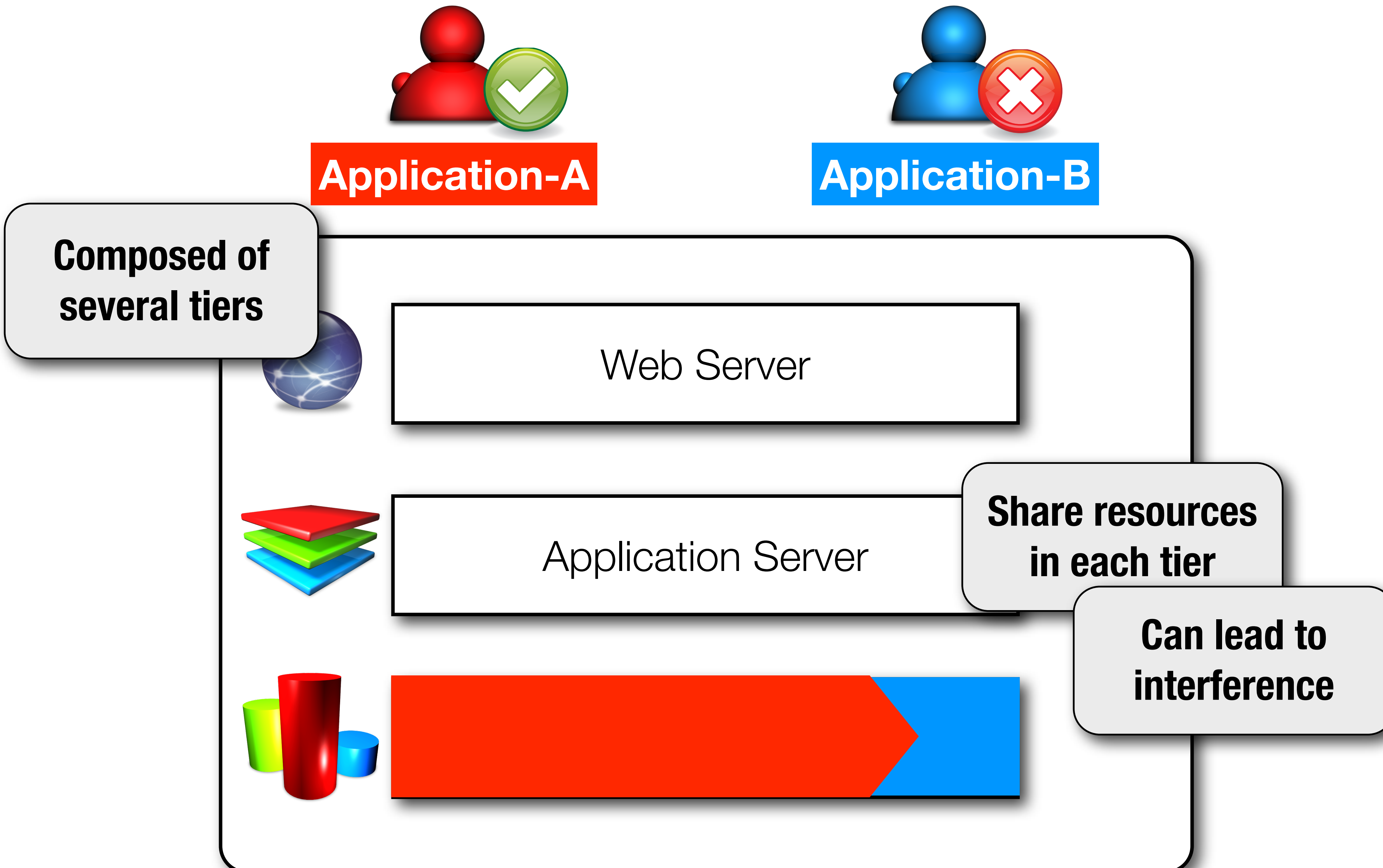
Dynamic Resource Allocation for Database Servers Running on Virtual Storage

Gokul Soundararajan, Daniel Lupei, Saeed Ghanbari, Adrian Daniel Popescu, Jin Chen, Cristiana Amza

