The Storage Hierarchy is Not a Hierarchy: Optimizing Caching on Modern Storage Devices with Orthus

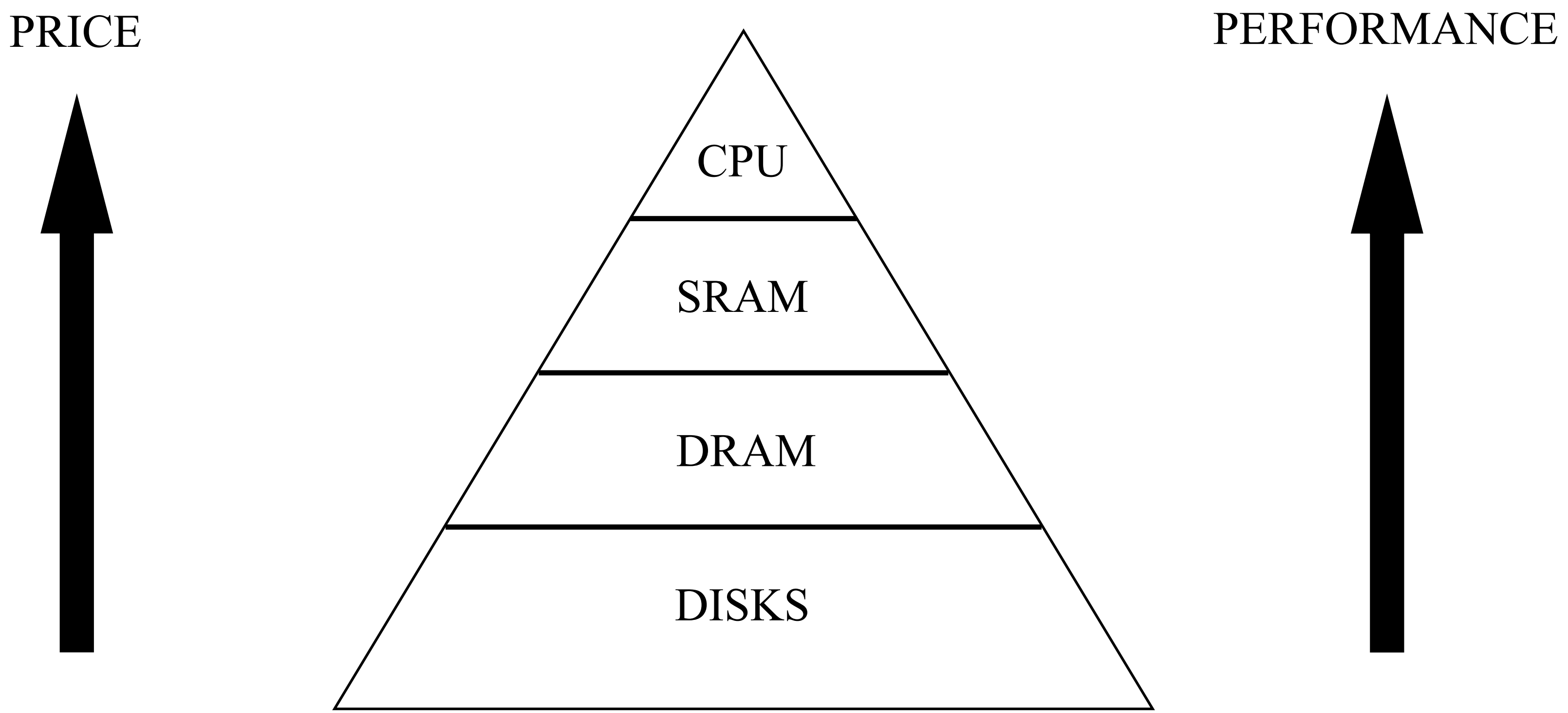
FAST'21

Authors: *Kan Wu, Zhihan Guo, Guanzhou Hu, Kaiwei Tu, Ramnatthan Alagappan, Rathijit Sen, Kwanghyun Park, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau*

