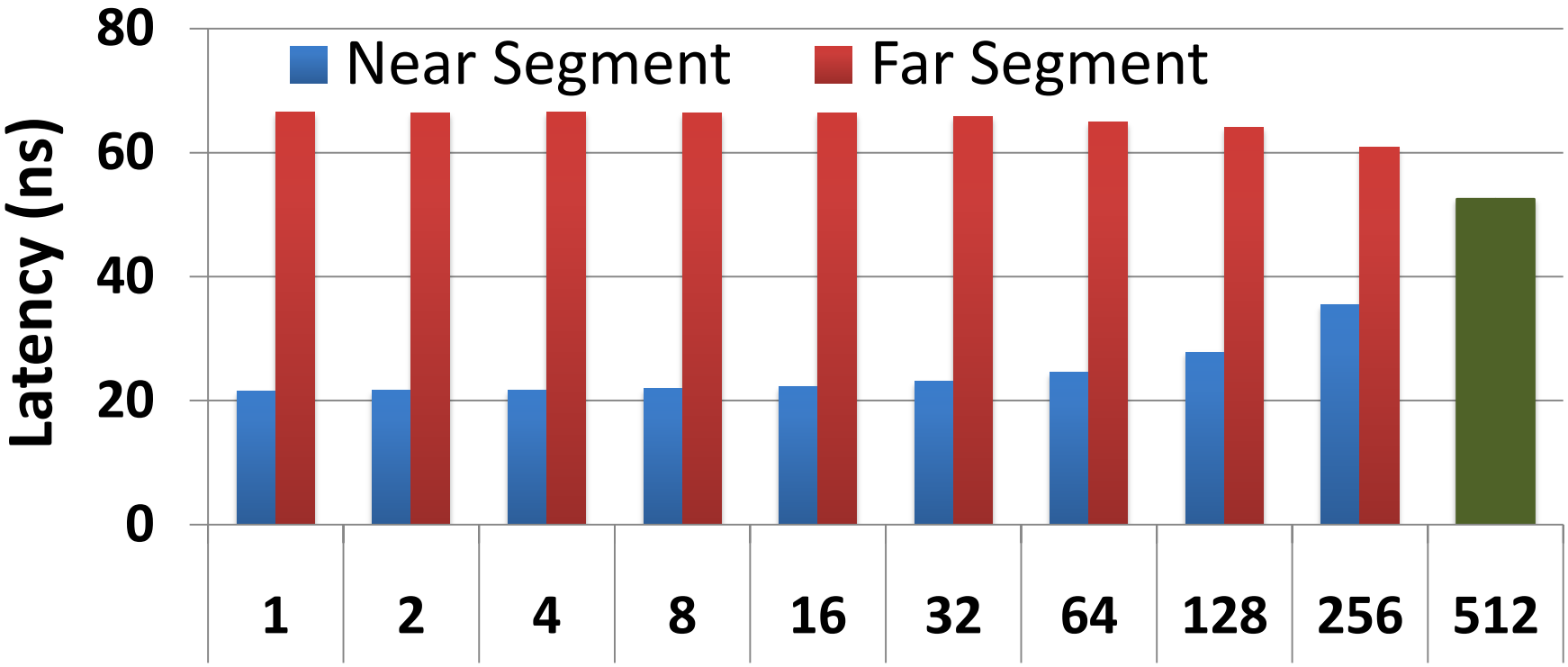
  ~3%: mainly due to the isolation transistors

Latency vs. Near Segment Length



Longer near segment length leads to higher near segment latency

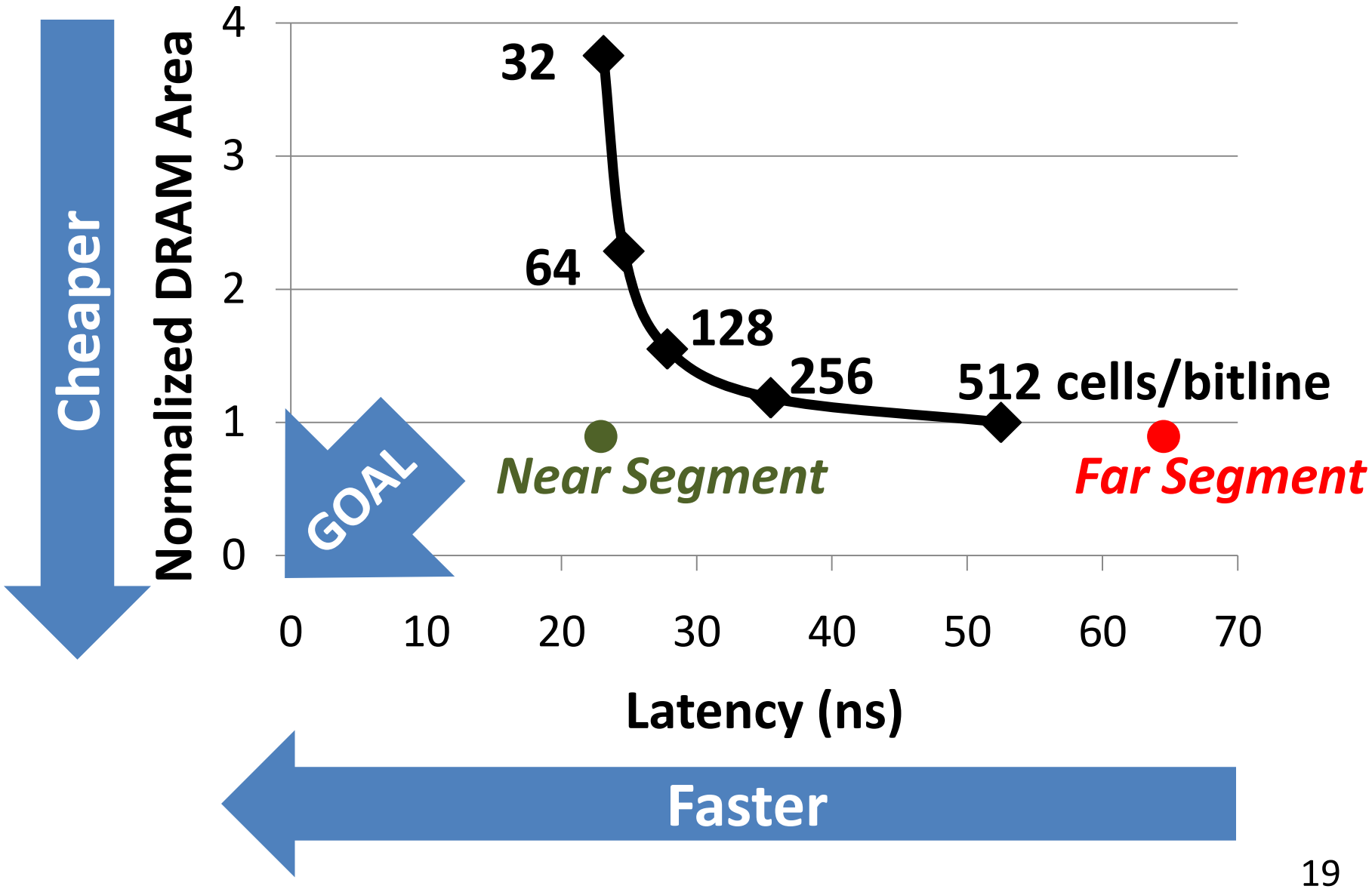
Latency vs. Near Segment Length



Far Segment Length = 512 – Near Segment Length

Far segment latency is higher than commodity DRAM latency

Trade-Off: Area (Die-Area) vs. Latency



Outline

- Motivation & Key Idea
- Tiered-Latency DRAM
- **Leveraging Tiered-Latency DRAM**
- Evaluation Results