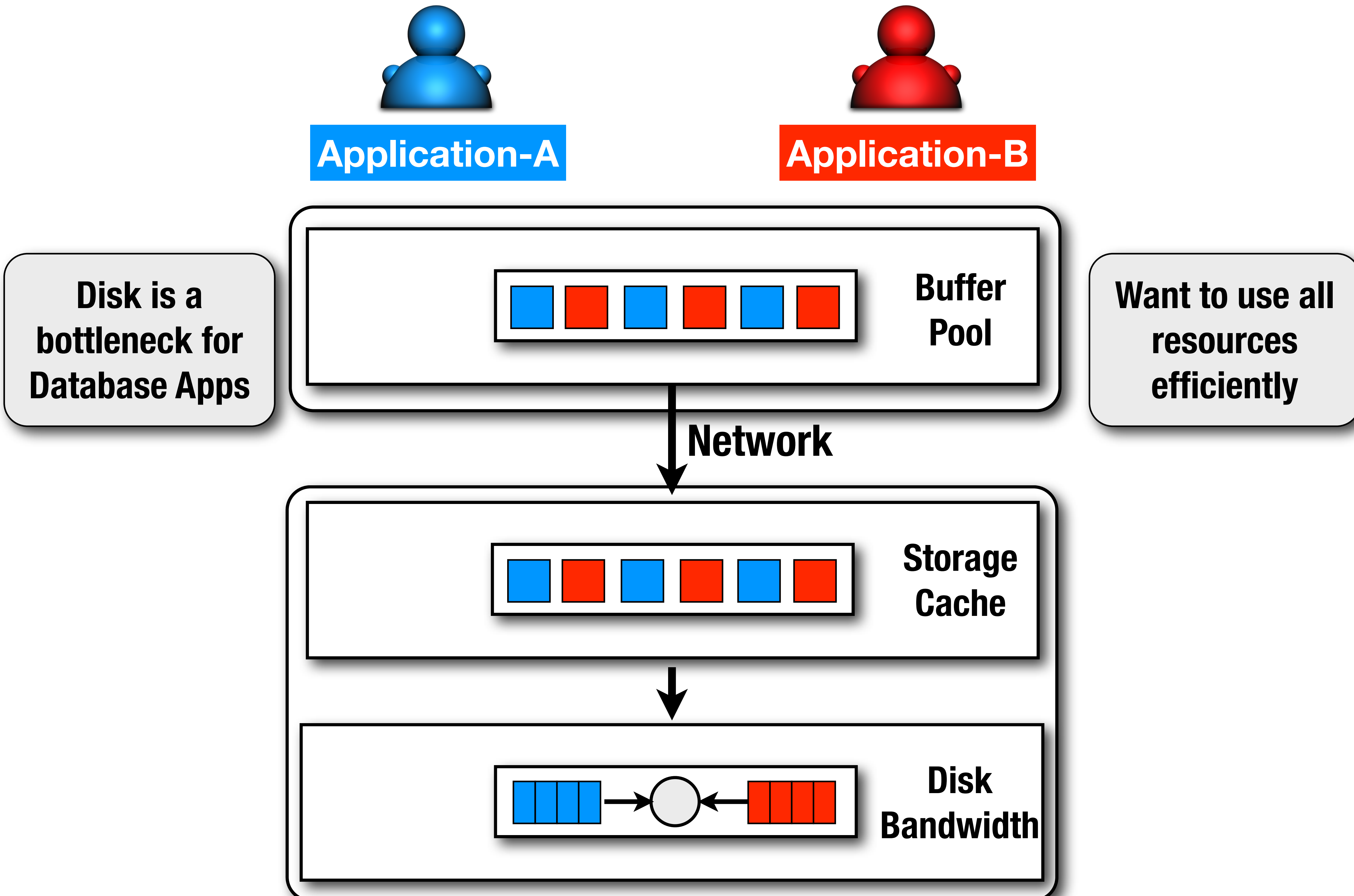
University of Toronto

Multi-tier Resource Allocation



Consolidated Environment

Our Focus: Storage Hierarchy

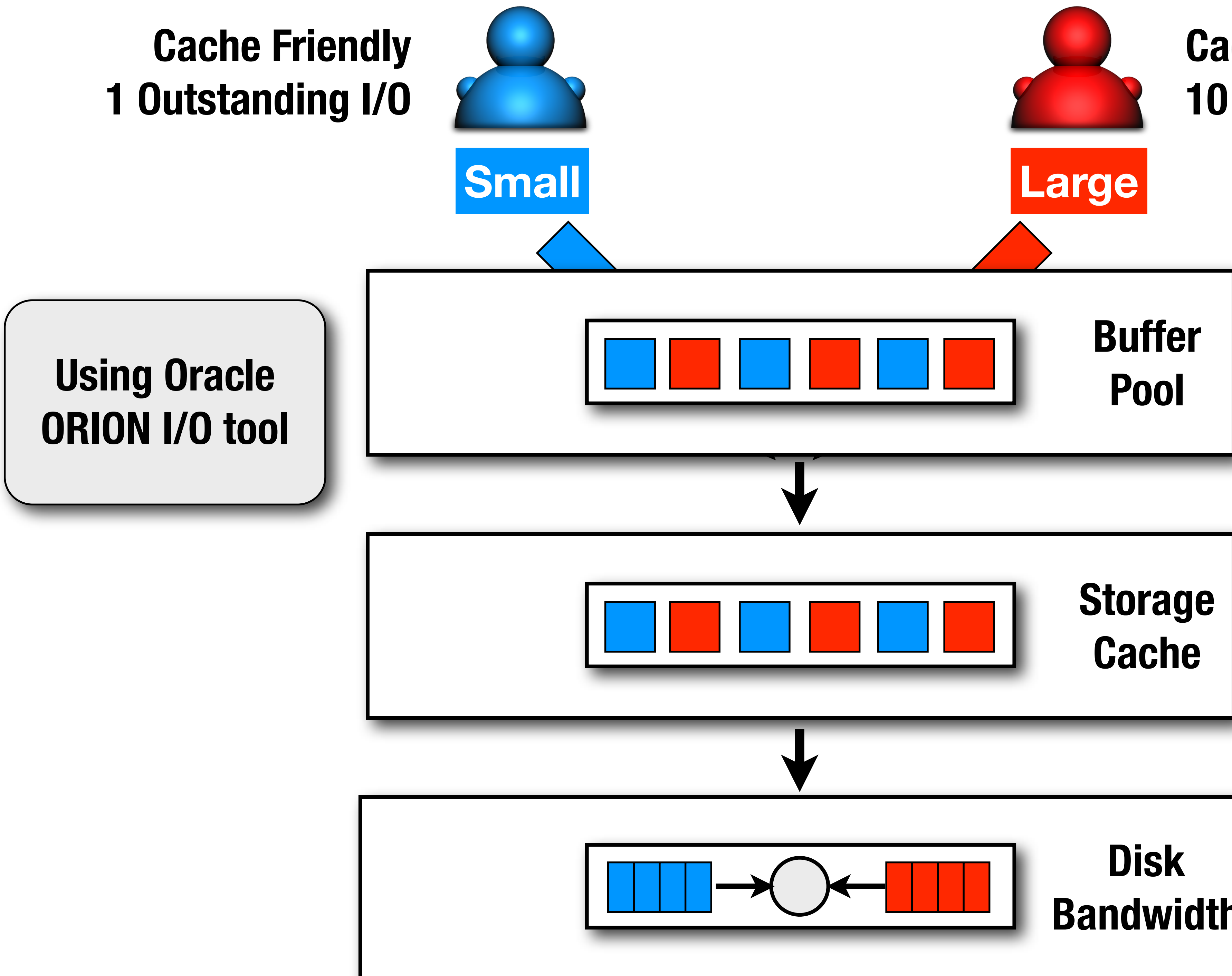


State of the Art

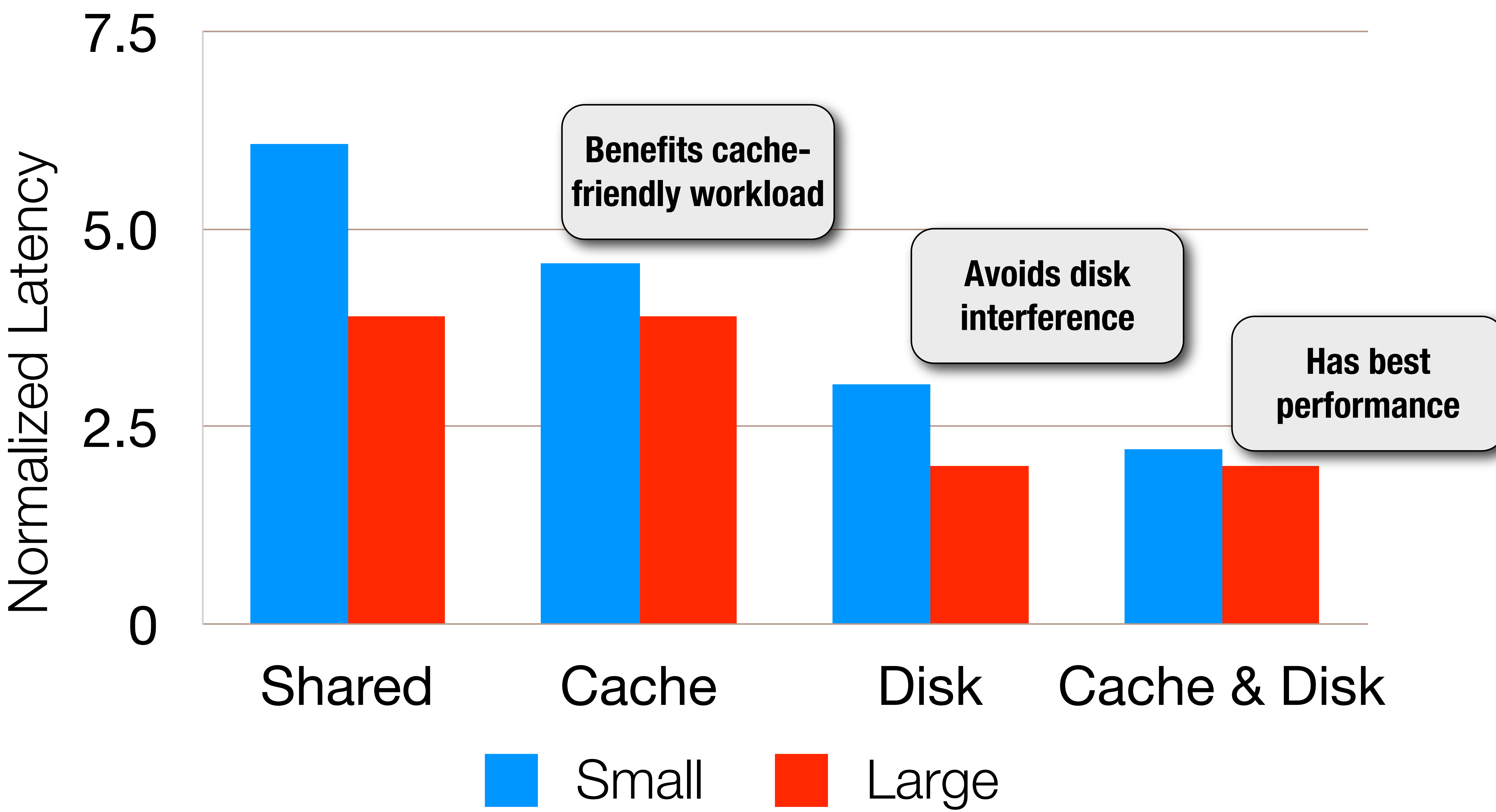
- **Previous work studied resources in isolation**
  - *Memory Partitioning:* MRC [ASPLOS'04]
  - *Disk Bandwidth*: Facade [FAST'03], Argon [FAST'07], etc.
  - *... and many more*

- **Want to use the storage hierarchy efficiently**
- **However, performance depends on all layers**
  - Interdependency between resources
  - E.g., Increasing buffer pool *reduces* number of storage accesses

Motivating Scenario



Motivating Scenario



Contributions

- **Build performance models dynamically**
  - Account for interdependencies between resources
  - Lightweight but still accurate

- **Multi-level Resource Allocator**
  - Uses performance models to guide resource allocation
  - Corrects model errors through runtime sampling
  - Uses global utility (SLOs) to partition resources
    - Minimize sum of application latencies

Approach

- **Build performance models**
  - One per application
  - Derive function to predict application latency given configuration

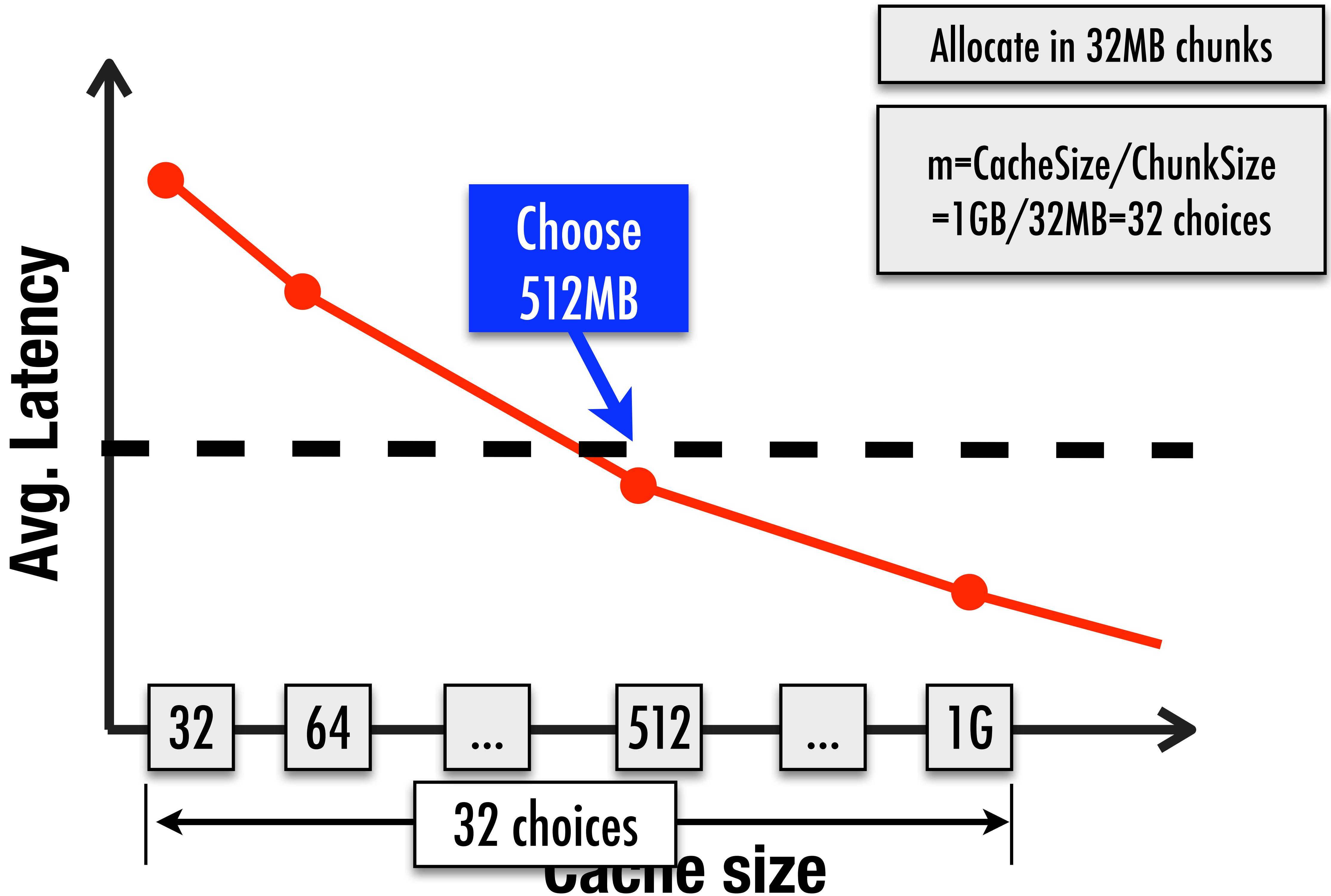
$$L_{avg} = f(\rho_c, \rho_s, \rho_d)$$

- **Find resource partitioning setting**
  - Minimize sum of application latencies
  - Find best setting using hill climbing

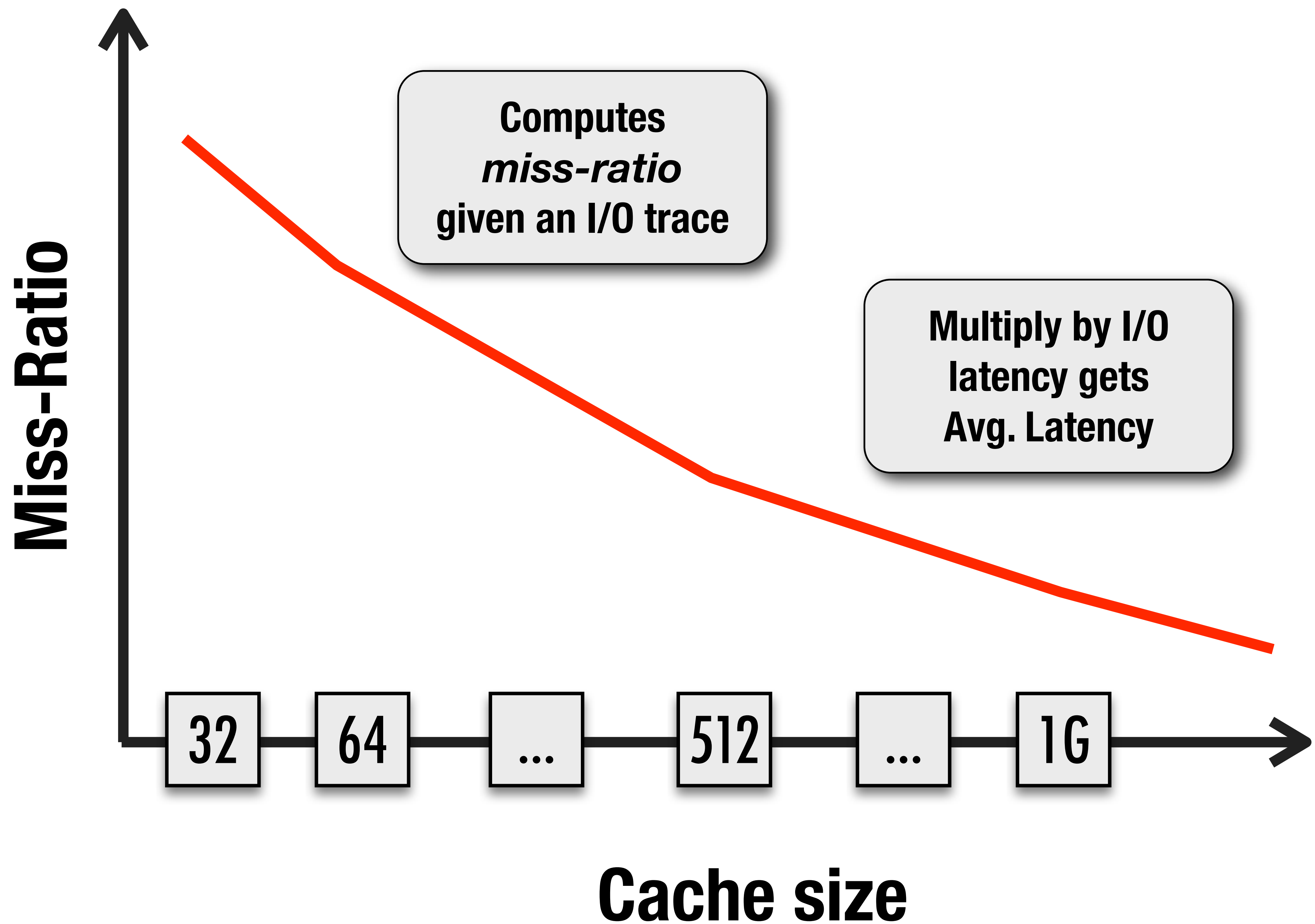
Outline

- **Online Performance Models**
  - What are they?
  - Why are they hard to build?
- **Multi-level Resource Allocator**
- **Prototype Implementation**
- **Experimental Results**
- **Conclusions**