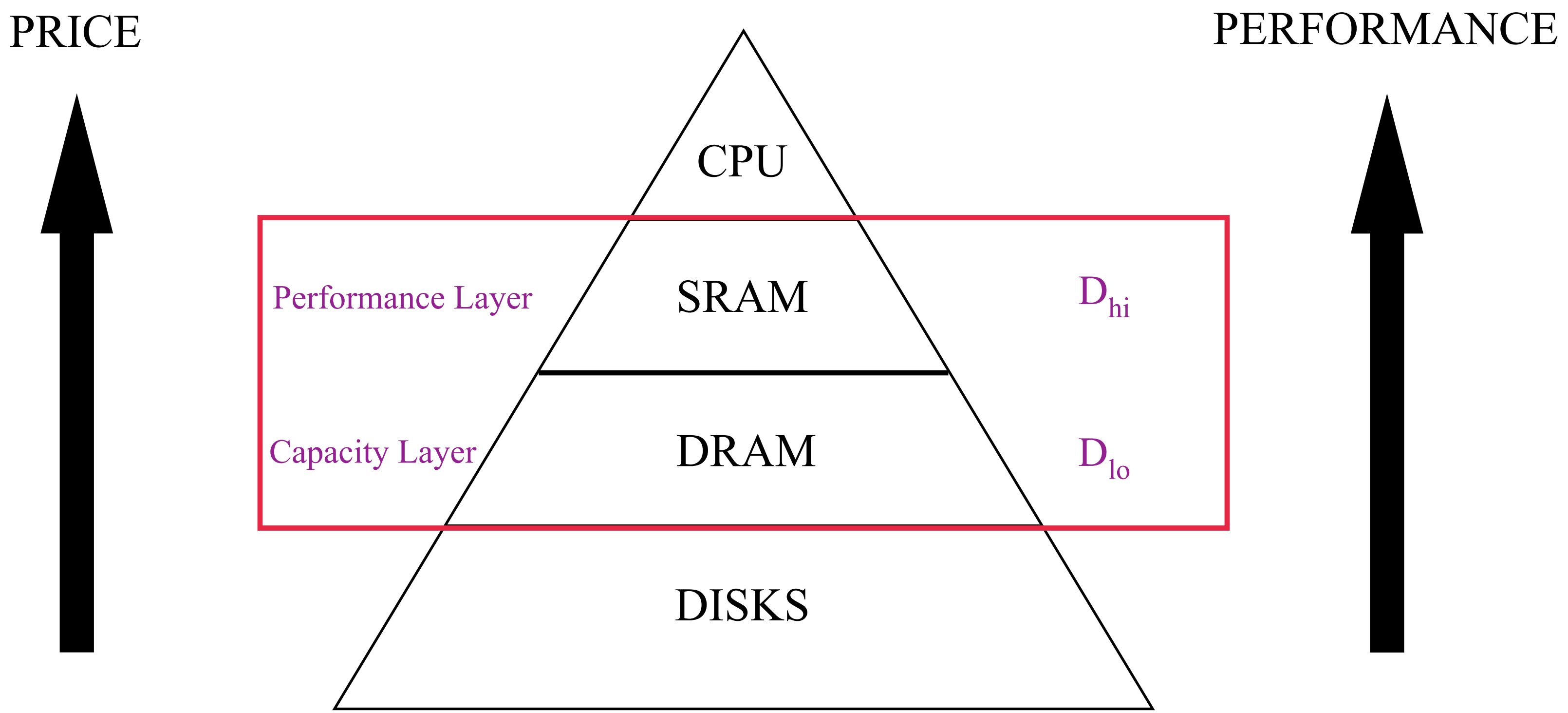
Presenter: **Jiaan Zhu, Chao Bi**

Outline

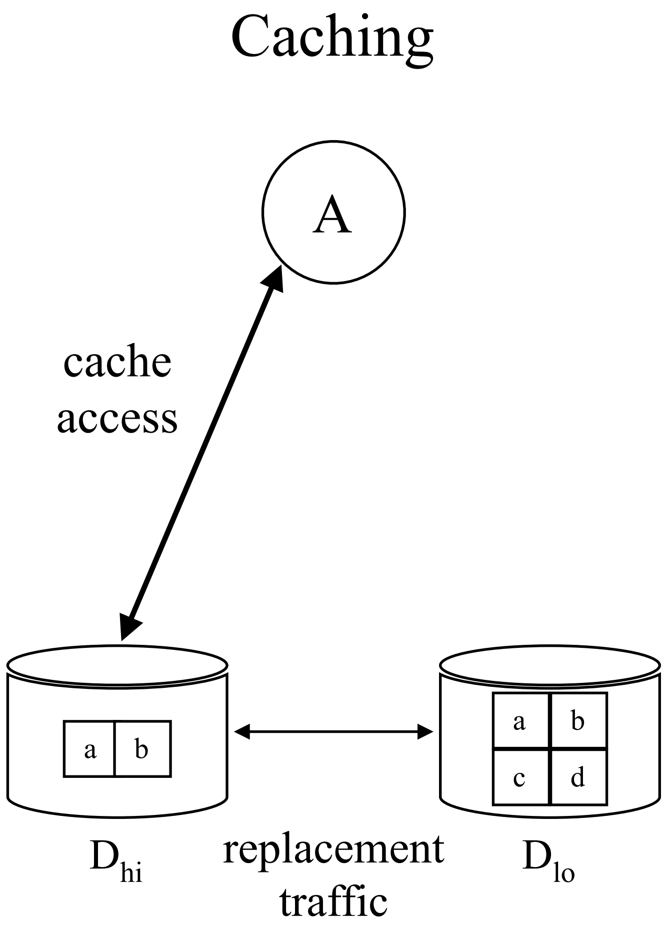
Traditional Hierarchy



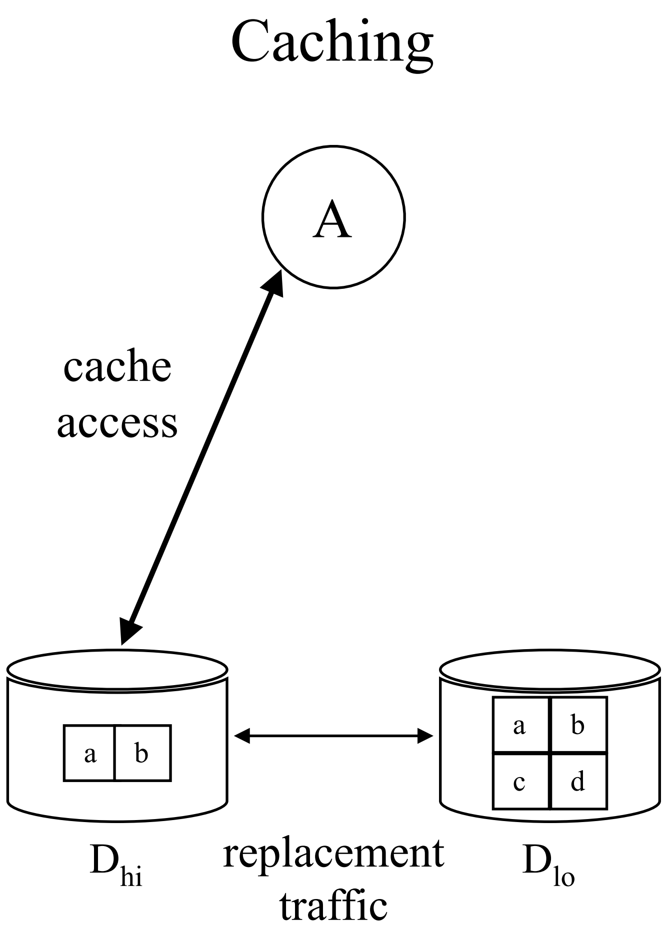
Traditional Hierarchy

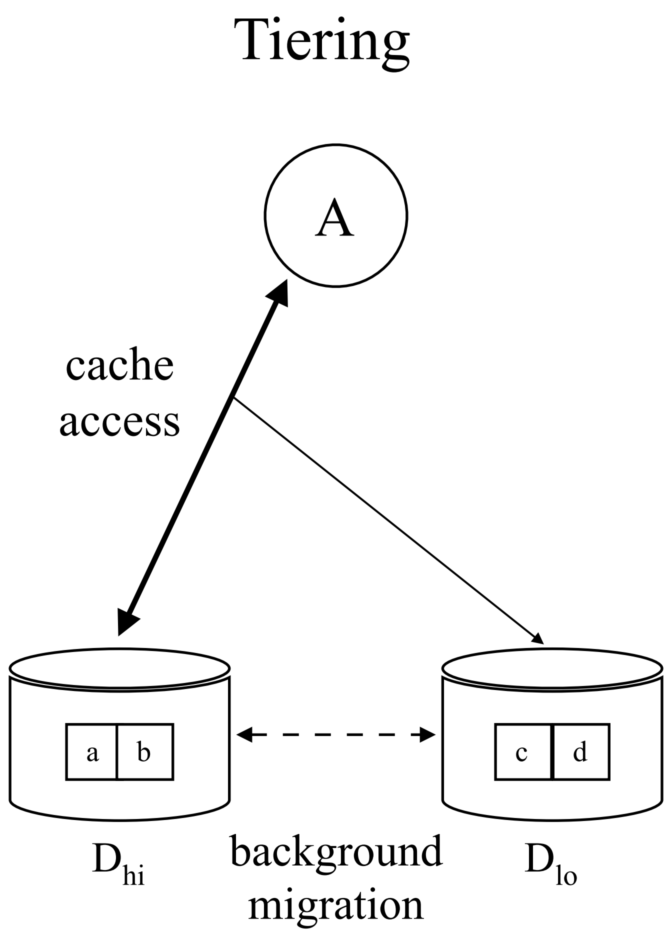


Traditional Hierarchy



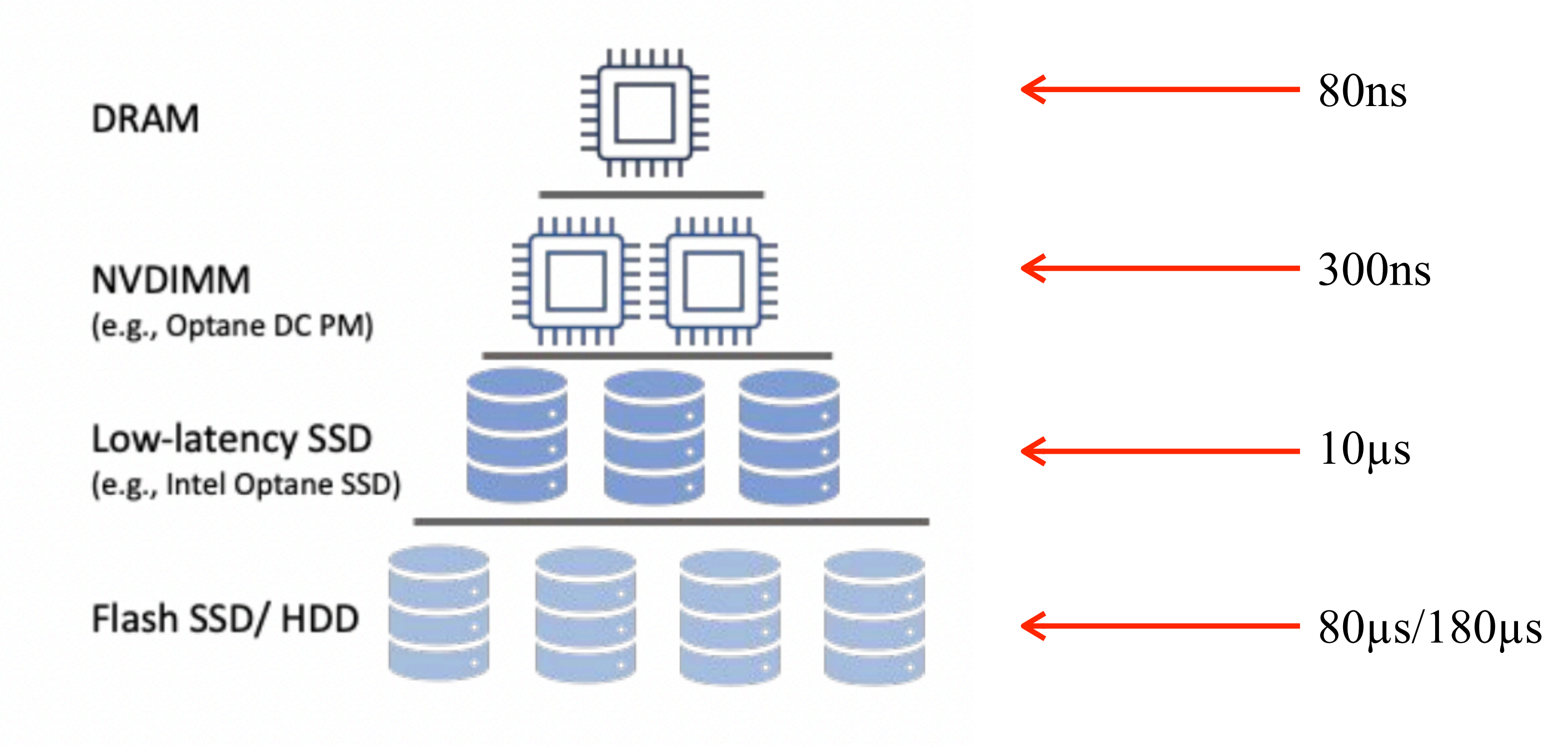
Traditional Hierarchy





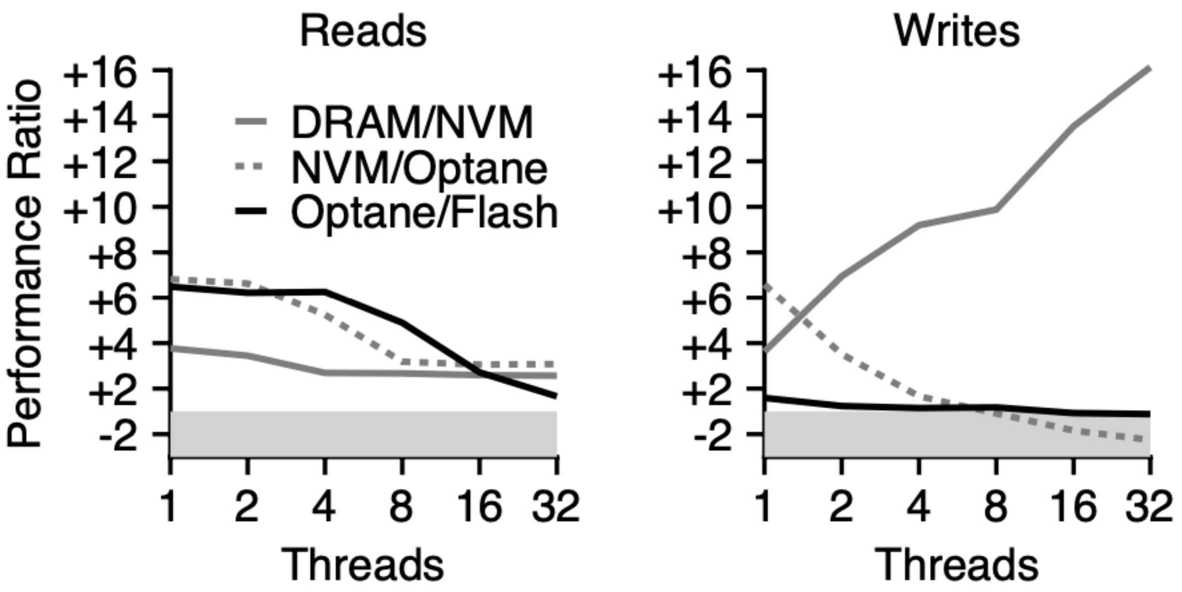
Challenge of Traditional Hierarchy

- The performance of the devices is similar in high concurrency situation.
  - Overlapping performance characteristics.



Challenge of Traditional Hierarchy

- The performance of the devices is similar in high concurrency situation.
  - Overlapping performance characteristics.

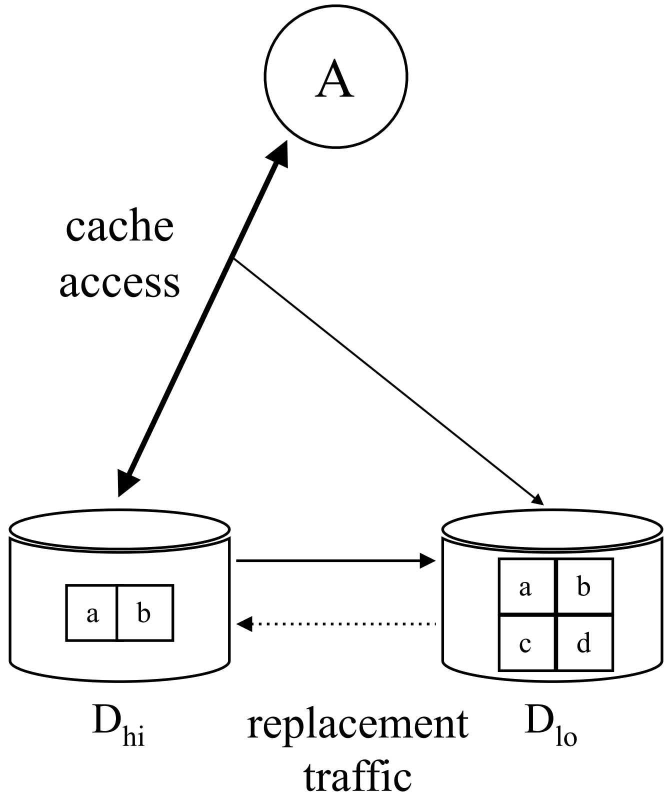


Challenge of Traditional Hierarchy