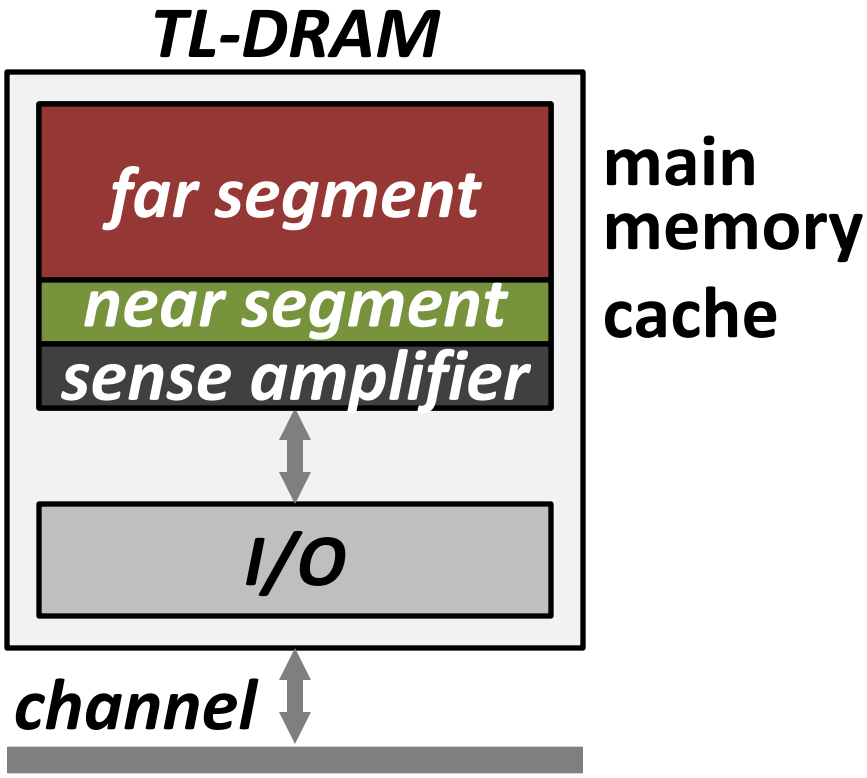
Leveraging Tiered-Latency DRAM

- TL-DRAM is a ***substrate*** that can be leveraged by the hardware and/or software

- Many potential uses

1. Use near segment as hardware-managed ***inclusive*** cache to far segment
2. Use near segment as hardware-managed ***exclusive*** cache to far segment
3. Profile-based page mapping by operating system
4. Simply replace DRAM with TL-DRAM

Near Segment as Hardware-Managed Cache

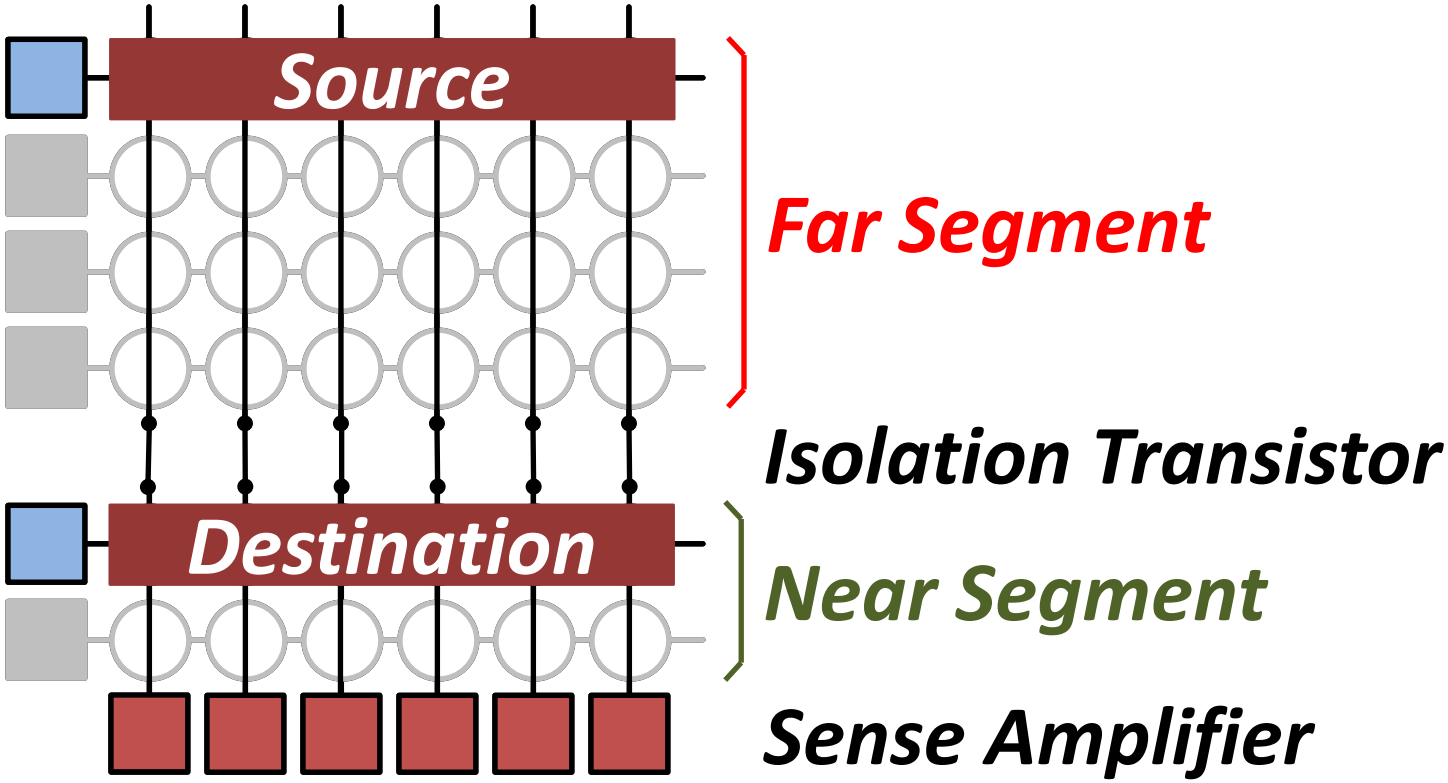


- **Challenge 1:** How to efficiently migrate a row between segments?
- **Challenge 2:** How to efficiently manage the cache?