One-Level Cache Model



MRC Cache Model



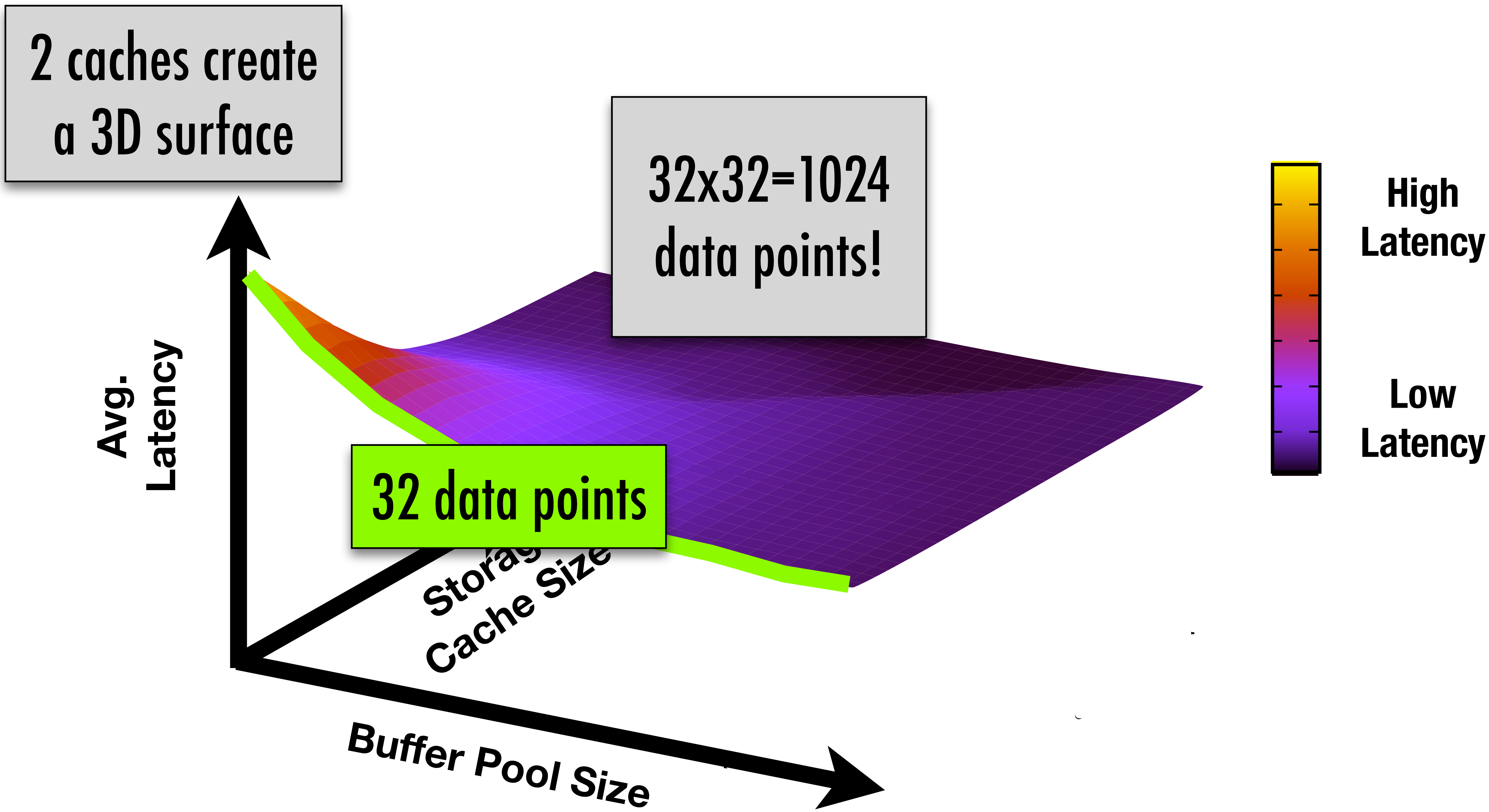
Two-Level Cache Model

- **Performance affected by**
  - DB Buffer Pool Size (**m** choices)
  - Storage Cache (**n** choices)

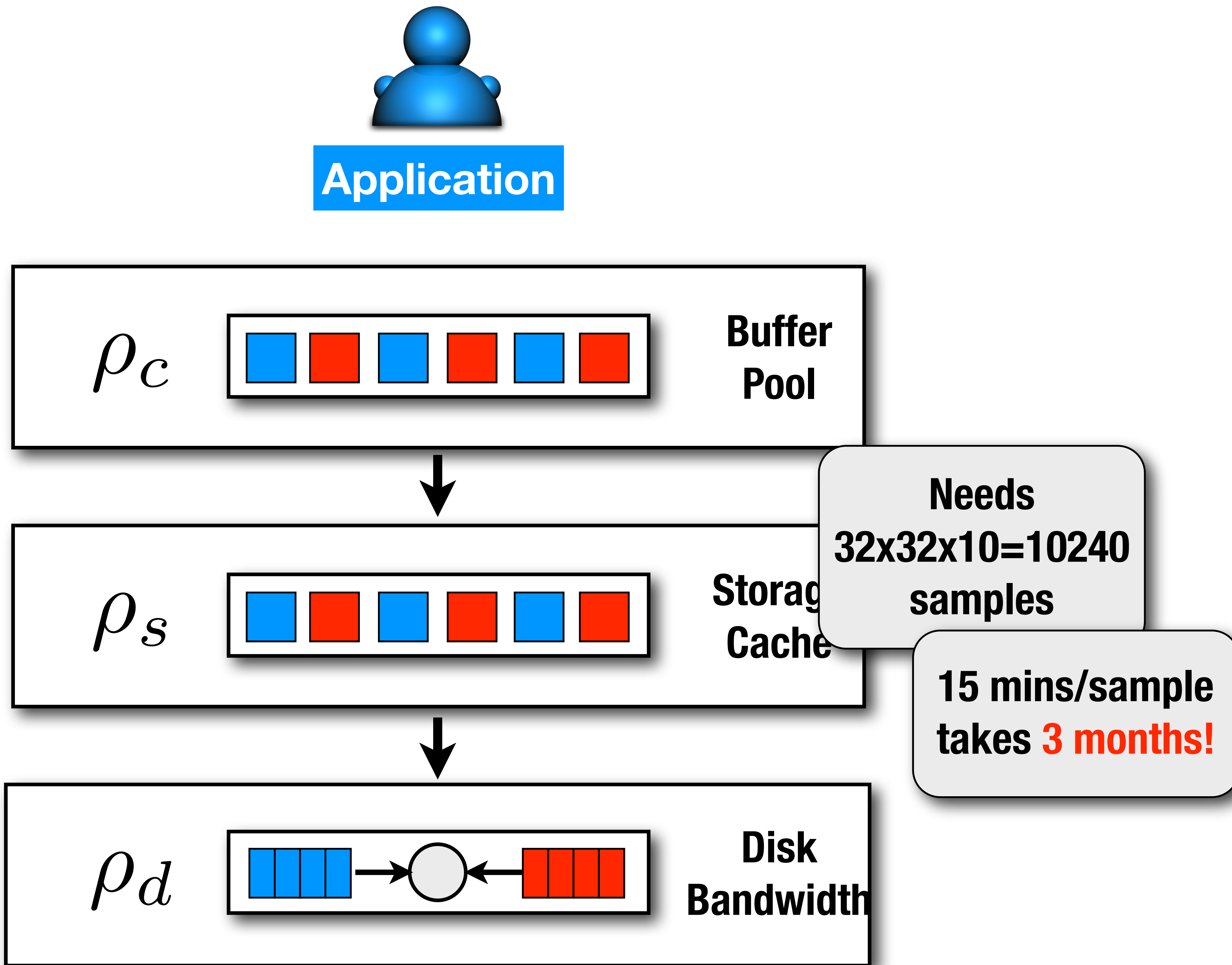
Changes the I/O trace at storage

- **Performance model**
  - Needs to consider all parameters (**m*n** choices)
  - 1GB caches allocated in 32MB chunks
    - **m** = 1GB/32MB = 32 settings
    - **m*n** = 1024 distinct settings

Two-Level Cache Model



Overall Performance Model



Outline

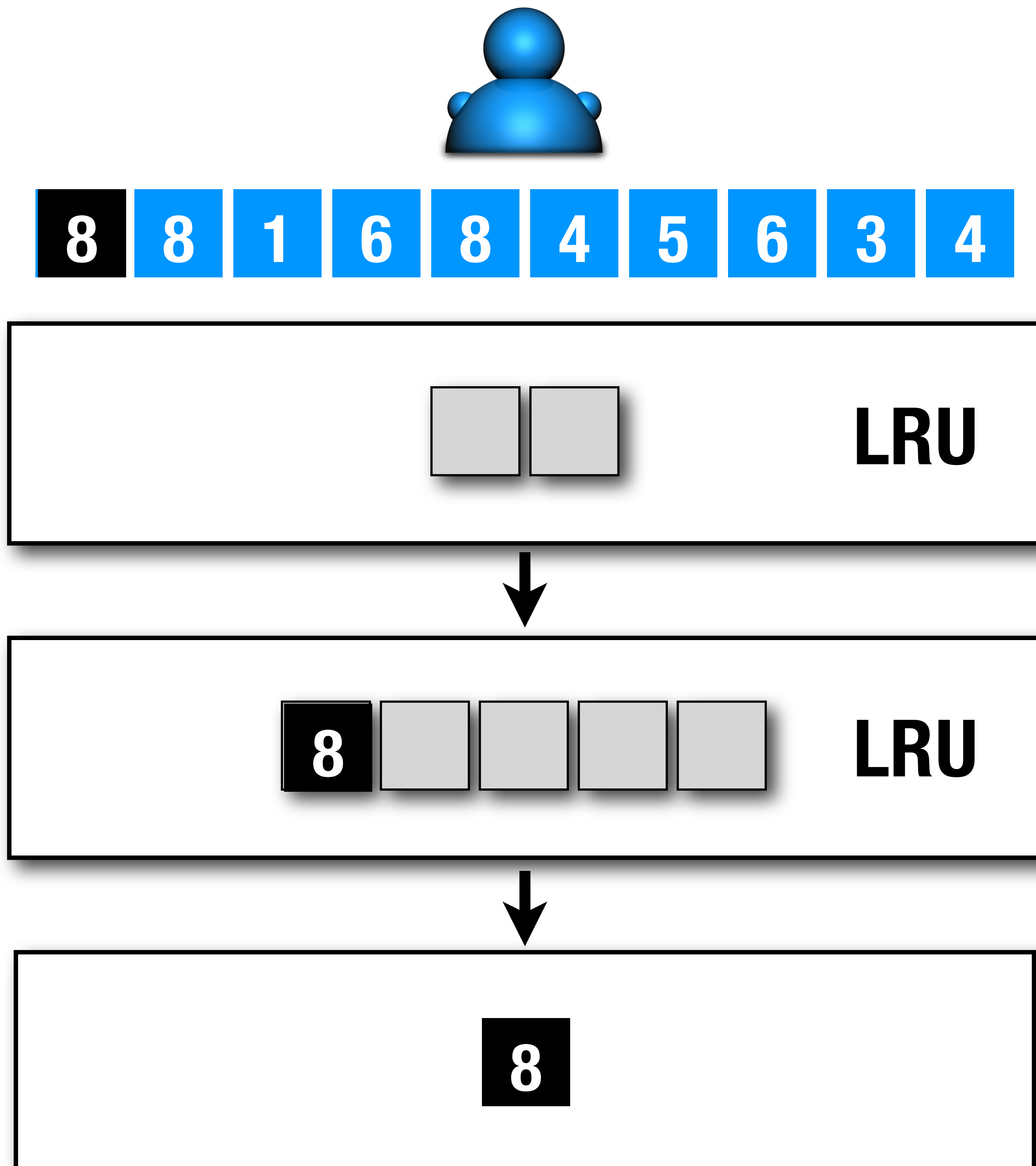
- **Online Performance Models**
- **Multi-level Resource Allocator**
  - Building performance models
  - Allocating resources using models
- **Prototype Implementation**
- **Experimental Results**
- **Conclusions**