- The performance of the devices is similar in high concurrency situation.
  - Overlapping performance characteristics.
- It's useless to improve hit rates.
- Waste the performance of capacity layers.

Non-Hierarchical caching

- Key insight: move some of excess load to capacity device
  - Delivers additional useful performance from the capacity device.
  - Avoids data movement between devices.



Outline

USTC, CHINA ADSLAB

1. Background & Motivation
2. Characterizing Caching
3. Non-Hierarchical caching
4. Evaluation & Conclusion

Model

- D_{hi}: R_{hi} ops/s
- D_{lo}: R_{lo} ops/s
- Concurrency: little / copious
- Read-only workload
- Hit rate: $H \in [0,1]$
- Split rate: $S \in [0,1]$

Model

$$
\begin{aligned}
T_{cache,1} &= H \cdot T_{hit} + (1 - H) \cdot T_{miss} \\
&= H \cdot \frac{1}{R_{hi}} + (1 - H) \cdot (\frac{1}{R_{hi}} + \frac{1}{R_{lo}}) \\
&= \frac{1}{R_{hi}} + \frac{1 - H}{R_{lo}}
\end{aligned}
$$

$$
\begin{aligned}
B_{cache,1} &= \frac{1}{T_{cache,1}} \\
&= \frac{1}{\frac{1}{R_{hi}} + \frac{1-H}{R_{lo}}}
\end{aligned}
$$

Model

$$T_{slow}(N) = N \cdot (1 - H) \cdot \frac{1}{R_{lo}}$$

$$T_{fast}(N) = N \cdot \frac{1}{R_{hi}}$$

$$B_{cache,many} = \frac{N}{T_{cache,many}} = \frac{1}{max(\frac{1-H}{R_{lo}}, \frac{1}{R_{hi}})}$$

$$T_{cache,many} = max(T_{slow}(N), T_{fast}(N)) = max(N \cdot \frac{1-H}{R_{lo}}, N \cdot \frac{1}{R_{hi}})$$

Model

$$T_{split,1} = S \cdot \frac{1}{R_{hi}} + (1 - S) \cdot \frac{1}{R_{lo}}$$
$$= \frac{S}{R_{hi}} + \frac{1 - S}{R_{lo}}$$

$$B_{split,1} = \frac{1}{T_{split,1}}$$
$$= \frac{1}{\frac{S}{R_{hi}} + \frac{1-S}{R_{lo}}}$$