Inter-Segment Migration

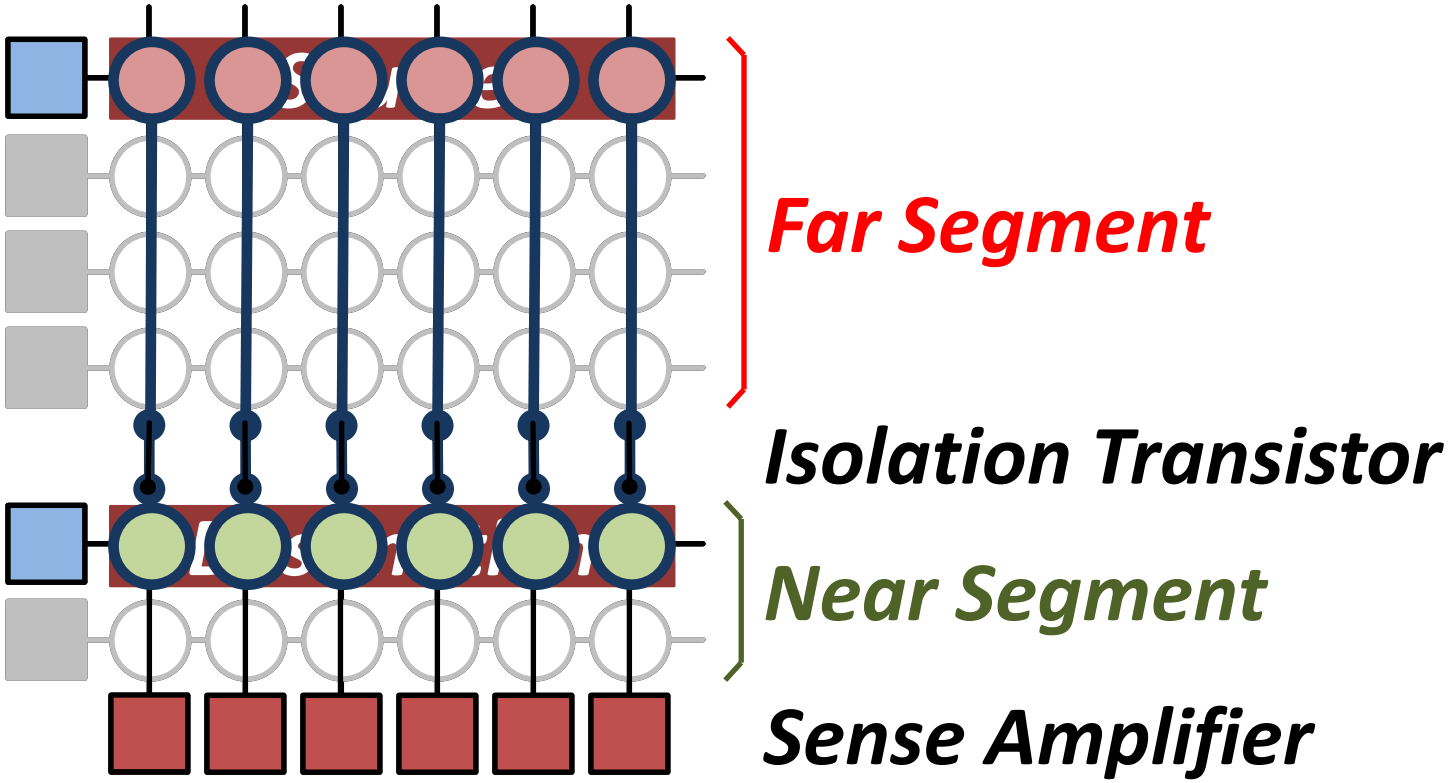
- **Goal:** Migrate source row into destination row
- **Naïve way:** Memory controller reads the source row *byte by byte* and writes to destination row *byte by byte*

→ High latency



Inter-Segment Migration

- **Our way:**
  - Source and destination cells ***share bitlines***
  - Transfer data from source to destination across ***shared bitlines*** concurrently



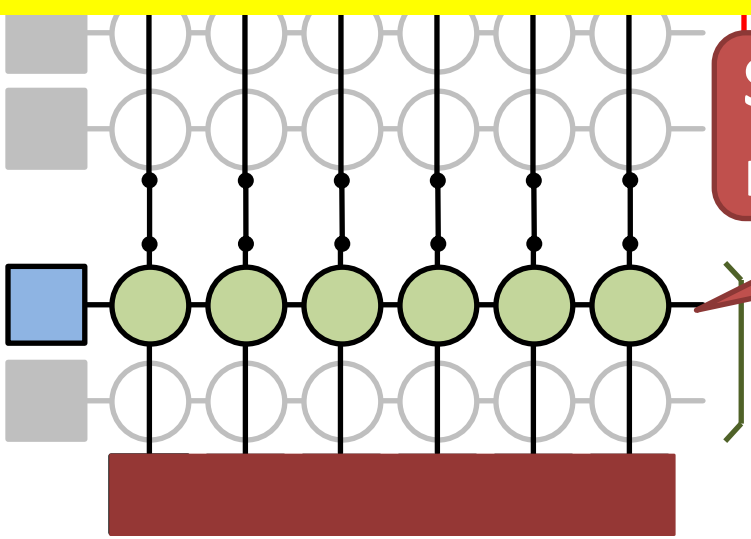
Inter-Segment Migration

- **Our way:**
  - Source and destination cells ***share bitlines***
  - Transfer data from so
    shared bitlines concu