Key Observations

- **Known cache replacement policies**
  - Most cache replacement algorithms are LRU
  - Only as effective as the largest cache (*cache inclusiveness*)

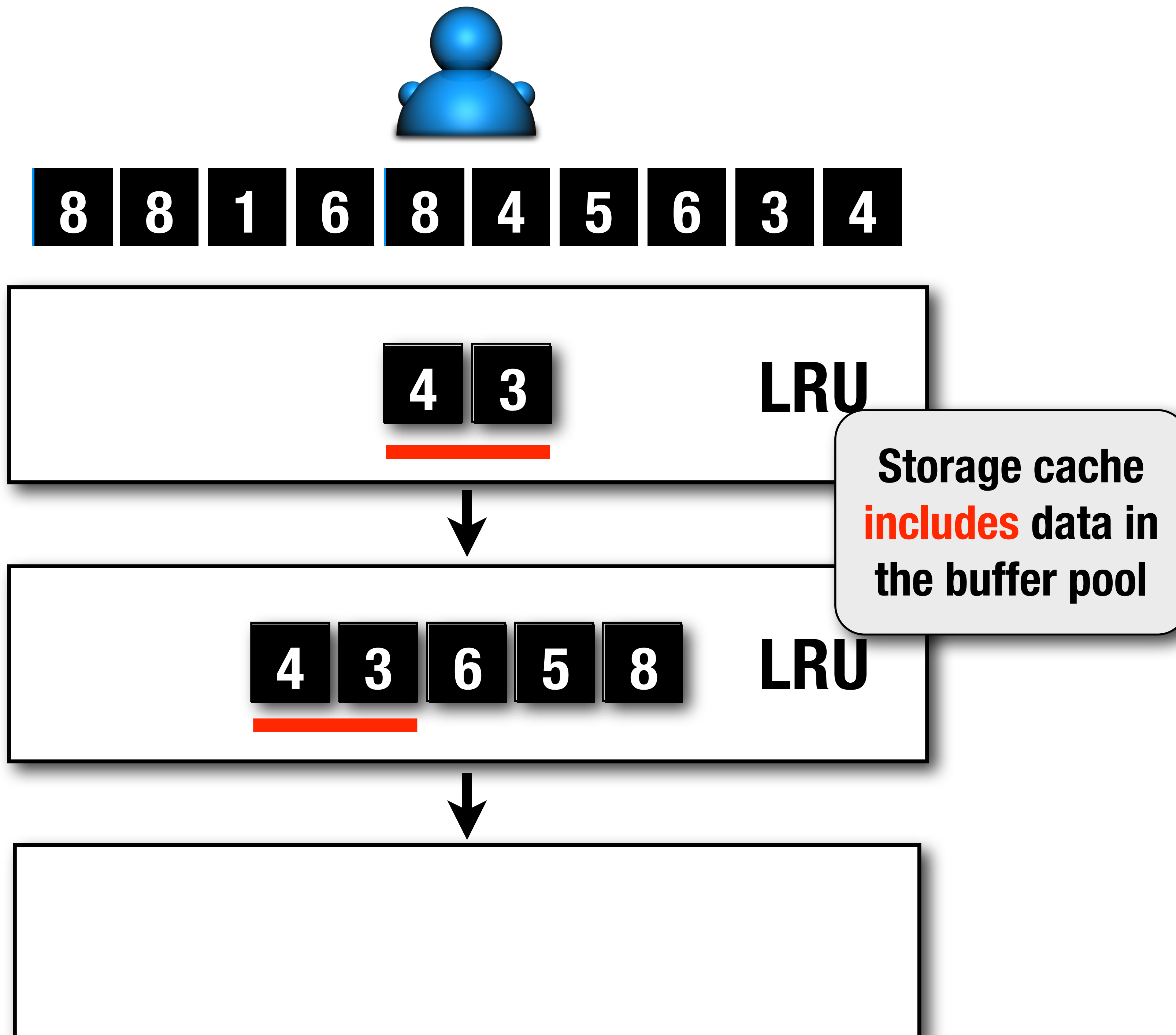
- **Disk is a closed loop system**
  - Rate of responses is same as rate of requests
  - Performance proportional to the disk bandwidth fraction

Cache Inclusiveness



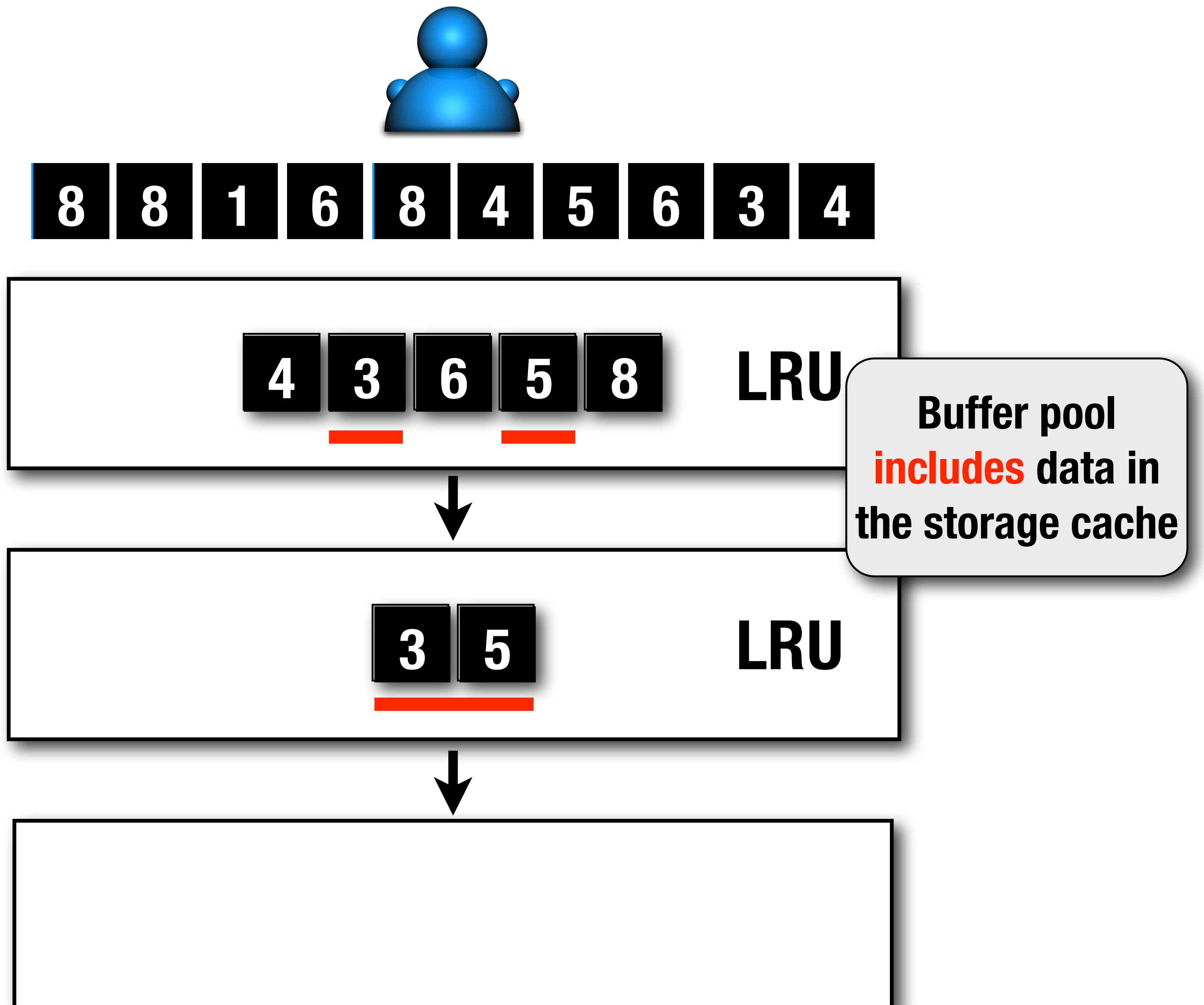
I/Os: 0

Cache Inclusiveness



I/Os: 6

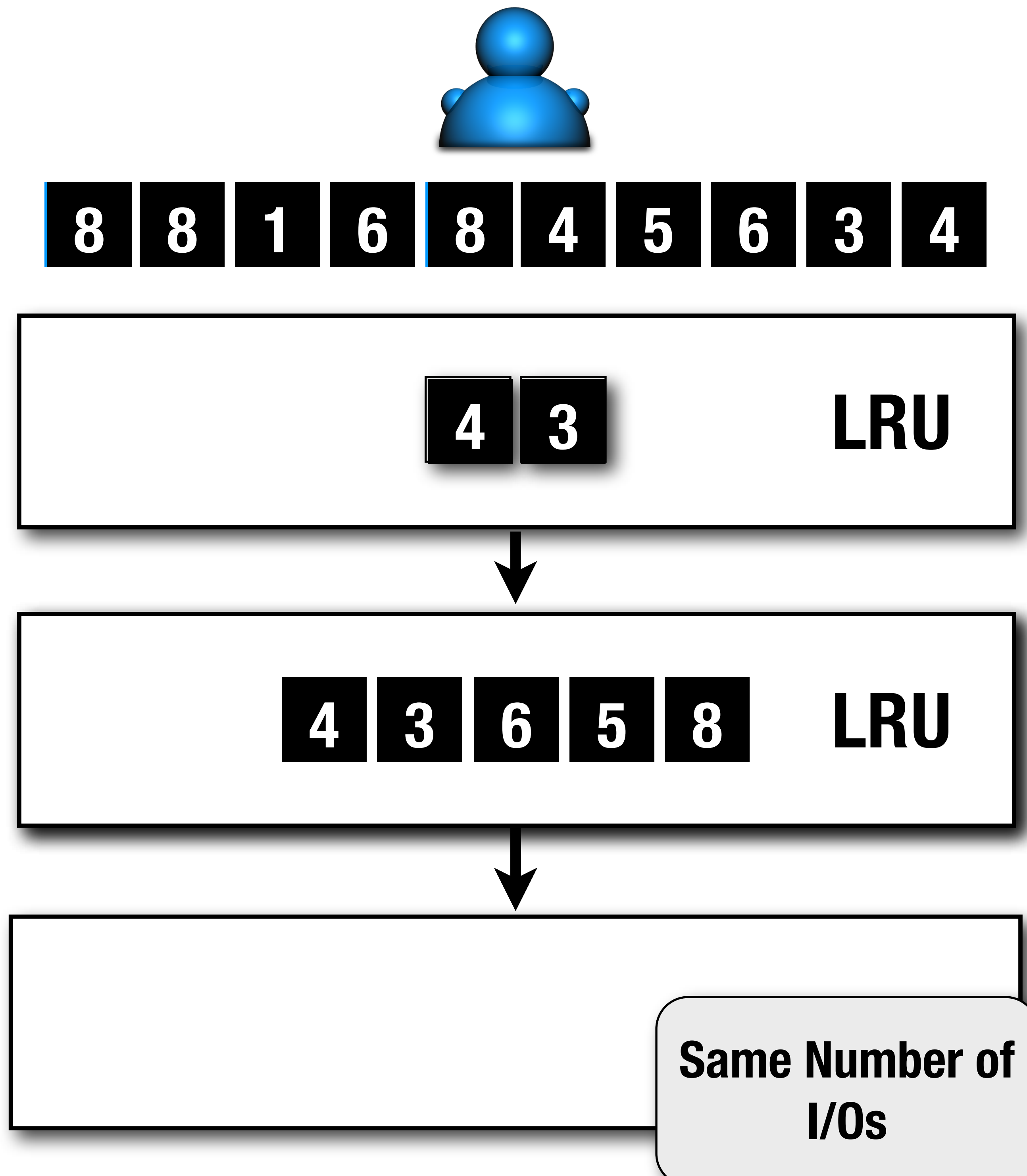
Cache Inclusiveness



I/Os: 6

Approximate Single Cache Model (LRU)

