



# Building bridges in the cloud: More co-design for efficient and robust disaggregated architectures

Tiemo Bang (Microsoft Gray Systems Lab)

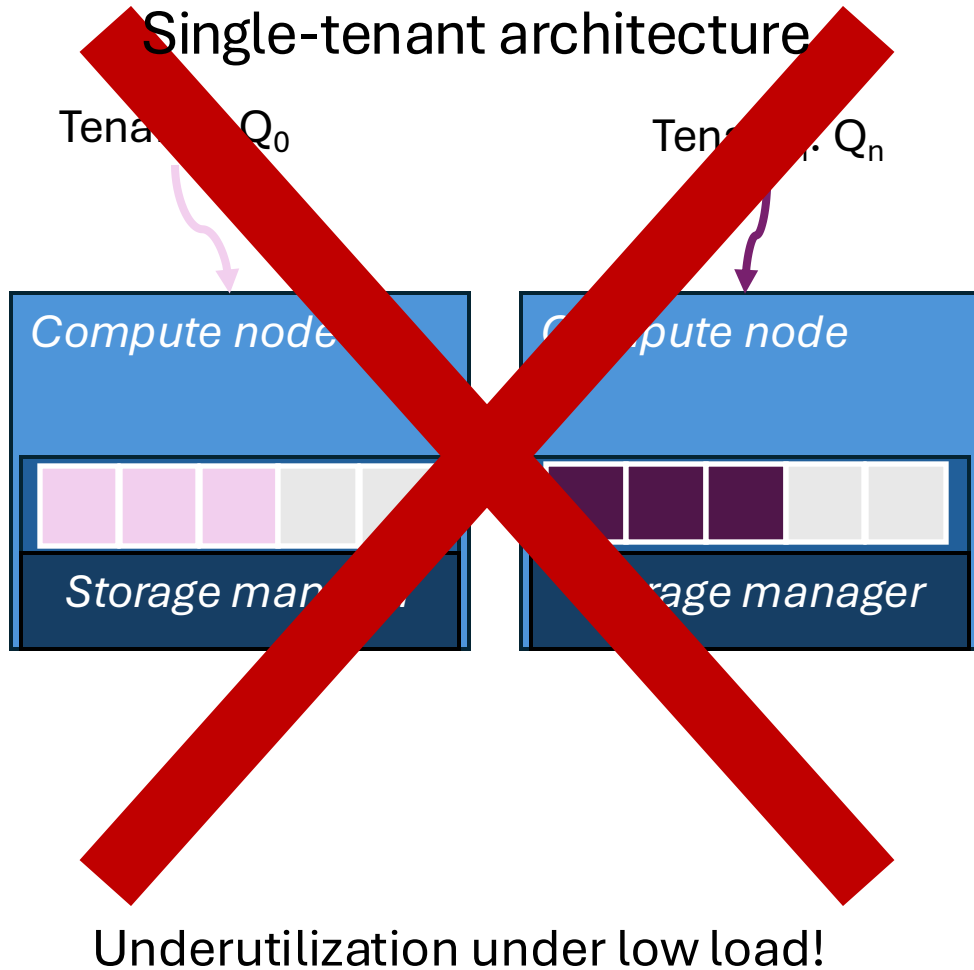
# A success story of co-design between theory and systems