Step 1: Activate source row

Migration is overlapped with source row access

Additional ~4ns over row access latency

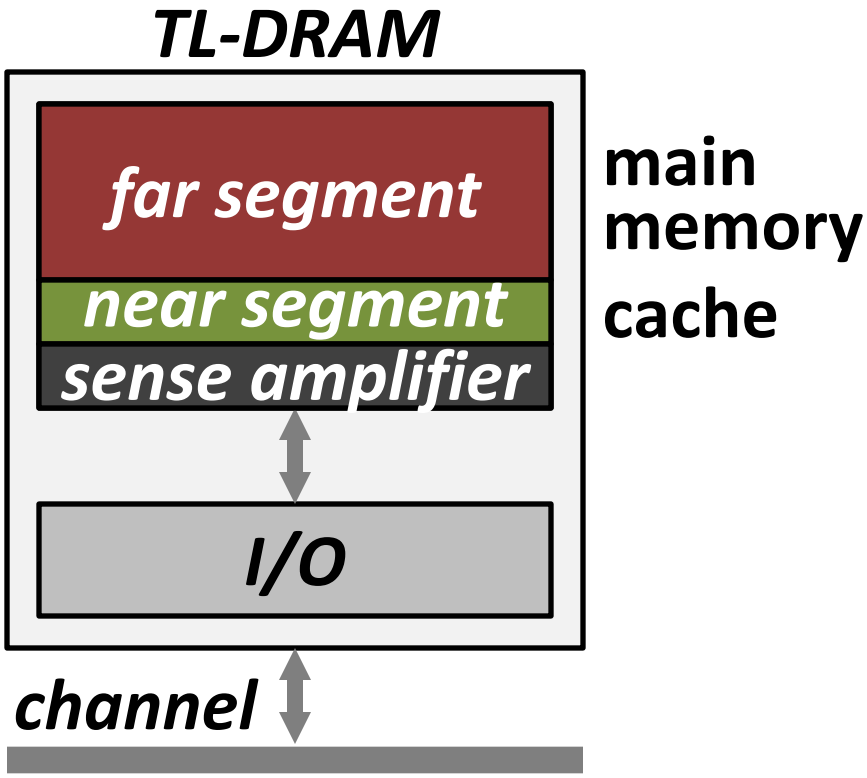


Step 2: Activate destination row to connect cell and bitline

Near Segment

Sense Amplifier

Near Segment as Hardware-Managed Cache



- **Challenge 1:** How to efficiently migrate a row between segments?
- **Challenge 2:** How to efficiently manage the cache?

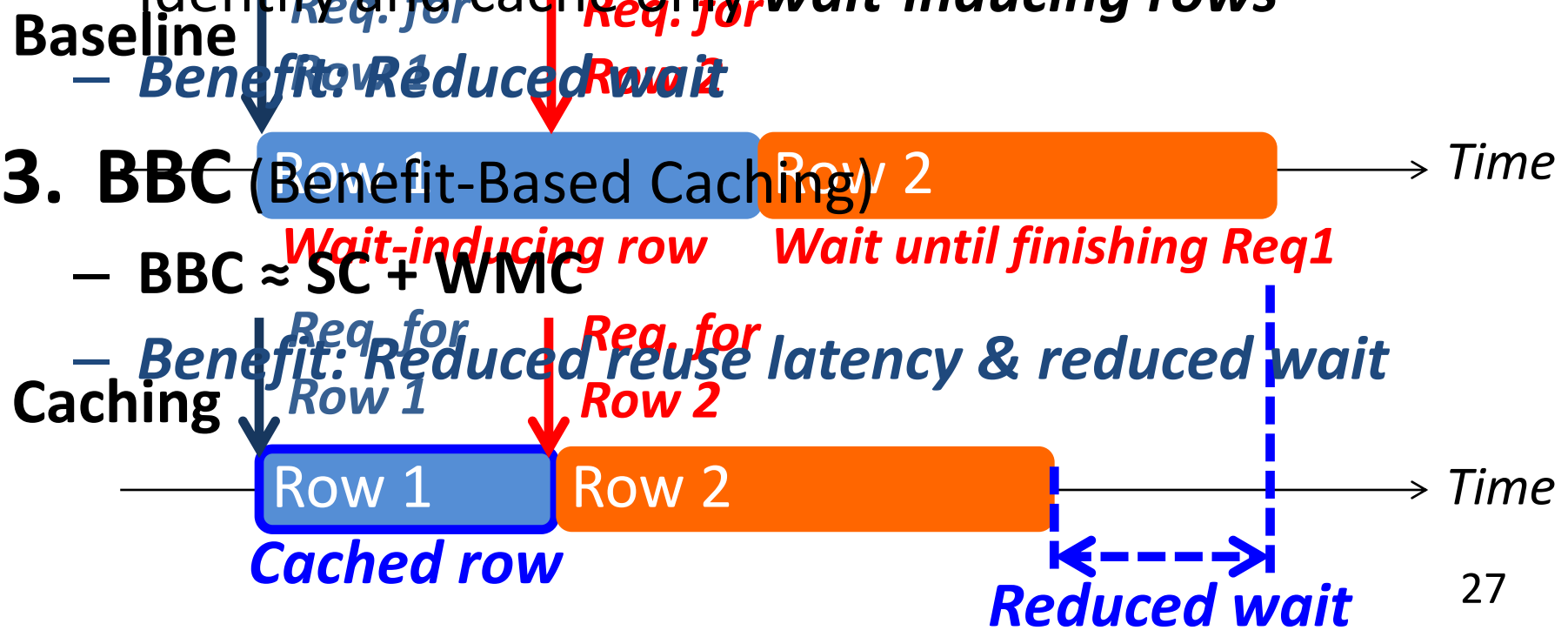
Three Caching Mechanisms

1. **SC** (Simple Caching)
   - Classic LRU cache
   - ***Benefit: Reduced reuse latency***

Is there another benefit of caching?

   - Identify and cache only ***wait-inducing rows***
   - ***Benefit: Reduced wait***

3. **BBC** (Benefit-Based Caching)
   - **BBC ≈ SC + WMC**
   - ***Benefit: Reduced reuse latency & reduced wait***



Outline

- Motivation & Key Idea
- Tiered-Latency DRAM
- Leveraging Tiered-Latency DRAM
- **Evaluation Results**

Evaluation Methodology

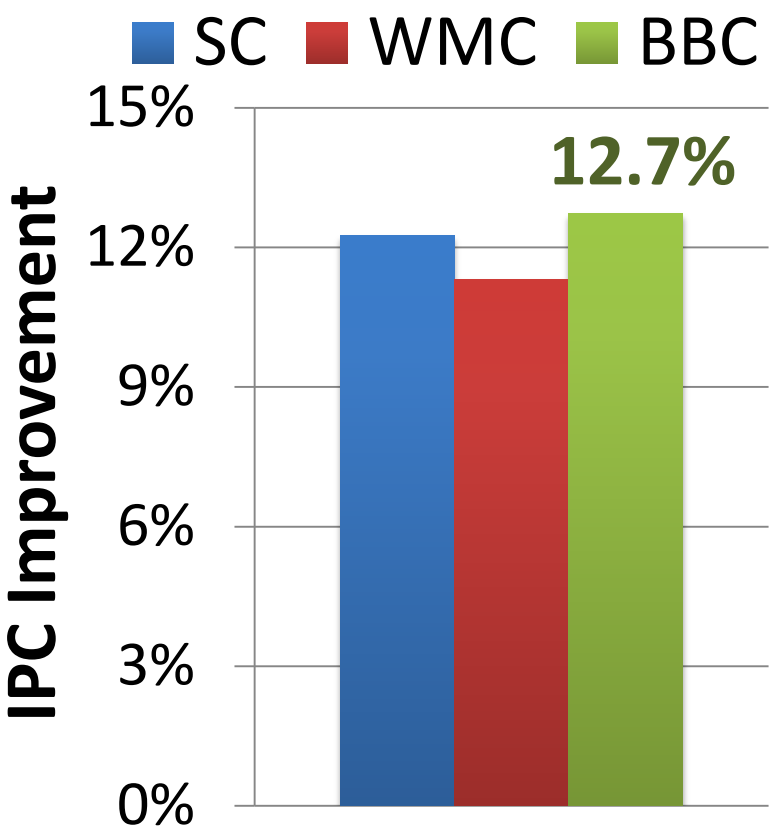
- **System simulator**
  - CPU: Instruction-trace-based x86 simulator
  - Memory: Cycle-accurate DDR3 DRAM simulator
- **Workloads**
  - 32 Benchmarks from TPC, STREAM, SPEC CPU2006
- **Metrics**
  - Single-core: Instructions-Per-Cycle
  - Multi-core: Weighted speedup