$$\mathcal{M}_c(\max[\rho_c, \rho_s])$$



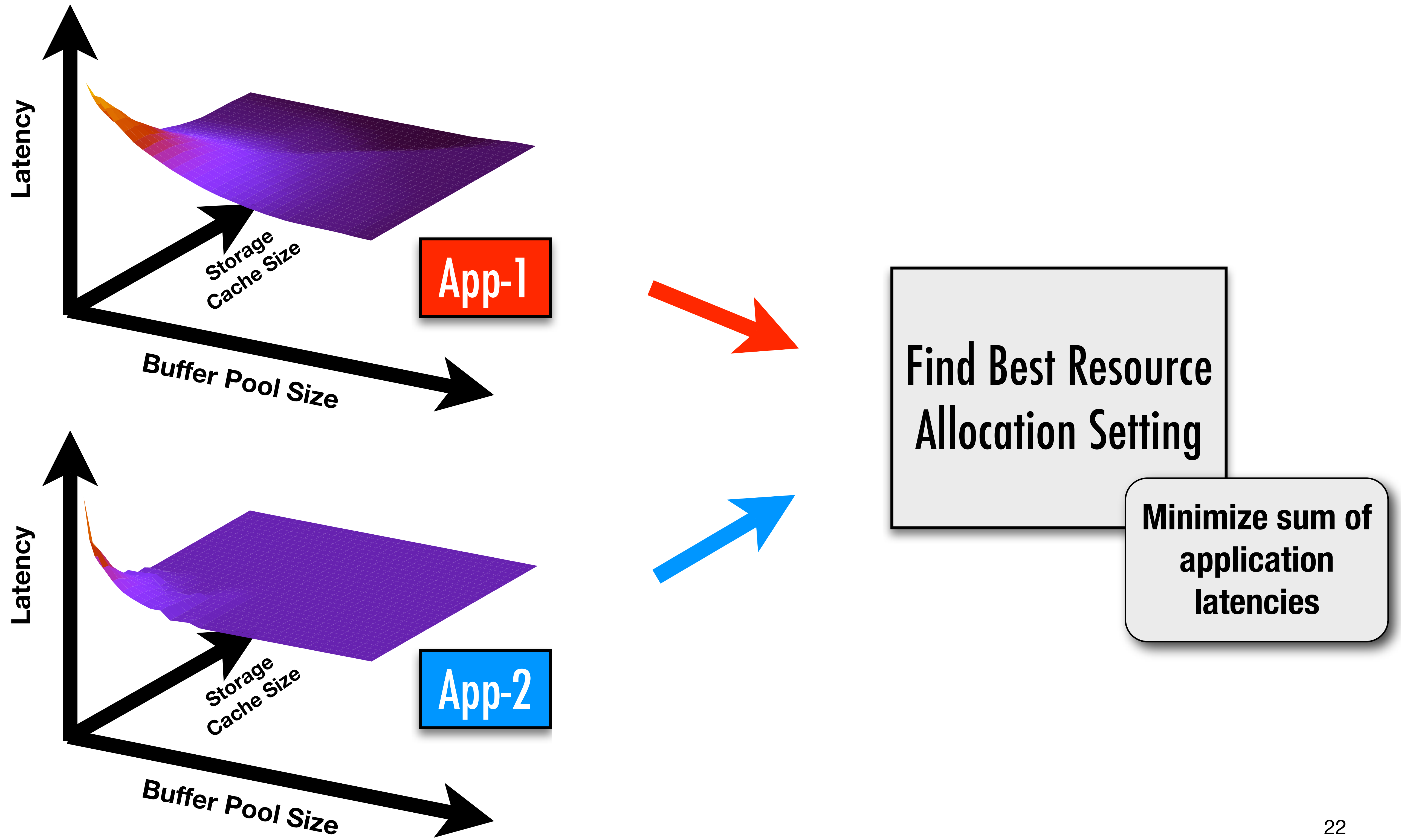
I/Os: 6

Cache Model (DEMOTE)

- **Maintain cache exclusiveness**
  - E.g., using DEMOTEs [USENIX'02]
  - Every block brought into *buffer pool* is not cached below
  - Only evictions from buffer pool cached in storage cache

- **Approximate performance using single cache**
  - $\mathcal{M}_c(\rho_c + \rho_s)$

Find Best Partitioning Setting



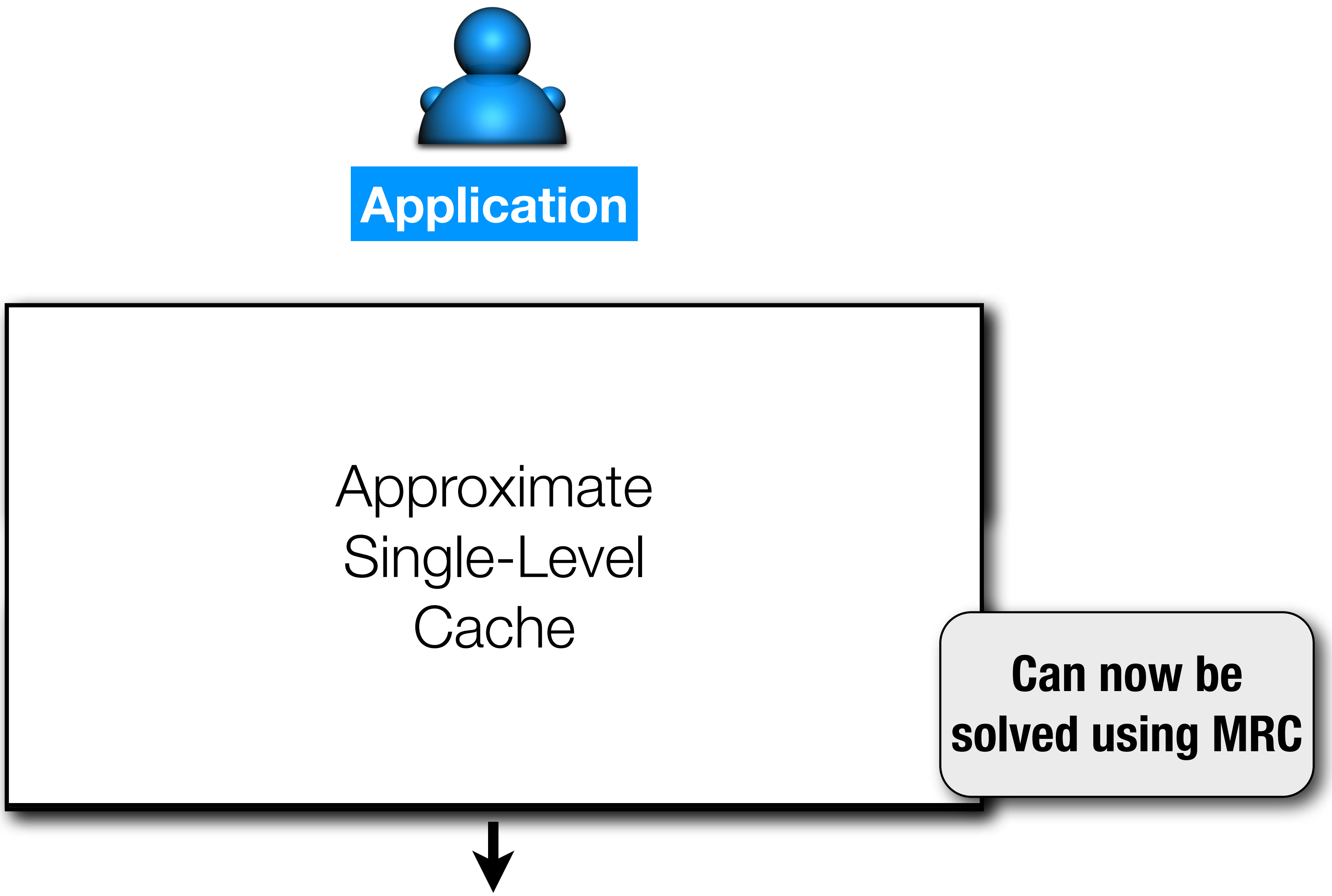
Disk Model

- **Observation: Closed loop system**
  - Rate of responses same as rate of requests
  - Use *interactive response time law*

- **Performance proportional to disk bandwidth fraction**
  - Measure base disk latency: $L_d(1)$
  - Predict latency for smaller bandwidth fractions

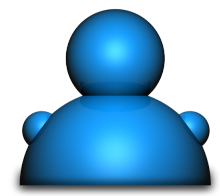
$$L_d(\rho_d) = \frac{L_d(1)}{\rho_d}$$

Putting it All Together



$$\underline{\mathcal{M}_c(\rho_c)\mathcal{M}_s(\rho_c, \rho_s)}N = \mathcal{M}_c(\max[\rho_c, \rho_s])N$$

Putting it all Together



Application

$$\mathcal{H}_c(\rho_c)L_c$$

↓

$$\mathcal{M}_c(\rho_c)\mathcal{H}_s(\rho_c,\rho_s)L_{net}$$

↓

$$\mathcal{M}_c(\rho_c)\mathcal{M}_s(\rho_c,\rho_s)L_d(\rho_d)$$

Inaccuracies in the Model

- **Cache Model**
  - Approximations to LRU, i.e., CLOCK
  - Large fraction of writes in the workload
- **Disk Model**
  - Using Quanta-based scheduler [Wachs et. al, FAST'07]
    - Interference due to disk seeks at small quanta