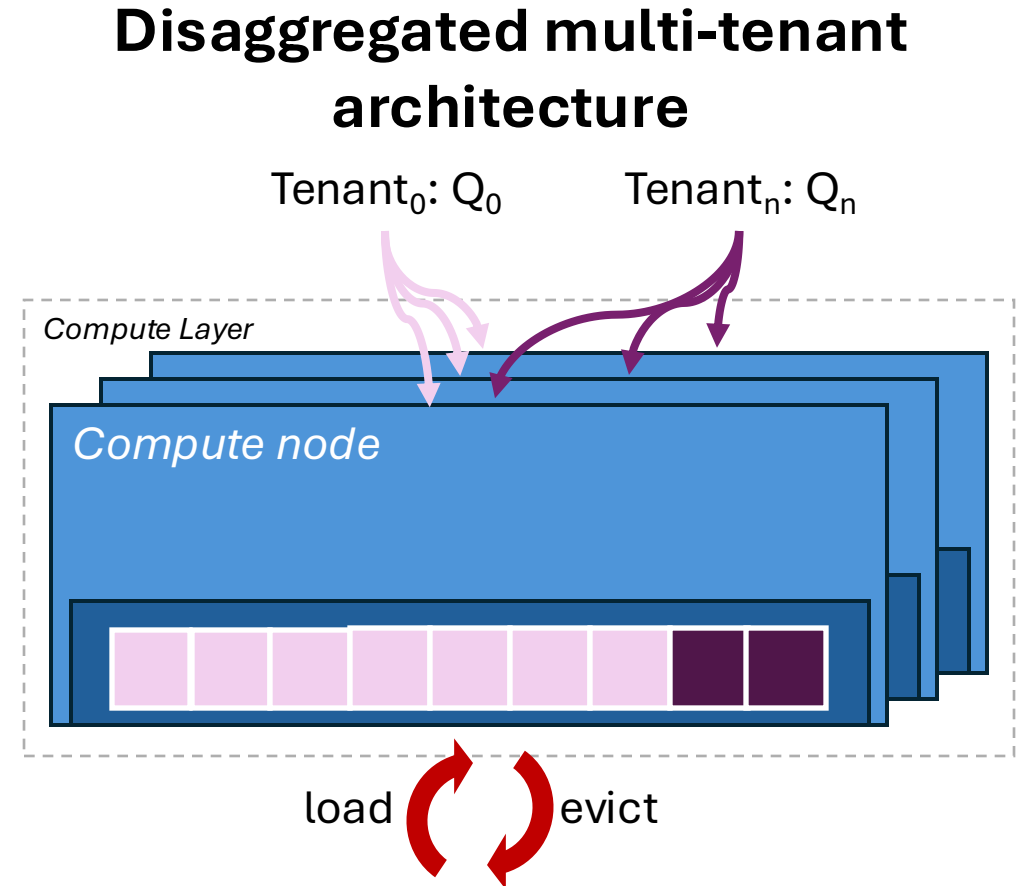
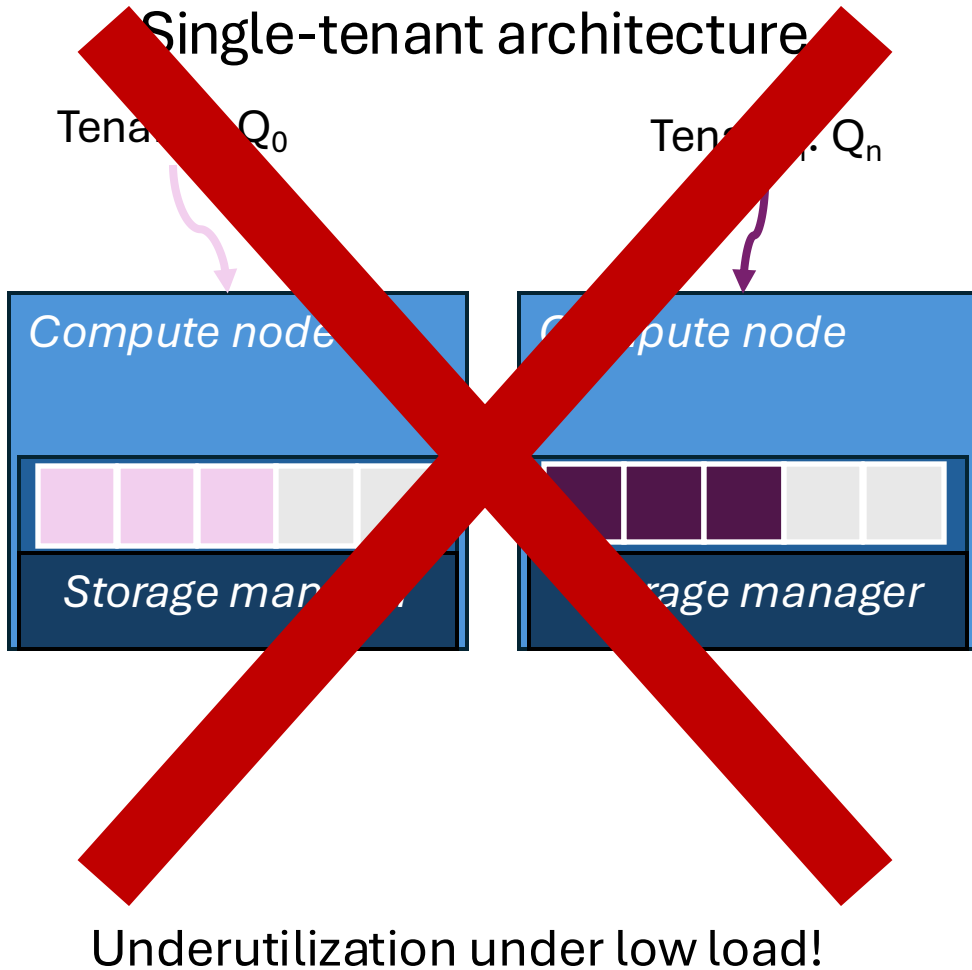
Relational algebra:

- Grounding relational system designs
- Continuous co-design, e.g., recently worst-case optimal joins!

# Utilization conundrum of disaggregated architectures

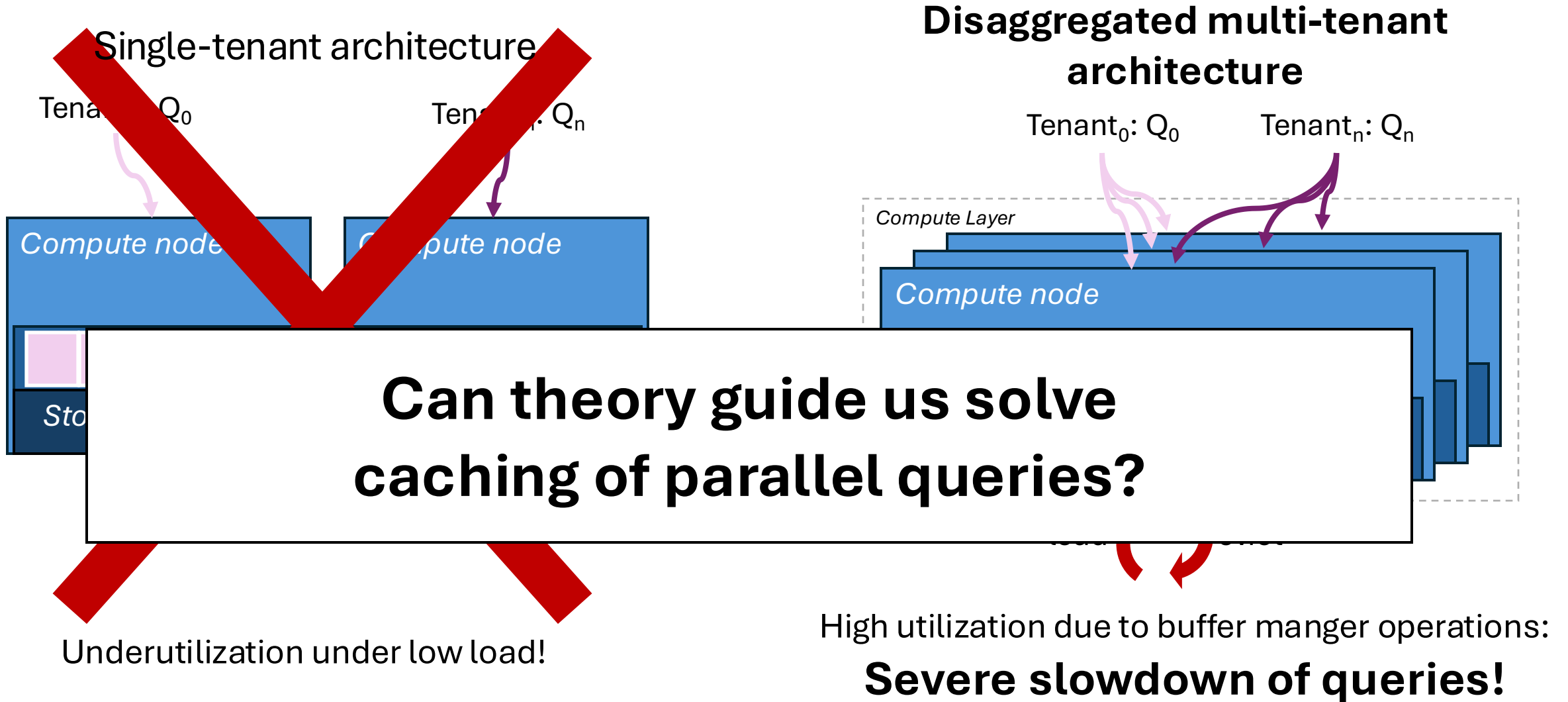


# Utilization conundrum of disaggregated architectures



High utilization due to buffer manager operations:  
**Severe slowdown of queries!**

# Utilization conundrum of disaggregated architectures





Challenge for co-design:  
How to find and  
understand theory, e.g.,  
for parallel caching?

# Outline

**LLM as superpower to bridge systems and theory**

**Co-design for caches in disaggregated data systems**

- The parallel paging problem
- Theoretic insights and **a breakthrough**
- More co-design required

# Fluffy questions make for fluffy answers!

 You

Explain the state-of-the-art caching algorithm for parallel queries known in algorithms theory.

 Copilot

One of the state-of-the-art caching algorithms for parallel queries is **SIEVE**. Developed by researchers from Emory University, Carnegie Mellon University, and the Pelikan Foundation,

SIEVE is designed to optimize web caching by simplifying and improving cache management <sup>1</sup>

<sup>2</sup>.





Copilot

Help me find the name of a pro...

Work

Web

+ New chat



You

Help me find the name of a problem type for caching algorithms.

I am starting from the basic online caching problem:

In the basic online caching problem, we have a cache of size  $B$  and a sequence of page requests  $\sigma = (p_1, \dots, p_T)$ . At every time step  $t$ , we receive an access to page  $p_t$  and if the page is not in the cache, we incur a page miss and the page  $p_t$  is brought in the cache. If the cache is full and there is no space for page  $p_t$ , the algorithm has to make a decision about which page to evict from the cache. The cost of any algorithm is the number of page misses over the complete sequence.

List formal names and key properties of problems known in algorithms theory research that could match description?



## Copilot

Certainly! Here are some formal names and key properties of problems related to the basic online caching problem:

### 1. Online Paging Problem:

- **Description:** This is the classic problem you described. It involves managing a cache of limited size and deciding which page to evict when a new page needs to be loaded.
- **Key Properties:**
  - **Competitive Ratio:** Measures the performance of an online algorithm compared to the optimal offline algorithm.
  - **Common Algorithms:** Least Recently Used (LRU), First-In-First-Out (FIFO), and Least Frequently Used (LFU).

### 2. k-Server Problem:



**You**

Online paging fits best.

What is the problem name and definition when online paging incurs parallel accesses?



**Copilot**

The problem you're referring to is known as the **Online Parallel Paging Problem**.

### **Definition:**

In the **Online Parallel Paging Problem**, multiple processors share a common cache. Each processor generates its own sequence of page requests, and the goal is to manage the cache in such a way that the overall performance is optimized. Specifically, the objective is to minimize the **makespan**, which is the time required to complete all page requests across all processors <sup>1</sup>

<sup>2</sup>.



**You**

Do you have references for optimal algorithms for online parallel paging?