Model

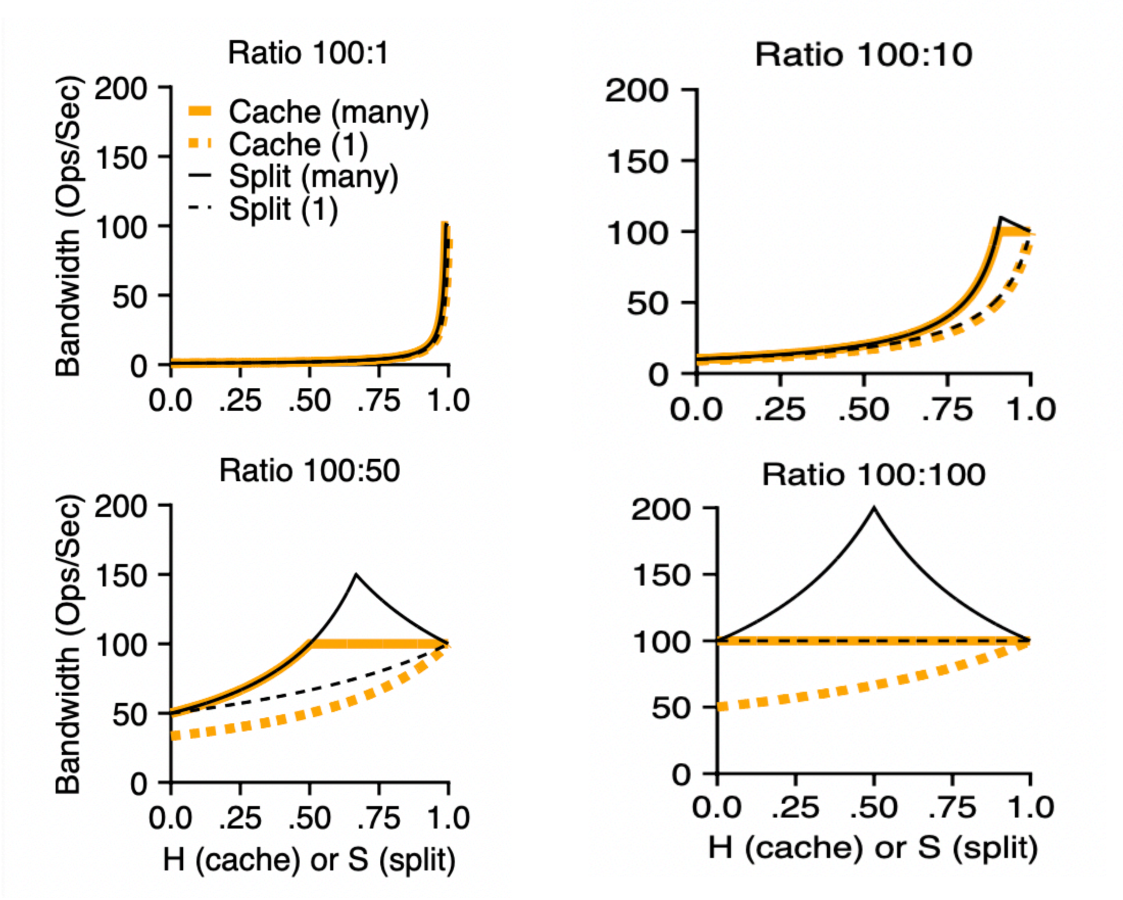
$$T_{slow}(N) = N \cdot (1 - S) \cdot \frac{1}{R_{lo}}$$

$$T_{fast}(N) = N \cdot \frac{S}{R_{hi}}$$

$$B_{split,many} = \frac{N}{T_{split,many}} = \frac{1}{max(\frac{1-S}{R_{lo}}, \frac{S}{R_{hi}})}$$

$$T_{split,many} = max(T_{slow}(N), T_{fast}(N)) = max(N \cdot \frac{1-S}{R_{lo}}, N \cdot \frac{S}{R_{hi}})$$

Modeled Performance



$$B_{cache,1} = \frac{1}{\frac{1}{R_{hi}} + \frac{1-H}{R_{lo}}}$$

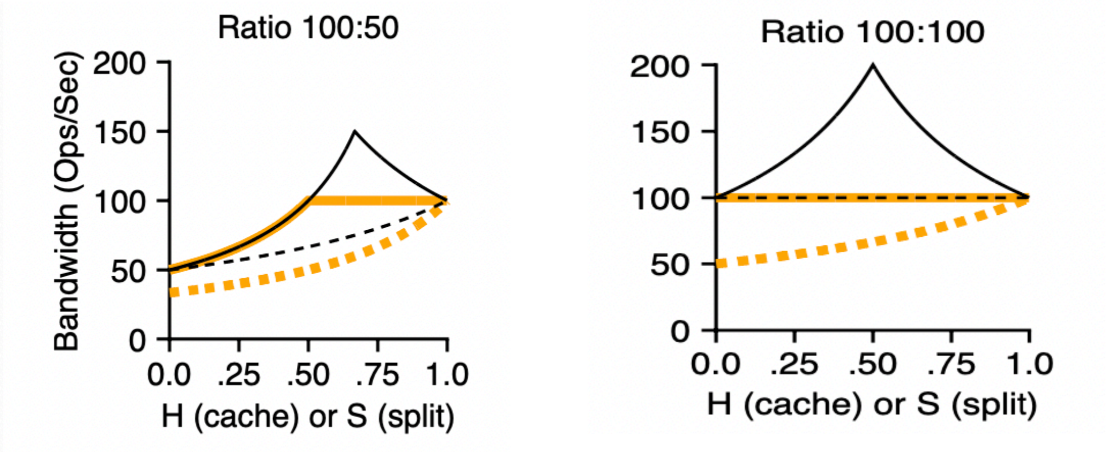
$$B_{cache,many} = \frac{1}{max(\frac{1-H}{R_{lo}}, \frac{1}{R_{hi}})}$$

$$B_{split,1} = \frac{1}{\frac{S}{R_{hi}} + \frac{1-S}{R_{lo}}}$$

$$B_{split,many} = \frac{1}{max(\frac{1-S}{R_{lo}}, \frac{S}{R_{hi}})}$$

Modeled Performance

- Classic caching is limited by the performance of D_{hi}.
- It's unnecessary to maximize the number of requests served by D_{hi}.