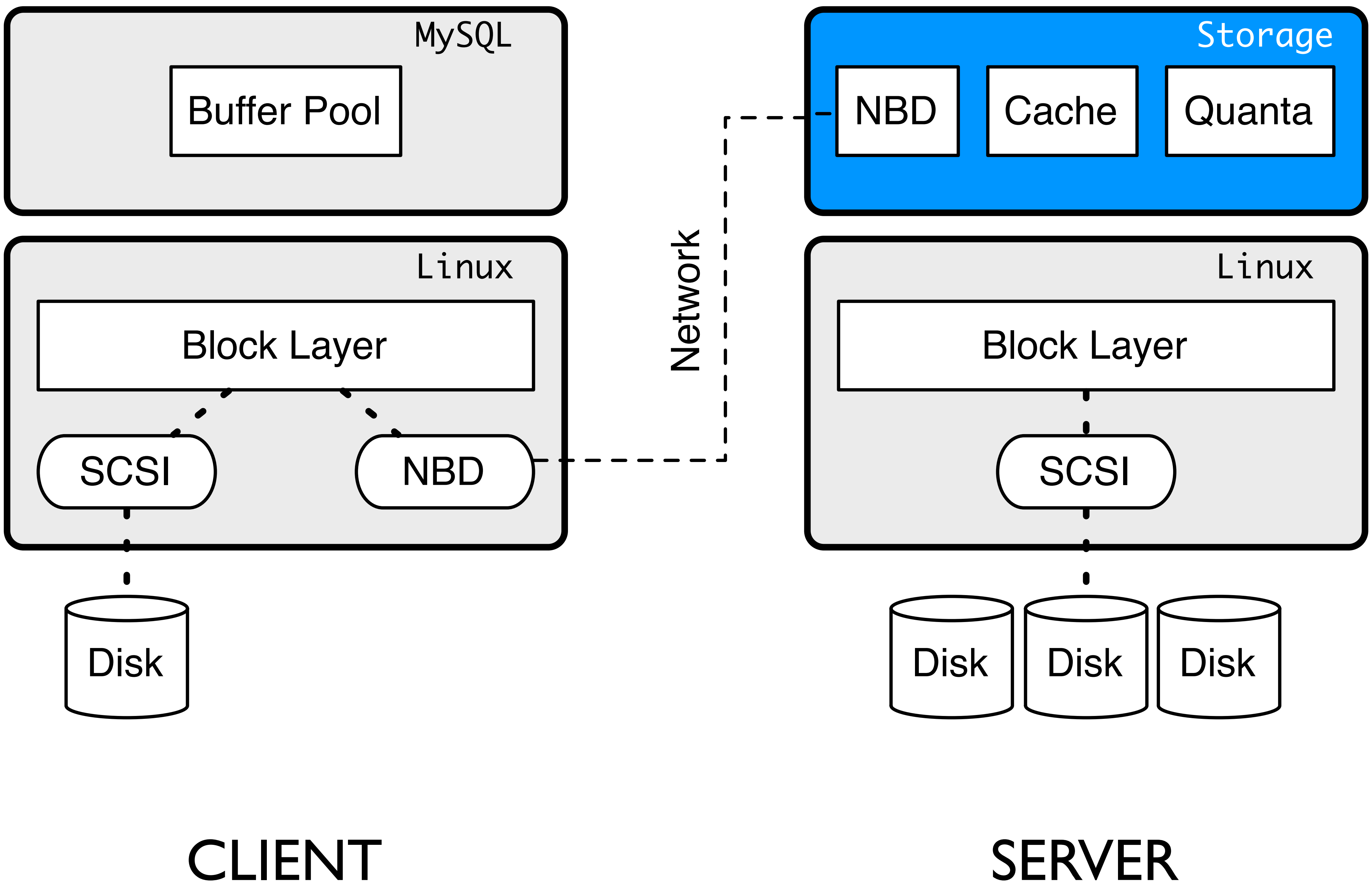
- **Inaccuracies localized in known regions**
  - E.g., Small disk quanta

Iterative Refinement

- **Build model**
  - Use trace collected at the database buffer pool

- **Refine the model**
  - Use cross-validation to measure quality
  - Selectively sample where error is high
  - Interpolate *computed* and *measured* samples
    - Using regression (SVM)

Virtual Storage Prototype



Experimental Setup

- **Benchmarks**
  - UNIFORM (*microbenchmark*), TPC-W and TPC-C
- **LAMP Architecture**
  - **L**inux, **A**pache 1.3, **M**ySQL/InnoDB 5.0, and **P**HP 5.0
- **Cache Configuration**
  - MySQL buffer pool = 1GB
  - Storage cache = 1GB
  - Using InnoDB cache replacement in MySQL, CLOCK in storage cache

Our Algorithms

- **GLOBAL**
  - Gather trace at the buffer pool
  - Measure base disk latency
  - Compute performance using performance model

- **GLOBAL+**
  - Run **GLOBAL**
  - Evaluate model accuracy
  - Refine model using runtime samples

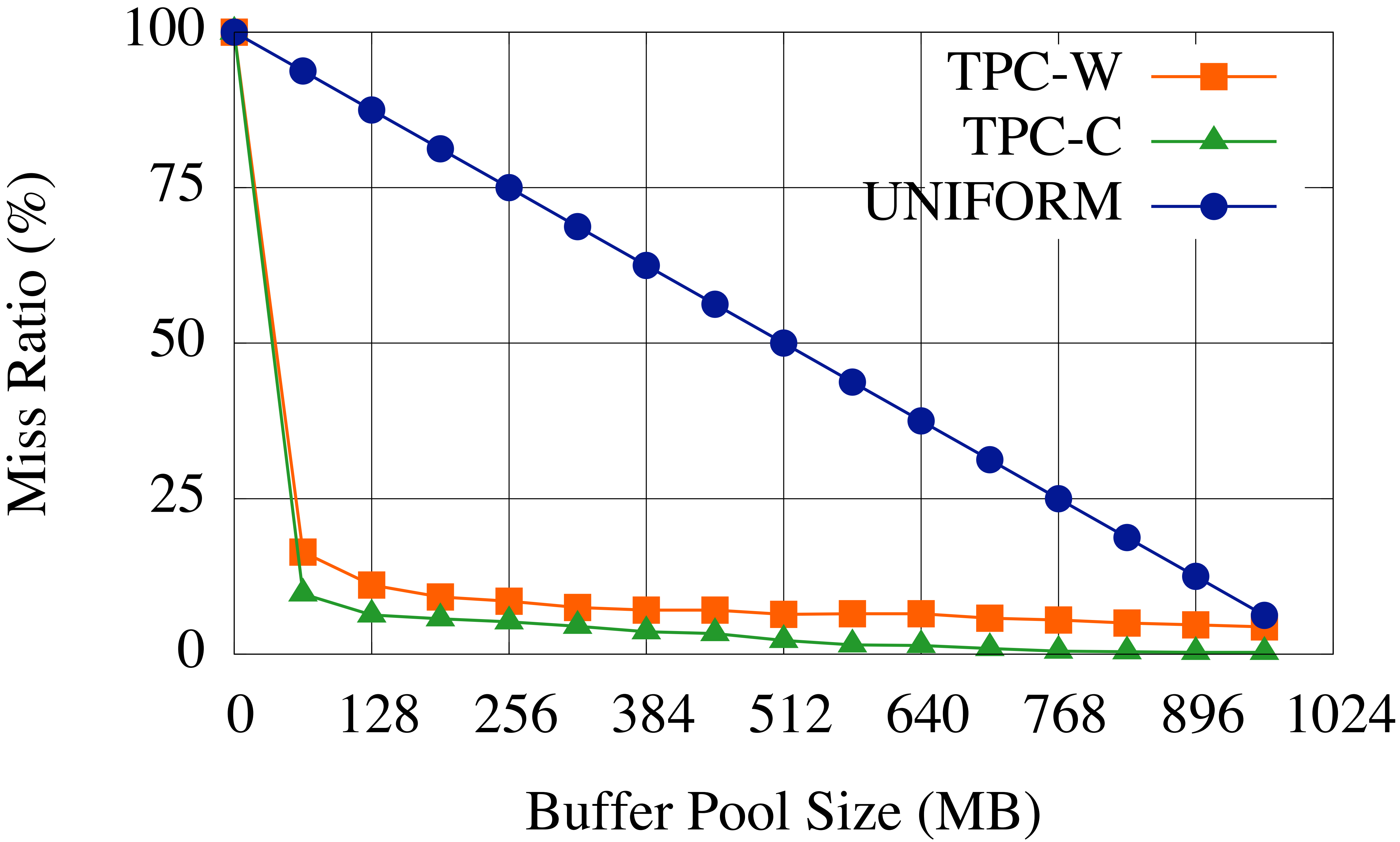
Algorithms for Comparison

- **MRC**
  - Partition cache (independently) using miss-ratio curves
- **DISK**
  - Partition caches equally, determine best disk quanta
- **MRC+DISK**
  - Run **MRC** then **DISK**
- **IDEAL***
  - Build model with SVM using 16*16*5=1280 sampled configurations

Roadmap of Results

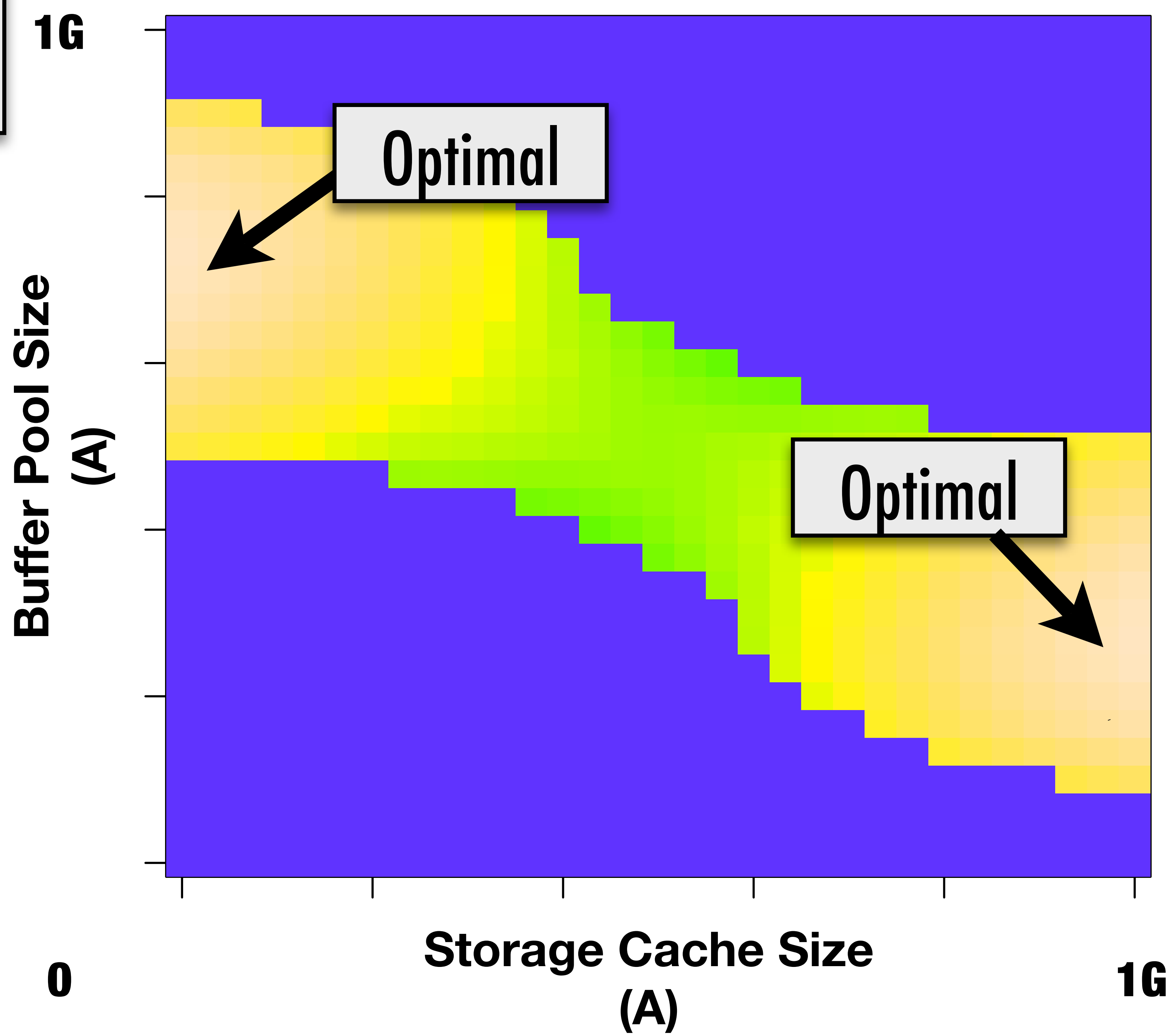
- **Multi-level cache allocator**
    - Using LRU and DEMOTE cache replacement policies
- **Multi-level cache and disk**
- **Accuracy of computed models**