**Copilot**

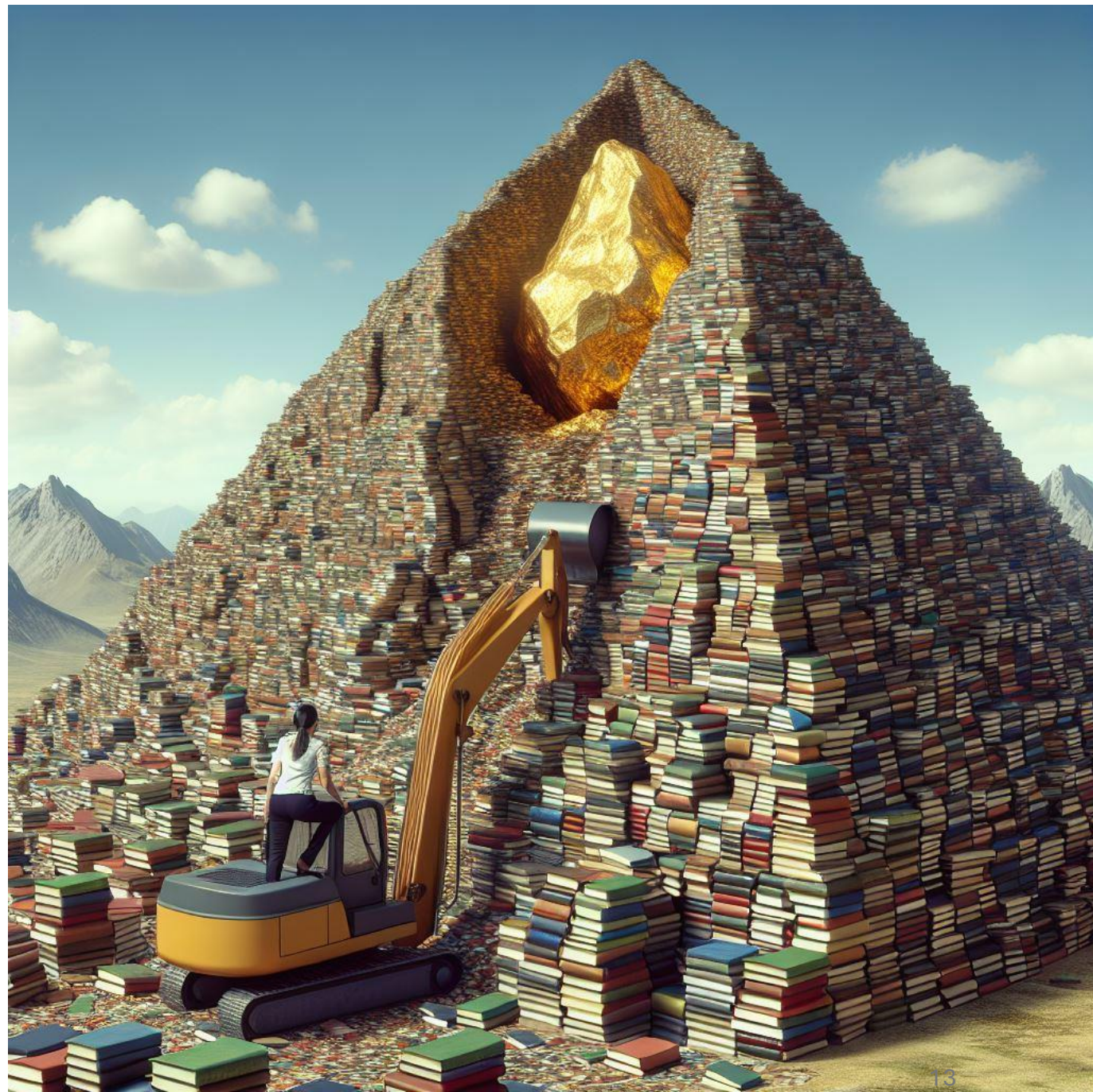
Yes, there are several key references for optimal algorithms in the context of online parallel paging:

1. **“Online Parallel Paging with Optimal Makespan”** by Kunal Agrawal, Michael A. Bender, Rathish Das, William Kuszmaul, Enoch Peserico, and Michele Scquizzato:

- This paper presents algorithms that achieve an optimal competitive ratio for the online parallel paging problem. The authors show that the optimal deterministic competitive ratio for this problem is in the range of