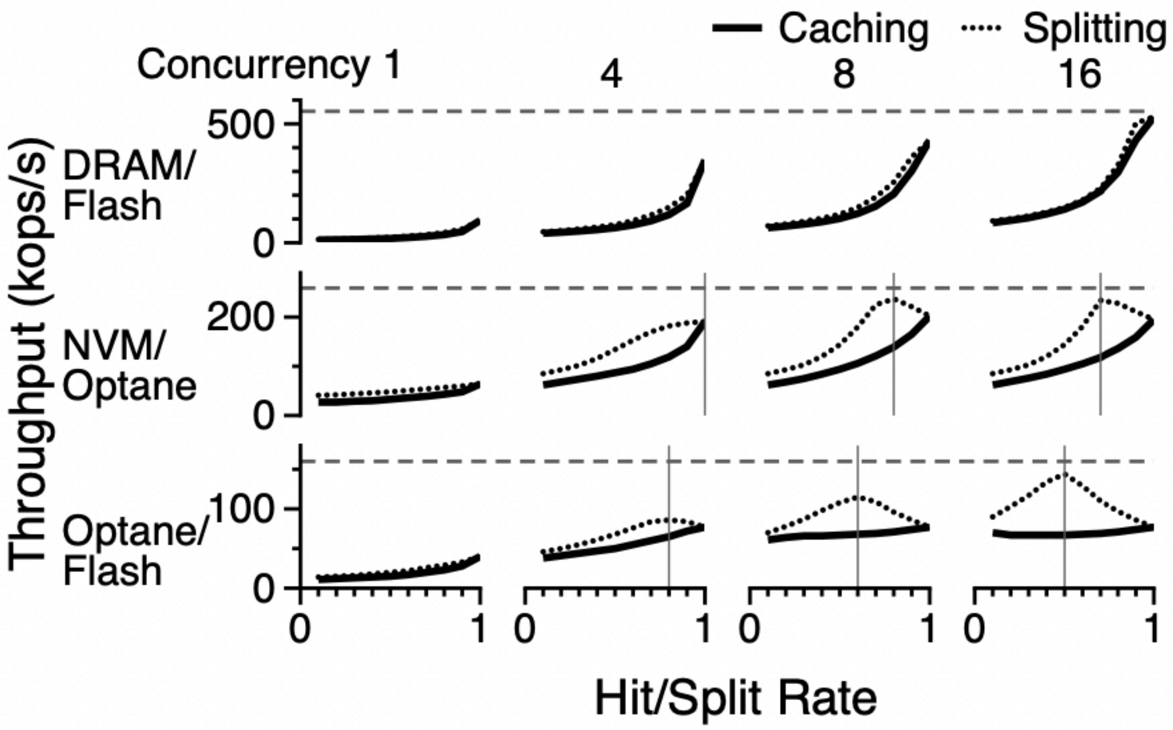


Evaluation with Optane DCPM and Optane SSD

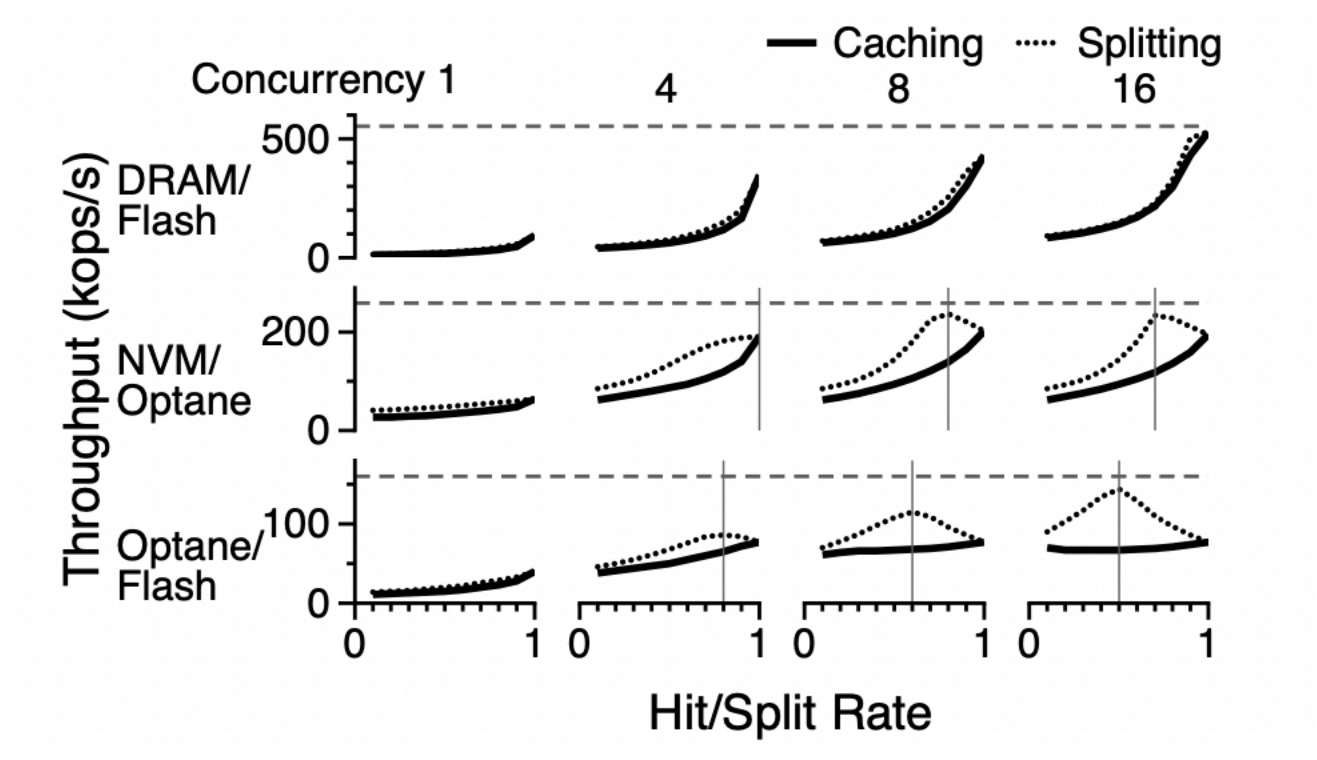
- NVM: Optane DCPM 128GB
- Optane: Optane 905P SSD
- Flash SSD
- Hierarchical Flexible I/O Benchmark(HFIO)