Miss-Ratio Curves



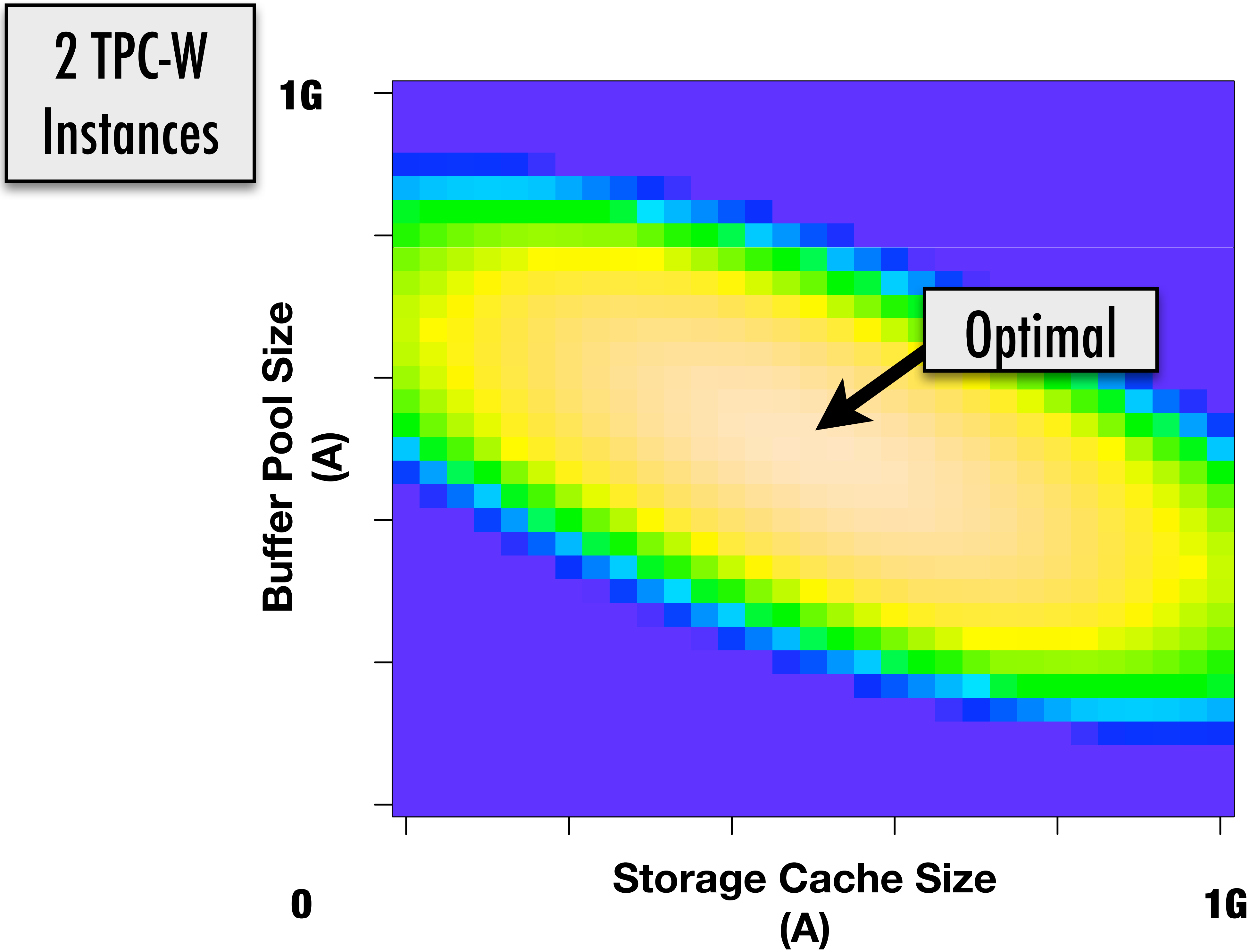
Multi-Level Caching (LRU)

2 TPC-W Instances



HeatMap
Lighter: Better
Darker: Worse

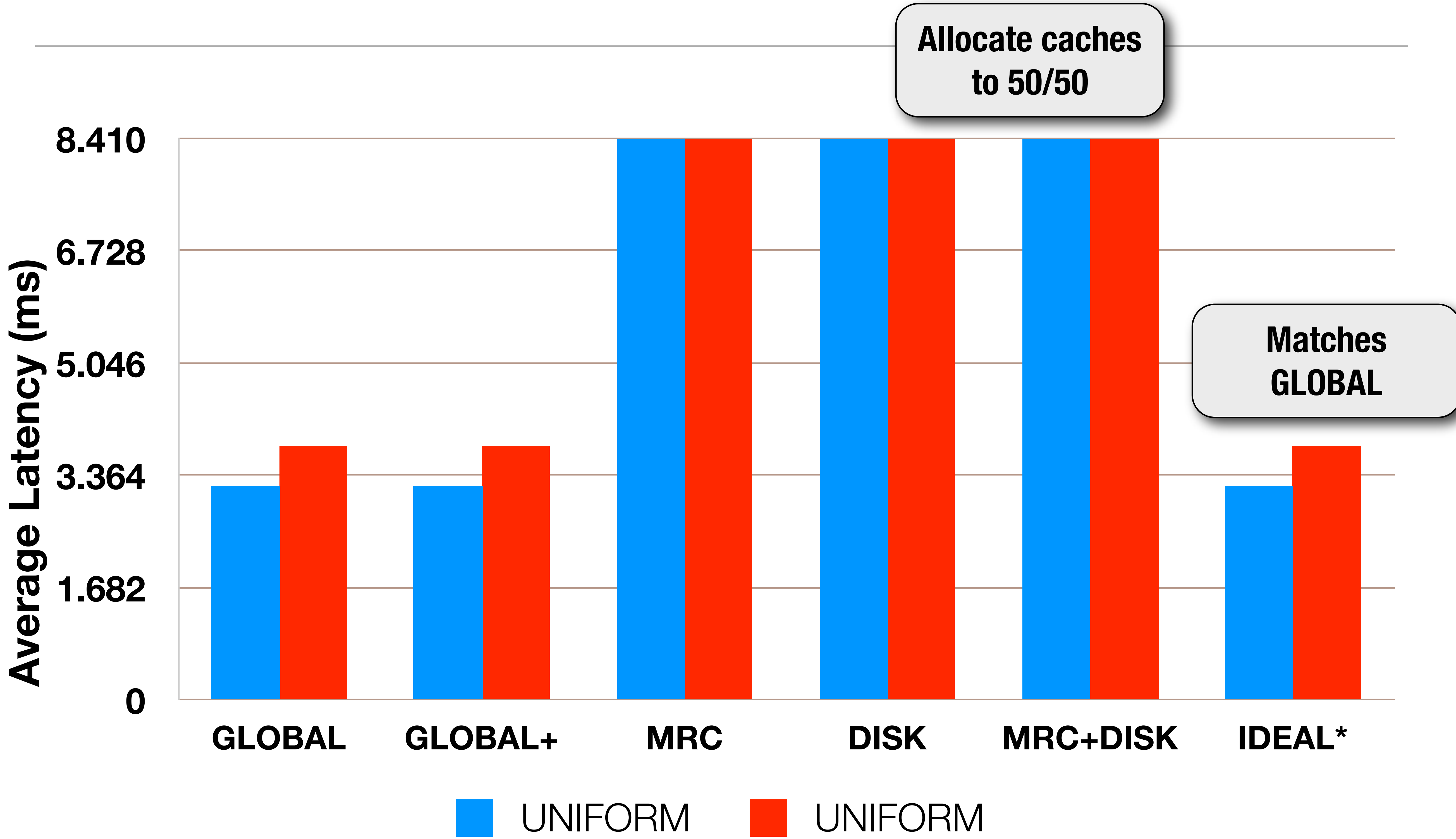
Multi-Level Caching (DEMOTE)



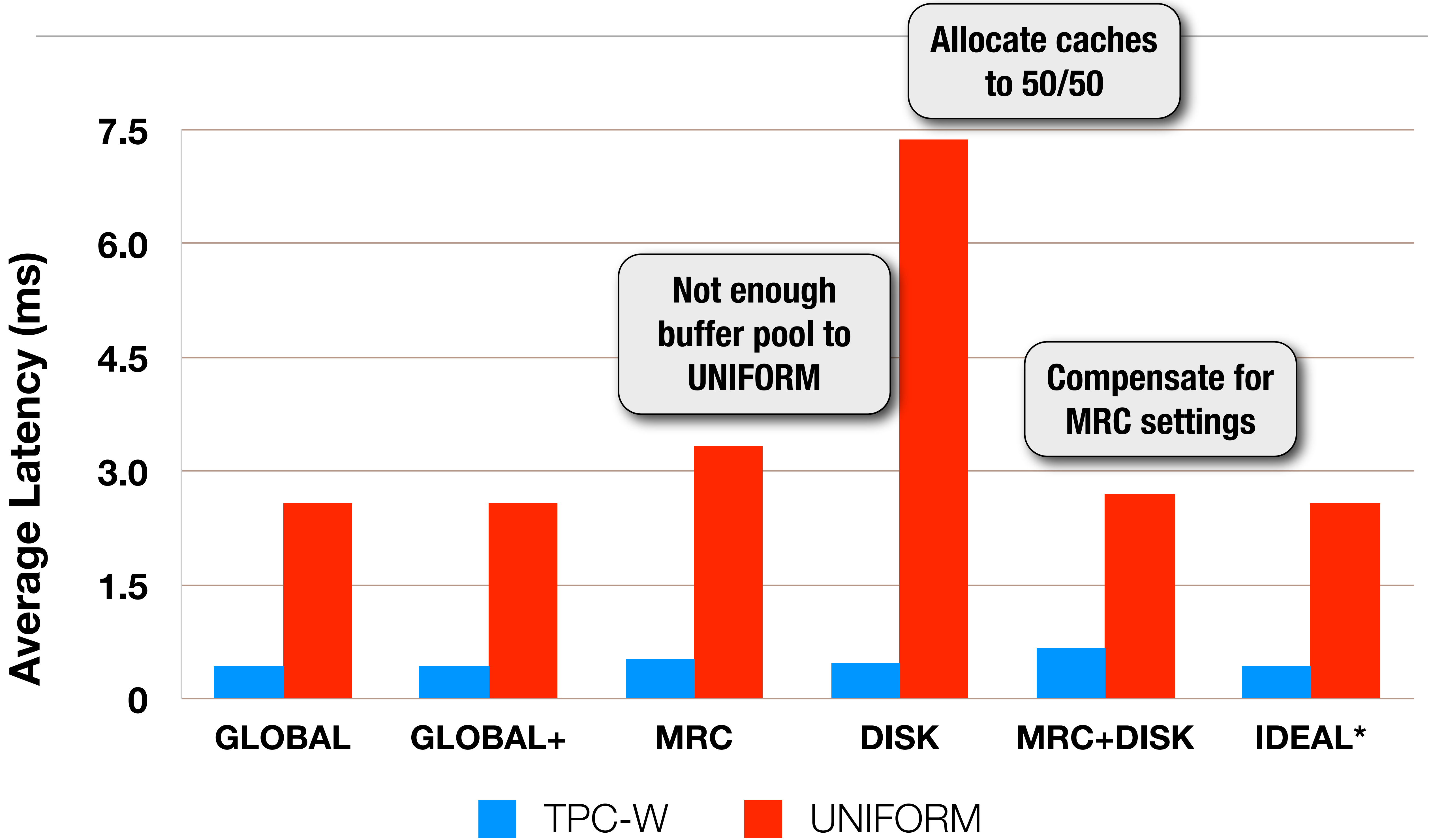
Roadmap of Results

- **Multi-level cache allocator**
- **Multi-level cache and disk**
  - Using two identical applications
  - Using different applications
- **Accuracy of computed models**