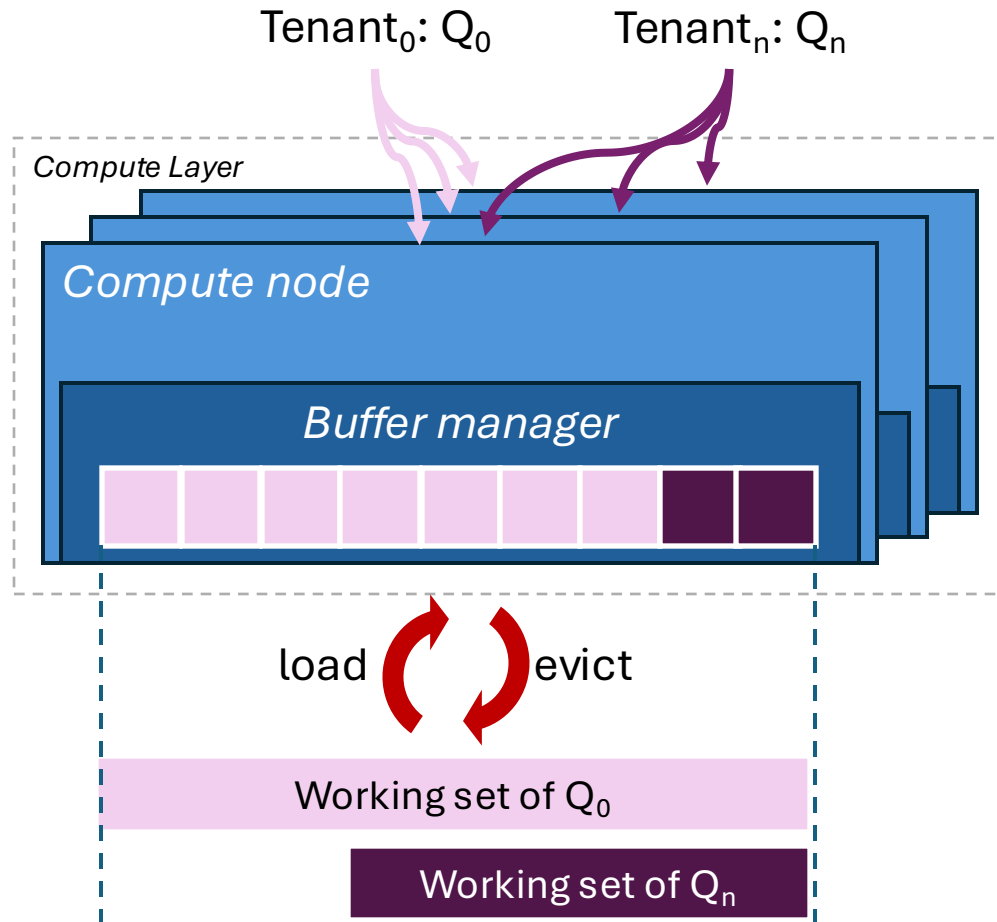
$$\Omega(\log k) \leq \text{ratio} \leq O(\log^2 k)$$

LLM superpower for co-design:  
Learn basics of parallel  
caching theory in a day!