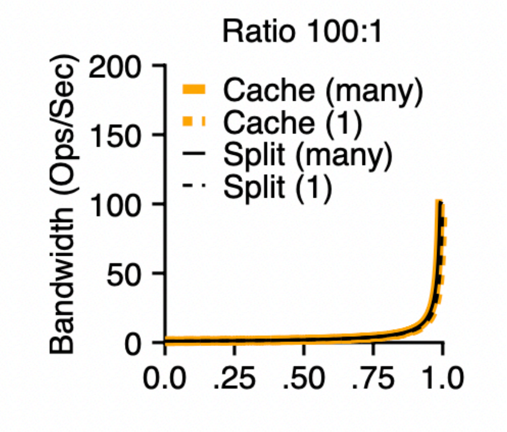
Evaluation with Optane DCPM and Optane SSD

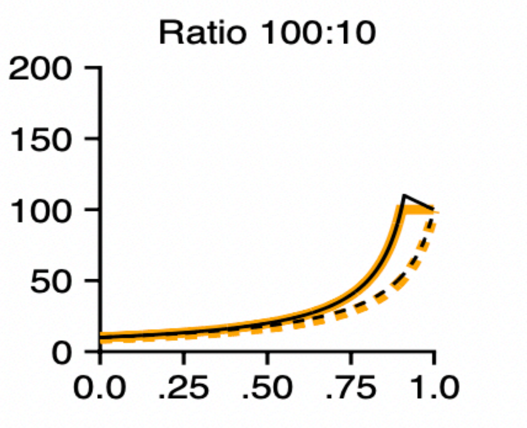
USTC, CHINA ADSLAB



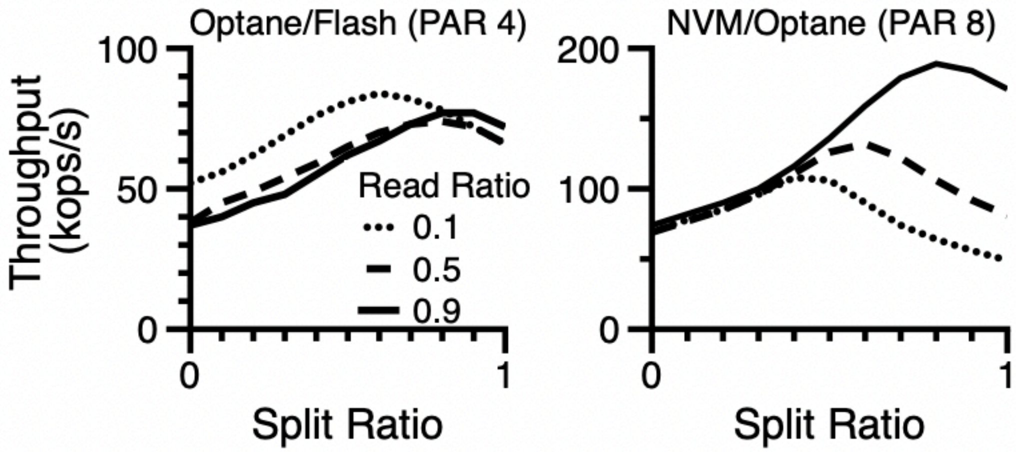
Evaluation with Optane DCPM and Optane SSD







Evaluation with Optane DCPM and Optane SSD



(a) Optane SSD + Flash (b) NVM + Optane SSD

Outline

Goals

- Perform as well or better than classic caching.
- Require no special knowledge or configuration.
- Be robust to dynamic workloads.

Steps of method

- Warming up system / After a significant workload change
  - Leverage classic caching
- After the hit rate has stabilized
  - Sending excess load to capacity layer
- If a workload changes
  - Return to classic caching

Formal Definitions

- W : Total workload
- $w \subset W$: Workload server by D_{hi}
- $P(W,w)=p_{hi}(w)+p_{lo}(W-w)$