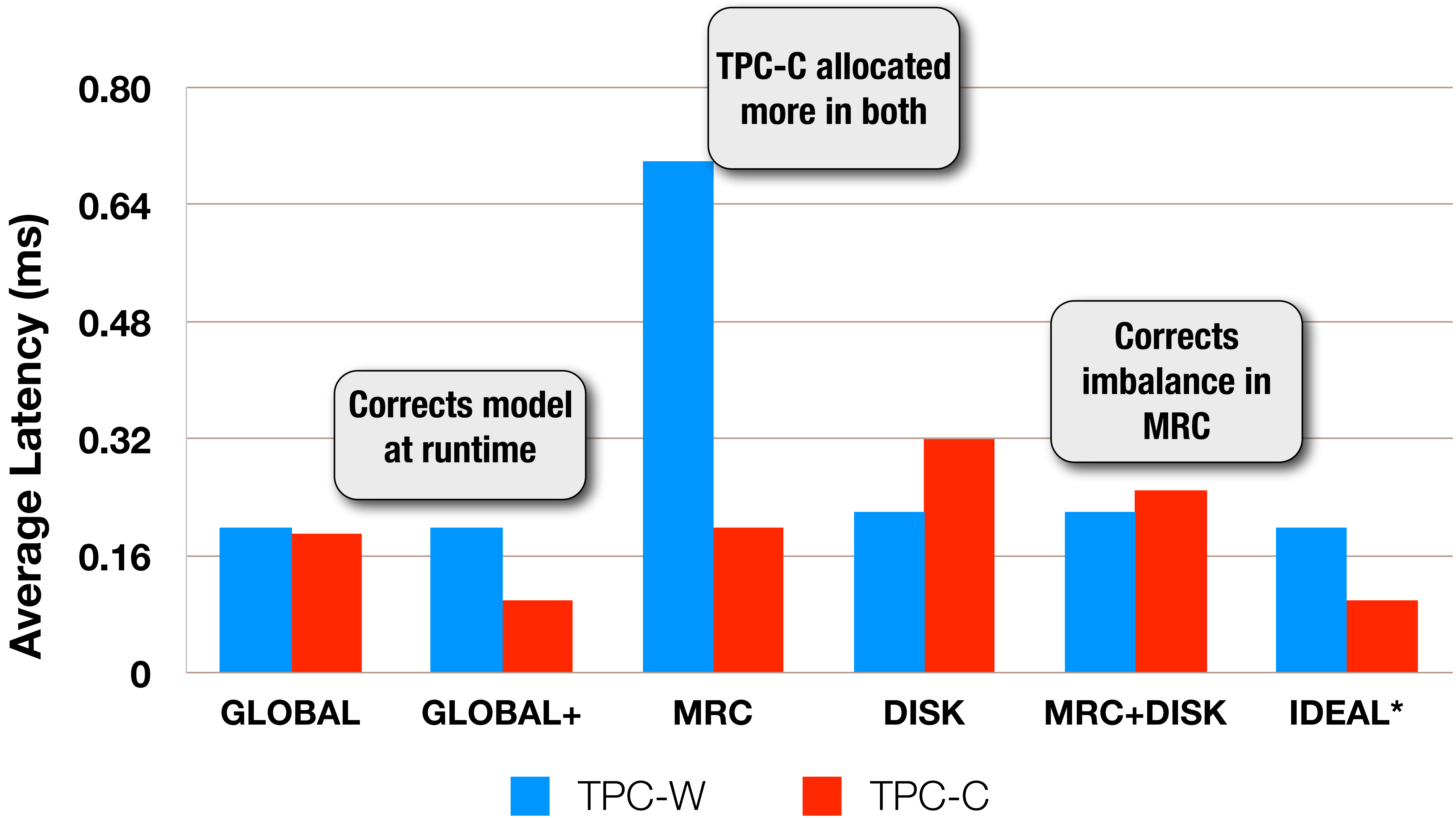
UNIFORM/UNIFORM



TPC-W/UNIFORM



TPC-W/TPC-C

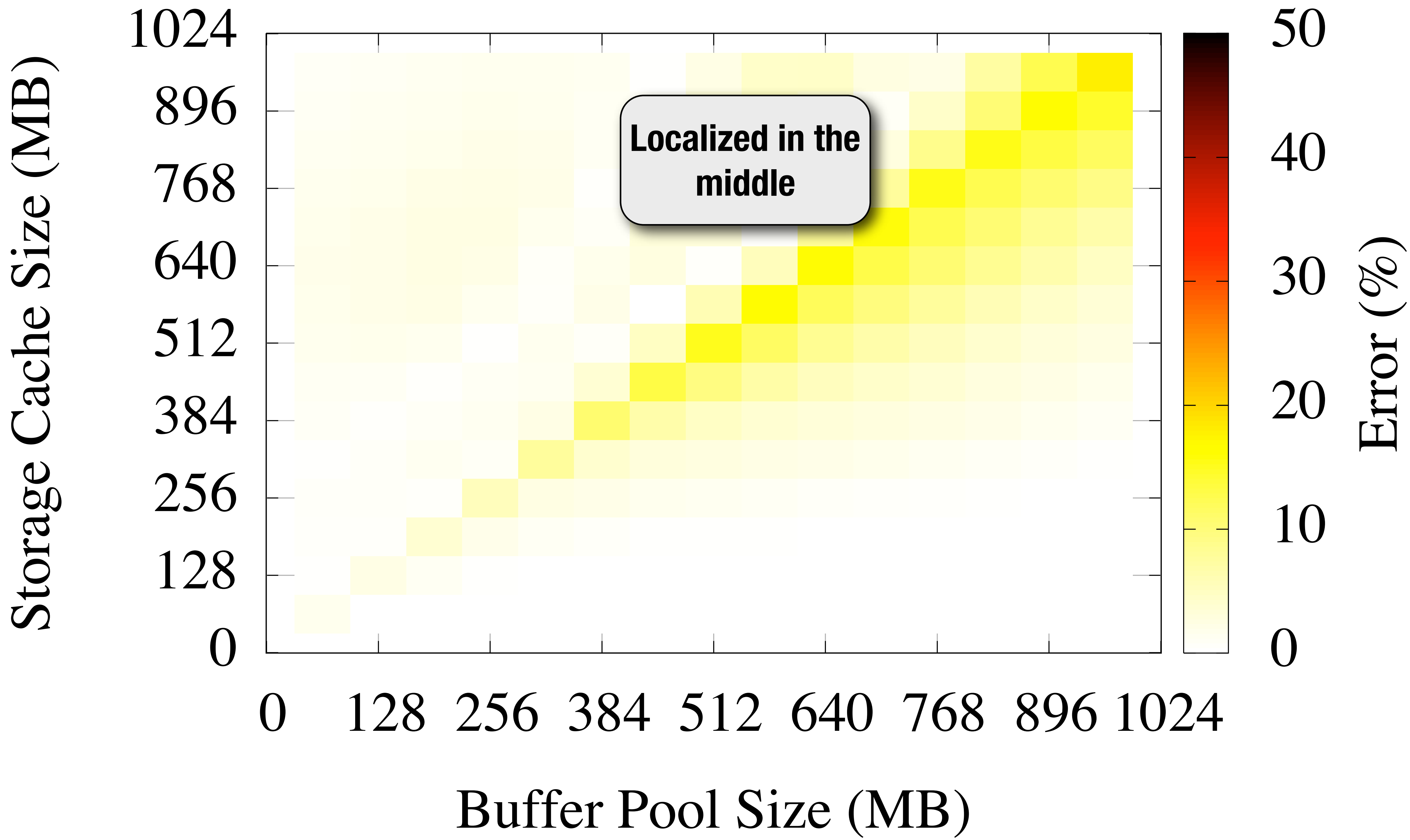


TPC-C allocated more in both
Corrects model at runtime
Corrects imbalance in MRC
0.80
0.64
0.48
0.32
0.16
0
Average Latency (ms)
GLOBAL
GLOBAL+
MRC
DISK
MRC+DISK
IDEAL*
TPC-W
TPC-C

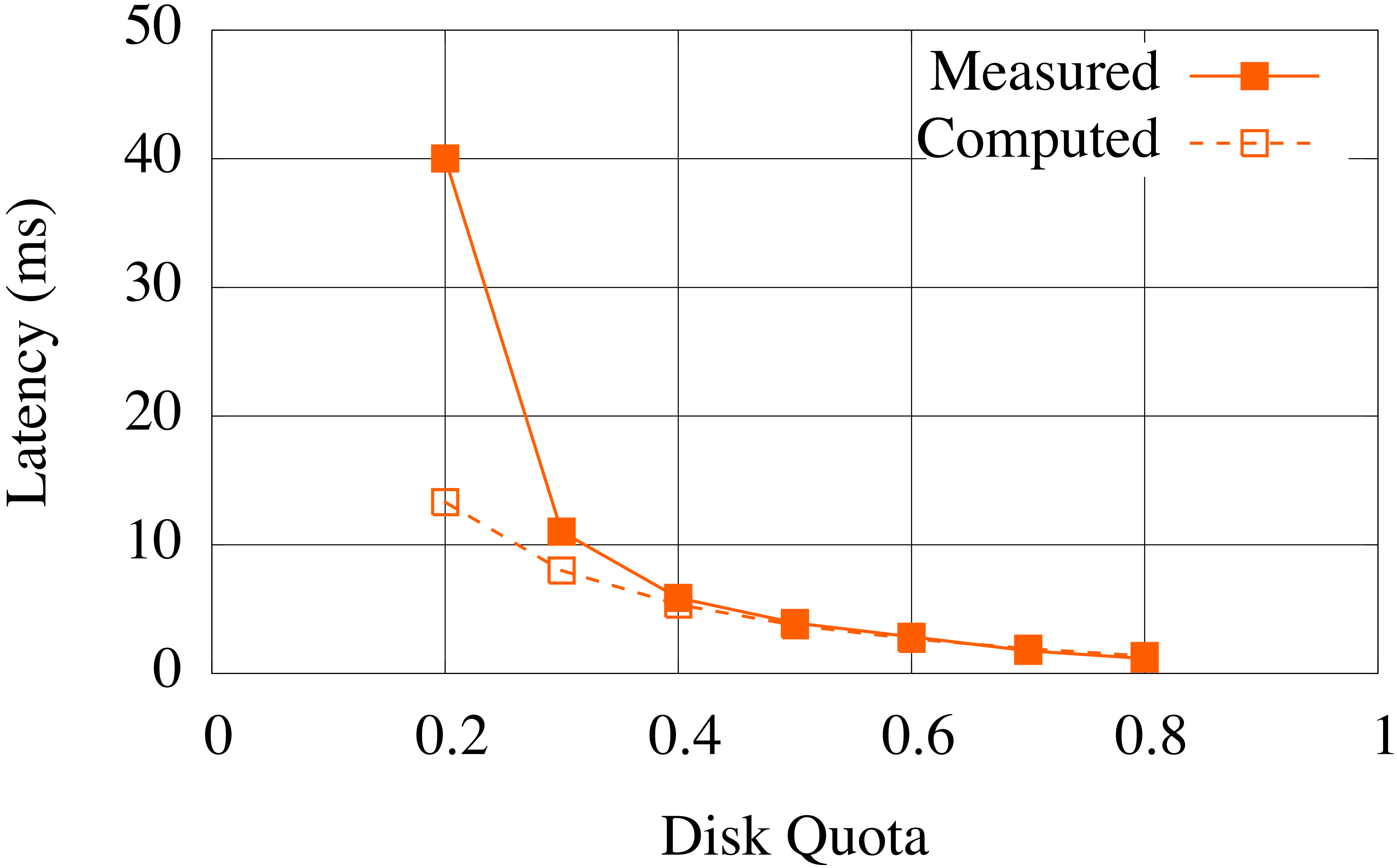
Roadmap of Results

- **Multi-level cache allocator**
- **Multi-level cache and disk**
- **Accuracy of computed models**
  - Cache model
  - Disk model

Cache Model Accuracy (TPC-W)



Disk Model Accuracy (TPC-W)



Conclusions

- **Problem**
  - Need to consider resources on multiple tiers
  - Independent cache/disk allocators are not sufficient
- **Dynamic allocation of cache hierarchy and disk**
  - Build performance models dynamically
  - Iteratively refine (if necessary)
  - Use models for global resource partitioning
- **Performance up to 2.9 better than single resource allocators**

Thank you.