# **Co-design for caches in disaggregated data systems**

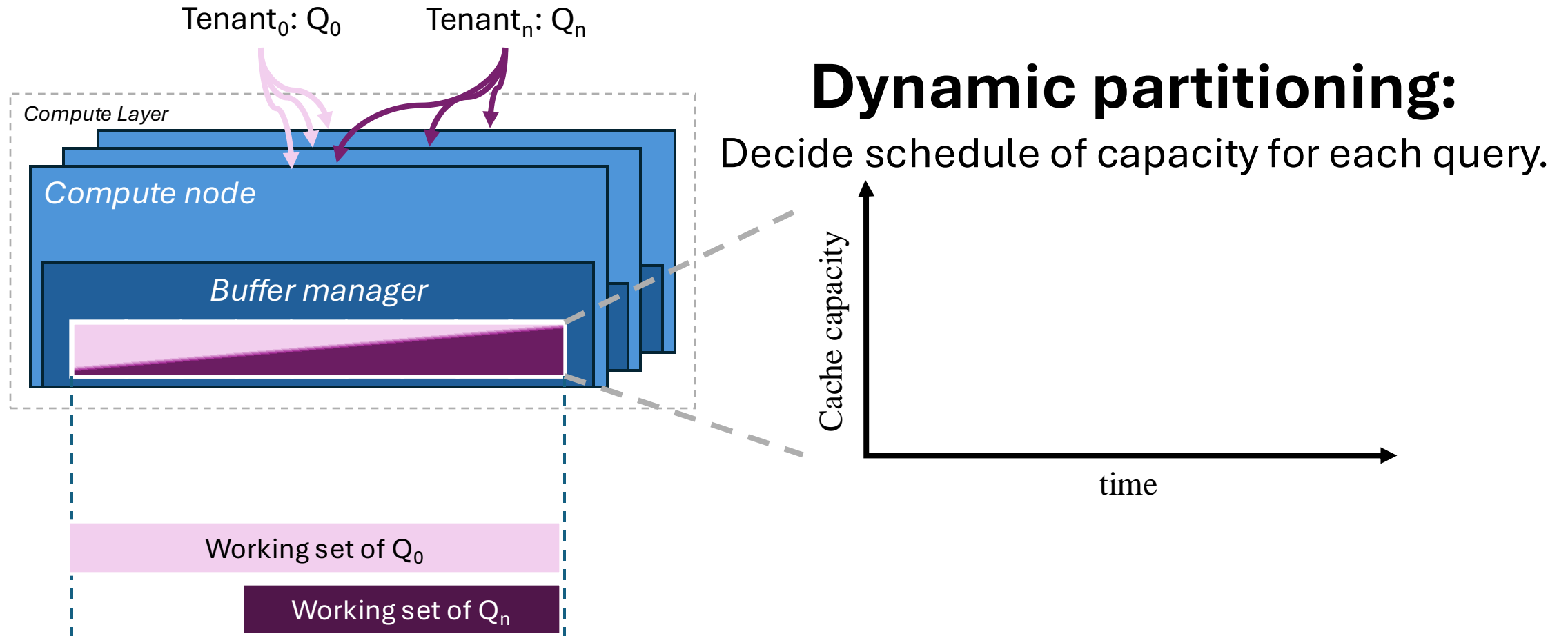
# Insight 1) Two subproblems in parallel paging: Page replacement + Dynamic