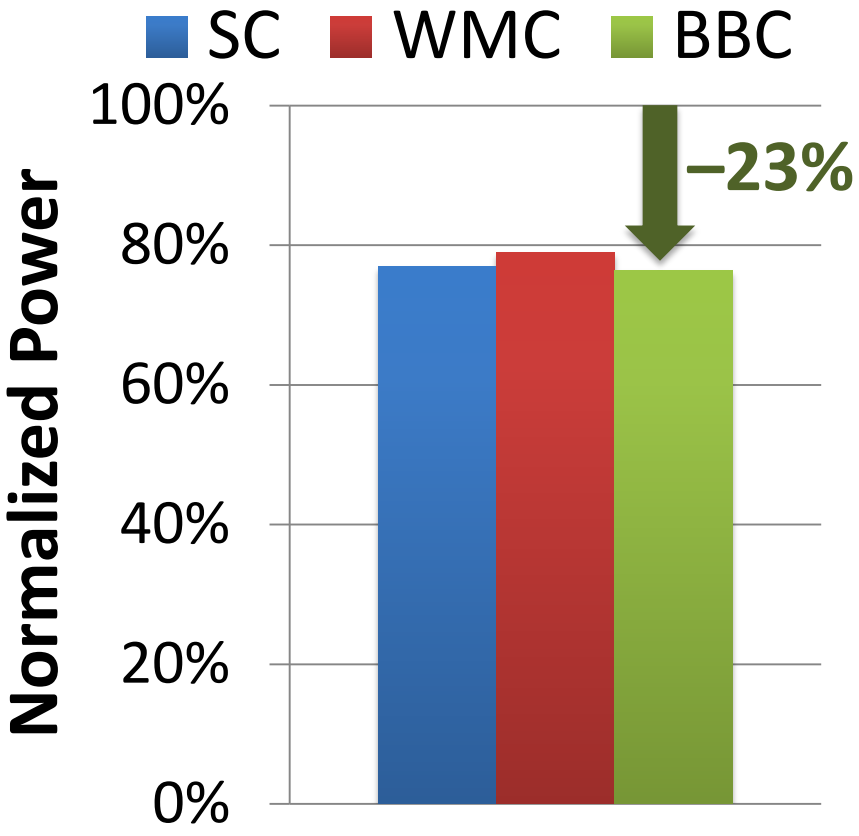
Configurations

- **System configuration**
  - CPU: 5.3GHz
  - LLC: 512kB private per core
  - **Memory: DDR3-1066**
    - 1-2 channel, 1 rank/channel
    - 8 banks, 32 subarrays/bank, **512 cells/bitline**
    - Row-interleaved mapping & closed-row policy

- **TL-DRAM configuration**
  - Total bitline length: **512 cells/bitline**
  - Near segment length: 1-256 cells