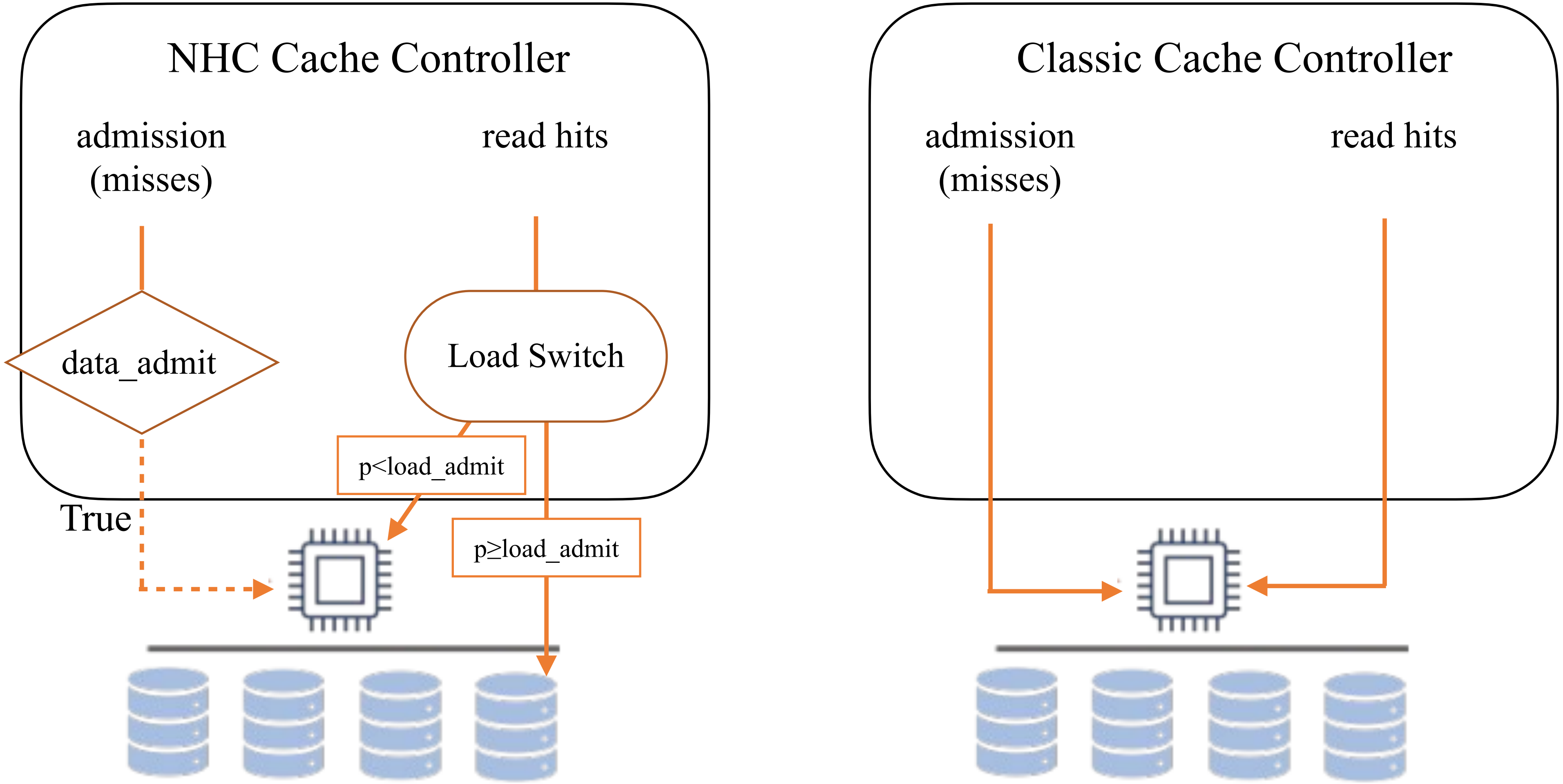
Formal Definitions

- Performance of a device has an upper bound.
  - $w_0 \subset w \quad p_{hi}(w_0)=L_{hi}$
- Increasing the workload on a device does not decrease performance.
- Before reaching upper limits, $p_{hi}(x)$ has a larger gradient than $p_{lo}(x)$

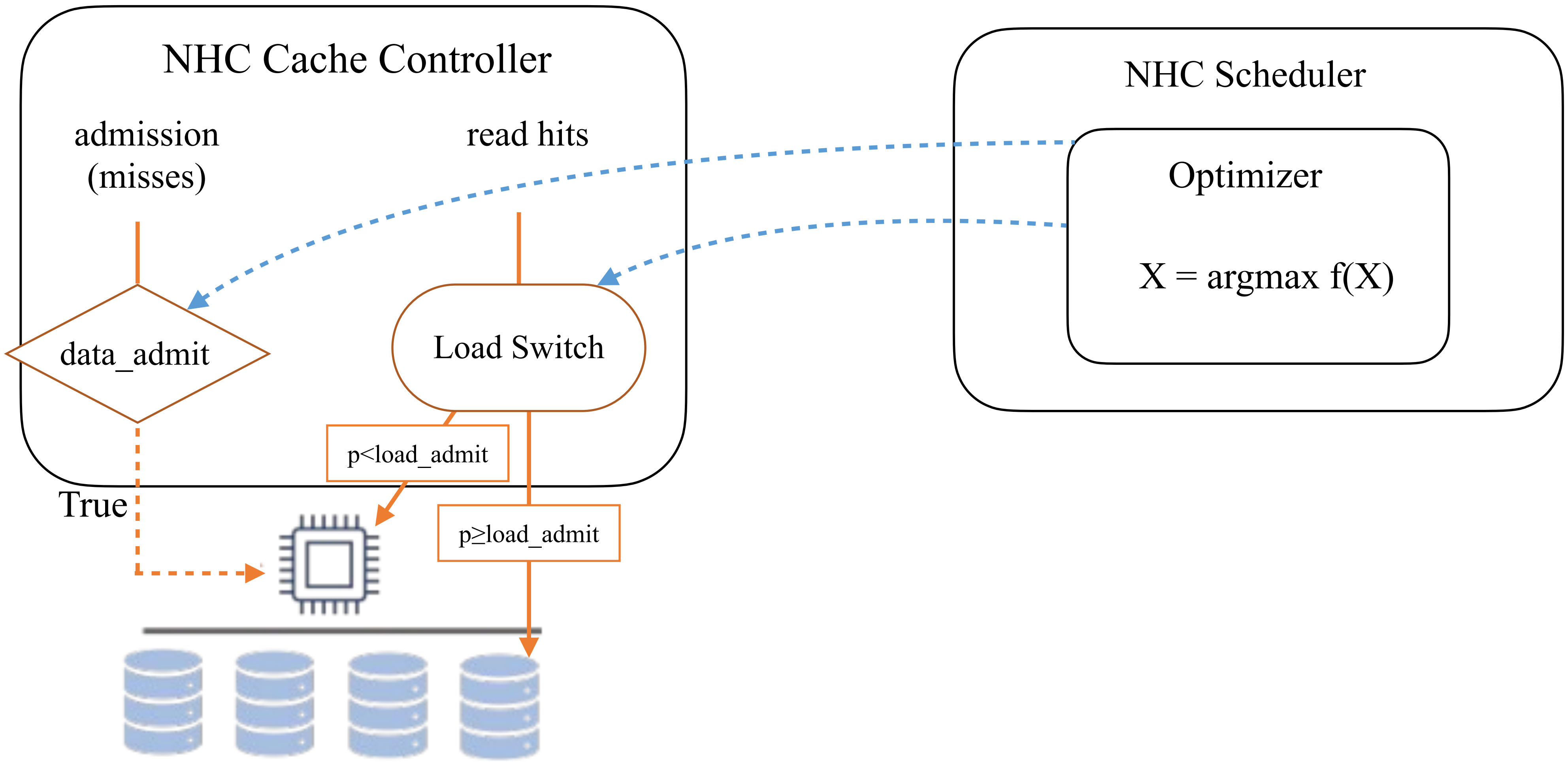
Formal Definitions

- Classic caching:
  - Finding working set w_{max} that maximizes the hit rate
- NHC:
  - w_0 in D_{hi}, $w_{max} - w_0$ in D_{lo}

Architecture



Architecture



Outline

The Storage Hierarchy is Not a Hierarchy: Optimizing Caching on Modern Storage Devices with Orthus

Thanks for your attention!