partitioning



## Page replacement

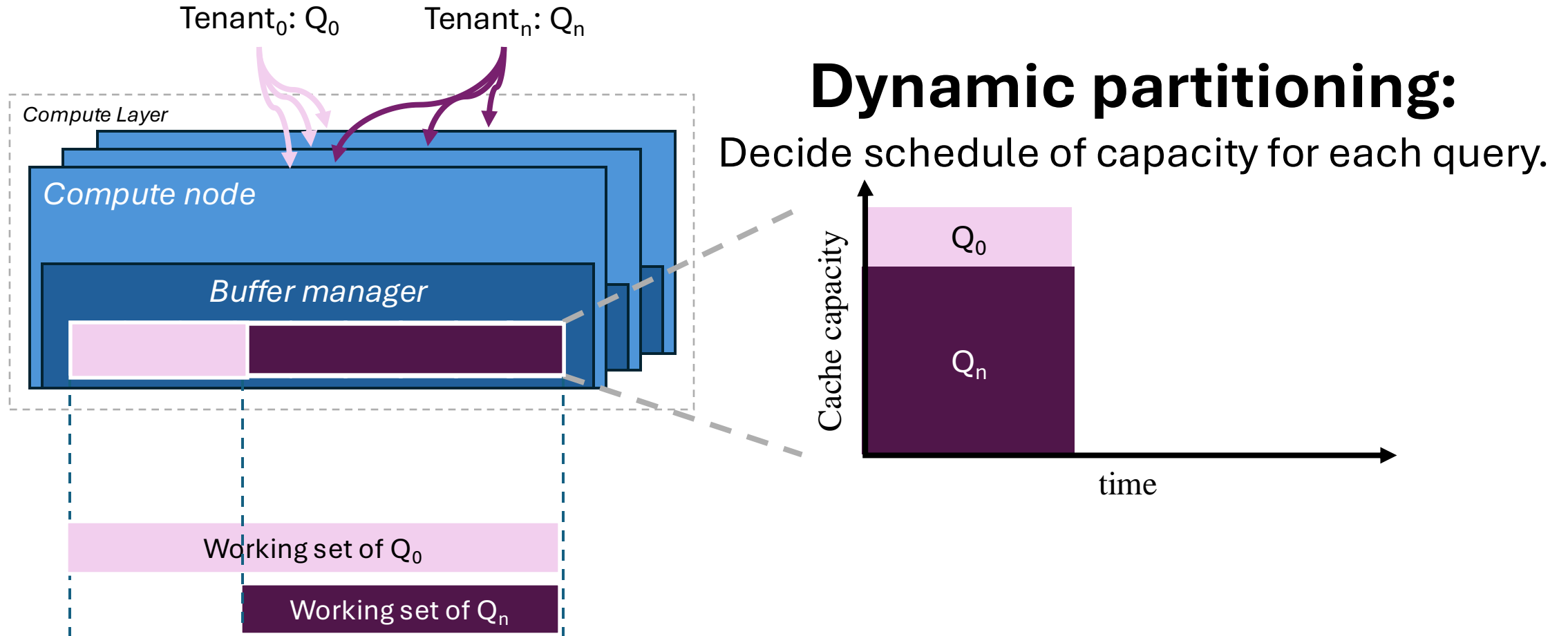
Decide page to evict when no capacity is left, via Least-Recently-Used (LRU), LFU, etc.