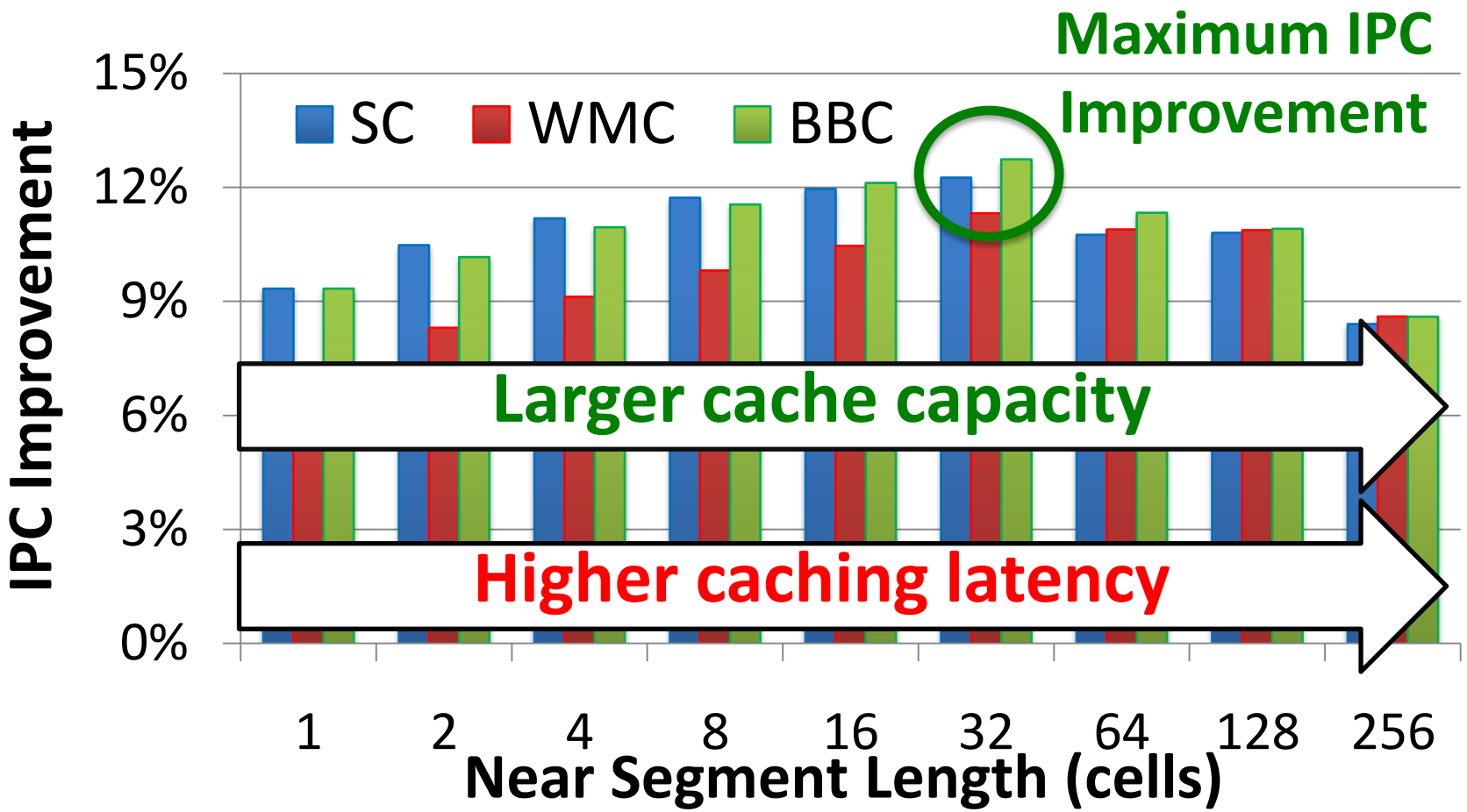
Single-Core: Performance & Power





Using near segment as a cache improves performance and reduces power consumption

Single-Core: Varying Near Segment Length



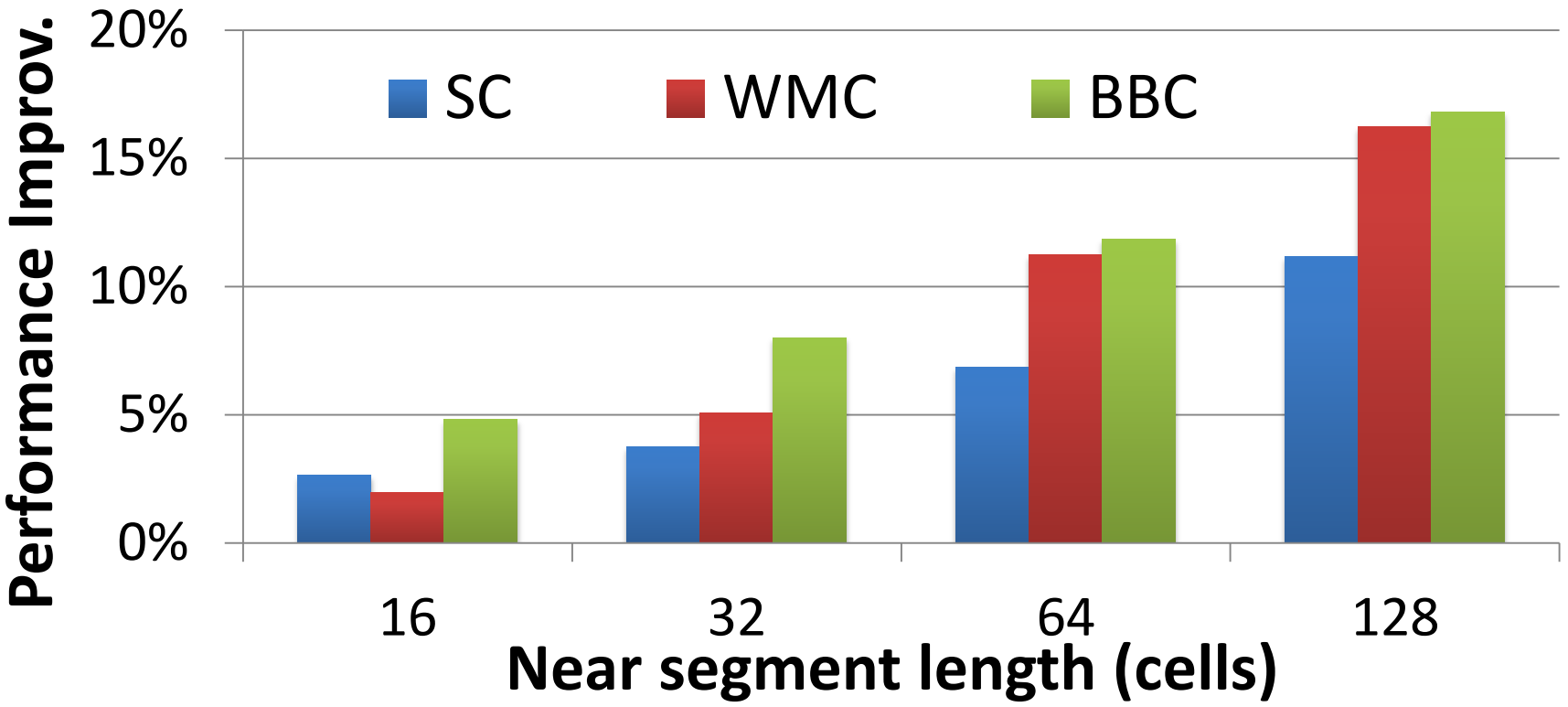
By adjusting the near segment length, we can trade off cache capacity for cache latency

Dual-Core Evaluation

- We categorize single-core benchmarks into two categories
  1. **Sens**: benchmarks whose performance is ***sensitive*** to near segment capacity
  2. **Insens**: benchmarks whose performance is ***insensitive*** to near segment capacity

- Dual-core workload categorization
  1. **Sens/Sens**
  2. **Sens/Insens**
  3. **Insens/Insens**

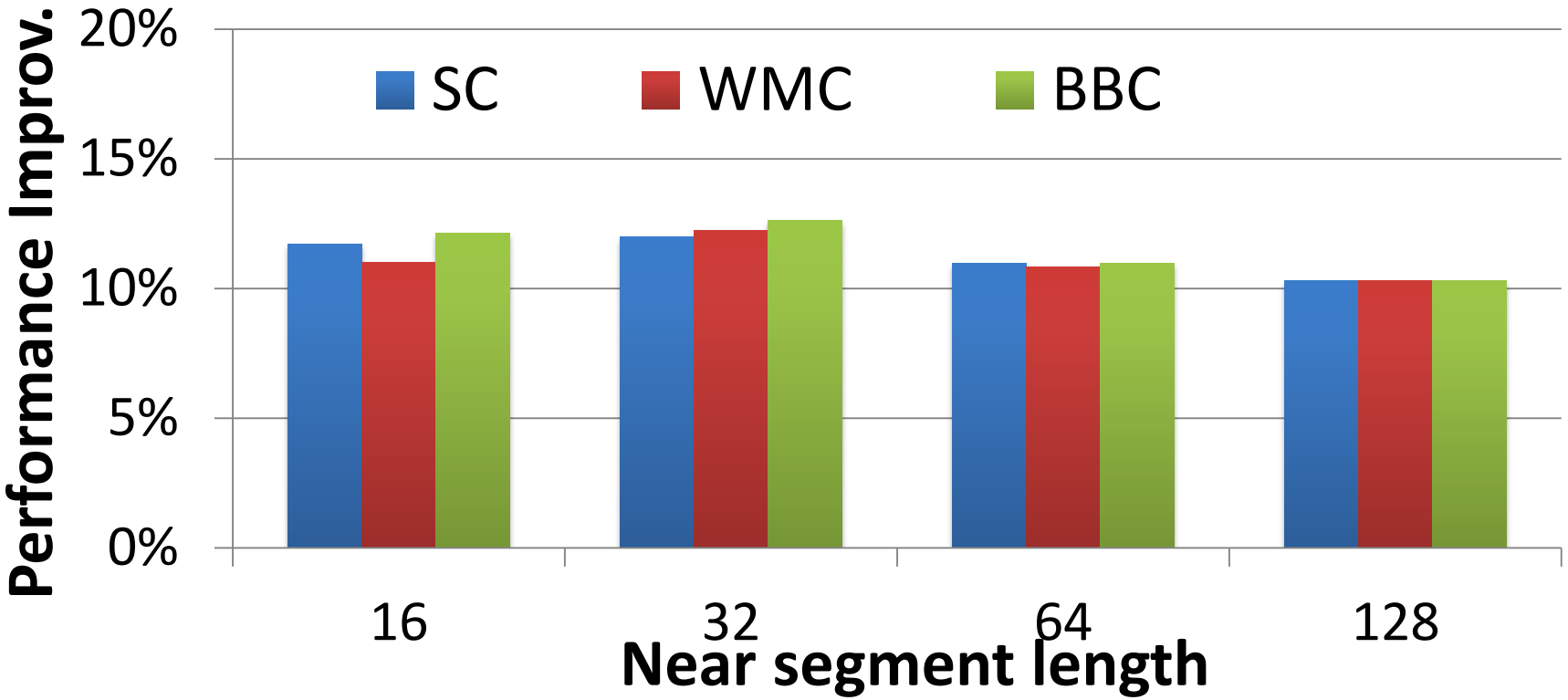
Dual-Core: Sens/Sens



Larger near segment capacity leads to higher performance improvement in sensitive workloads

BBC/WMC show more perf. improvement

Dual-Core: Sens/Insens & Insens/Insens



Using near segment as a cache provides high performance improvement regardless of near segment capacity

Other Mechanisms & Results in Paper

- **More mechanisms** for leveraging TL-DRAM
  - Hardware-managed ***exclusive*** caching mechanism
  - Profile-based page mapping to near segment
  - TL-DRAM improves performance and reduces power consumption with other mechanisms
- **More than two tiers**
  - Latency evaluation for three-tier TL-DRAM
- **Detailed circuit evaluation** for DRAM latency and power consumption
  - Examination of tRC and tRCD
- **Implementation details** and **storage cost analysis** in memory controller

Conclusion

- **Problem: DRAM latency is a critical performance bottleneck**
- **Our Goal**: Reduce DRAM latency with low area cost
- **Observation**: Long bitlines in DRAM are the dominant source of DRAM latency
- **Key Idea: Divide long bitlines into two shorter segments**
  - **Fast and slow segments**
- **Tiered-latency DRAM: Enables latency heterogeneity in DRAM**
  - **Can leverage this in many ways to improve performance and reduce power consumption**
- **Results**: When the fast segment is used as a cache to the slow segment → Significant performance improvement (>12%) and power reduction (>23%) at low area cost (3%)

Thank You

Tiered-Latency DRAM: A Low Latency and A Low Cost DRAM Architecture

Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, Onur Mutlu

Carnegie Mellon

SAFARI

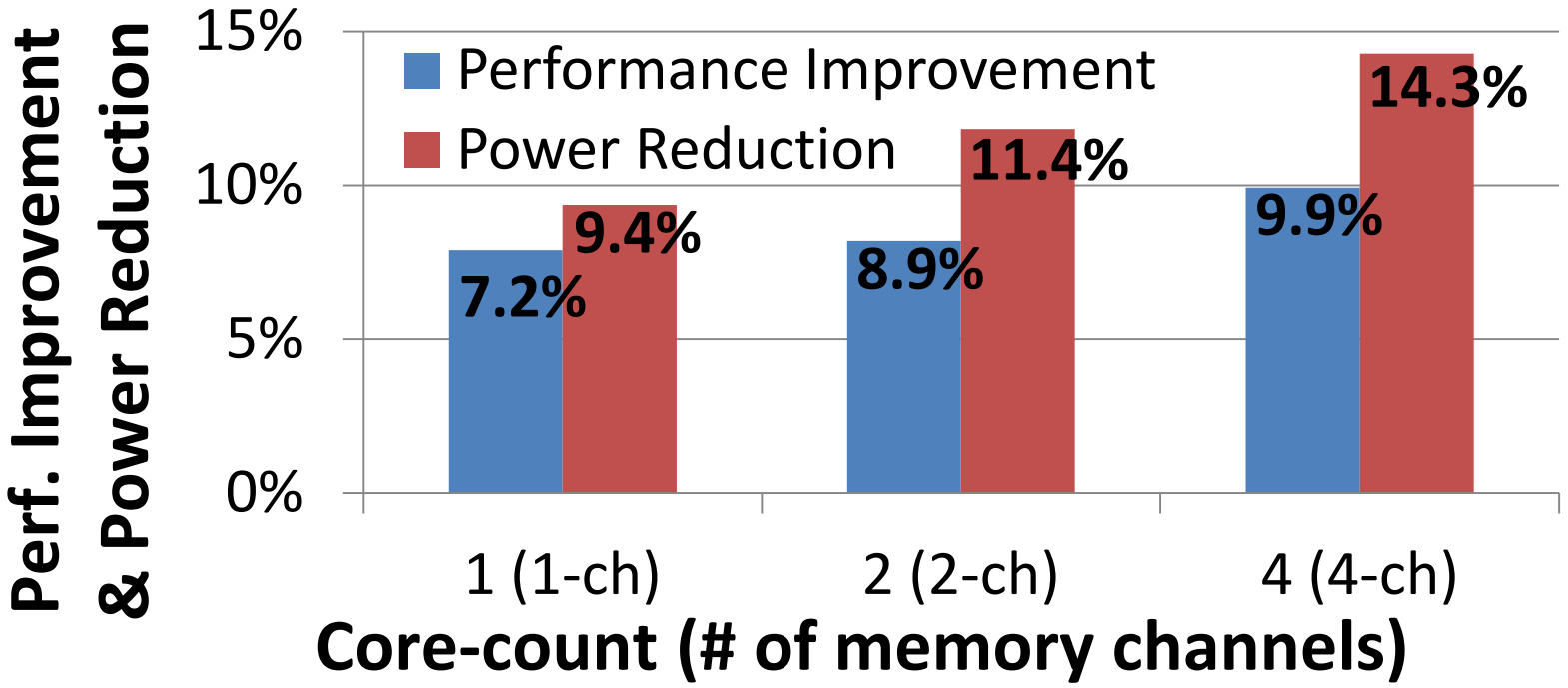
Backup Slides

Storage Cost in Memory Controller

- **Organization**
  - Bitline Length: 512 cells/bitline
  - Near Segment Length: 32 cells
  - Far Segment Length: 480 cells
  - **Inclusive Caching**
- **Simple caching** and **wait-minimized caching**
  - Tag Storage: 9 KB
  - Replace Information: 5 KB
- **Benefit-based caching**
  - Tag storage: 9 KB
  - Replace Information: 8 KB
    (8 bit benefit field/near segment row)

Hardware-managed Exclusive Cache

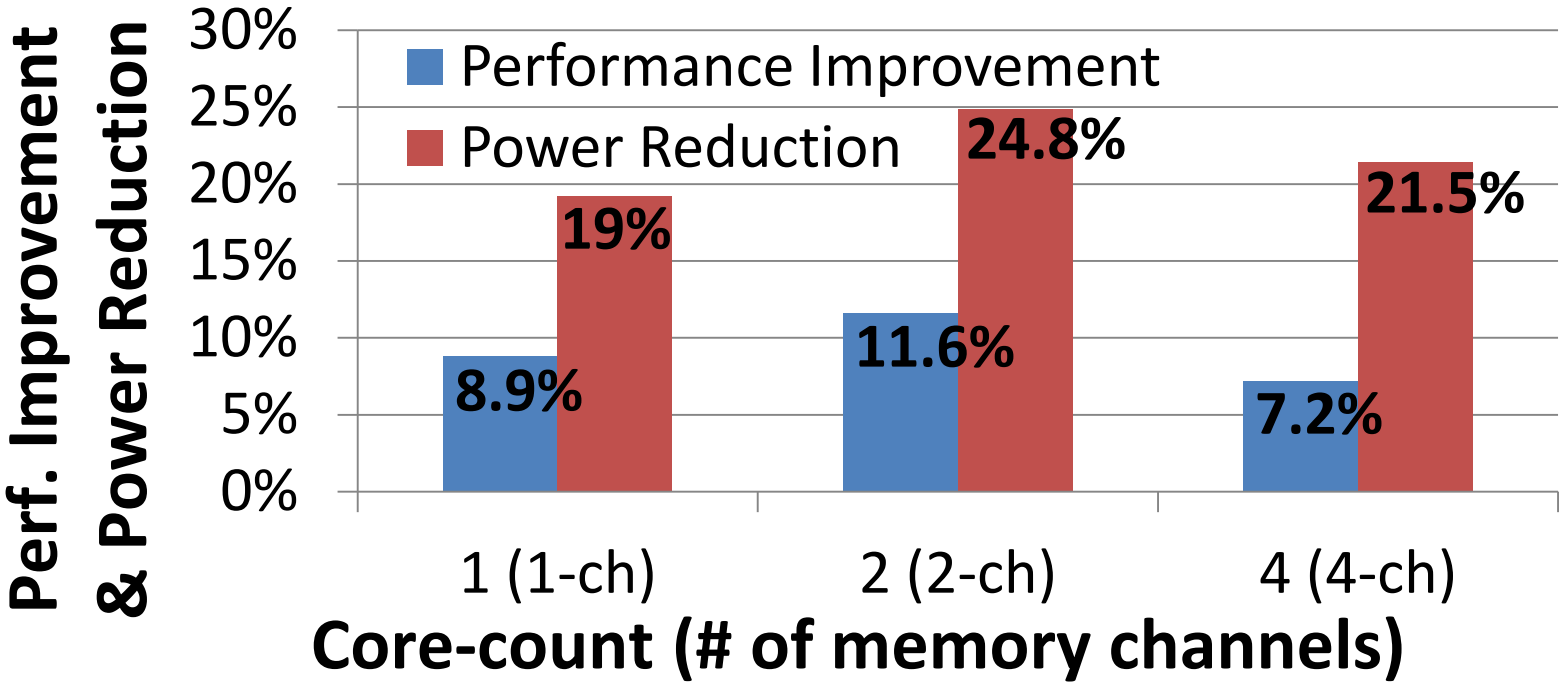
- Near and Far segment: Main memory
- Caching: Swapping near and far segment row
  - Need one dummy row to swap



Performance improvement is lower than Inclusive caching due to high swapping latency

Profile-Based Page Mapping

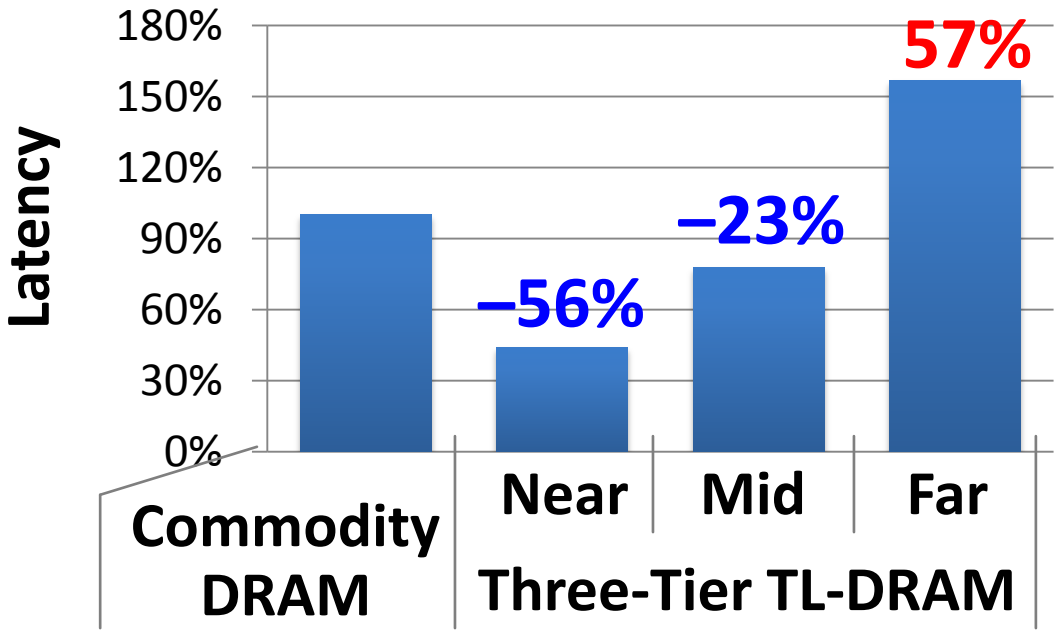
- Operating system profiles applications and maps frequently accessed rows to the near segment



Allocating frequently accessed rows in the near segment provides performance improvement

Three-Tier Analysis

- **Three tiers**
  - Add two isolation transistors
  - Near/Mid/Far segment length: 32/224/256 Cells



More tiers enable finer-grained caching and partitioning mechanisms