# Insight 1) Two subproblems in parallel paging: Page replacement + Dynamic partitioning

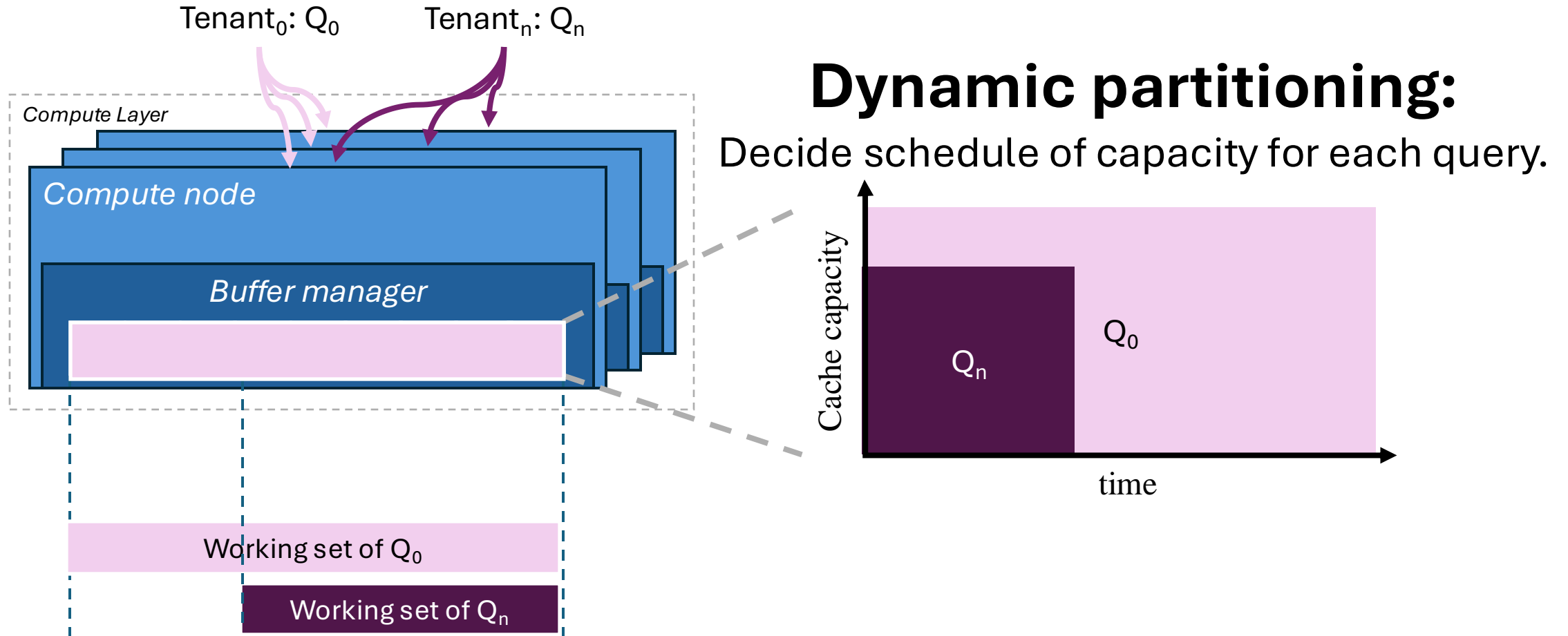




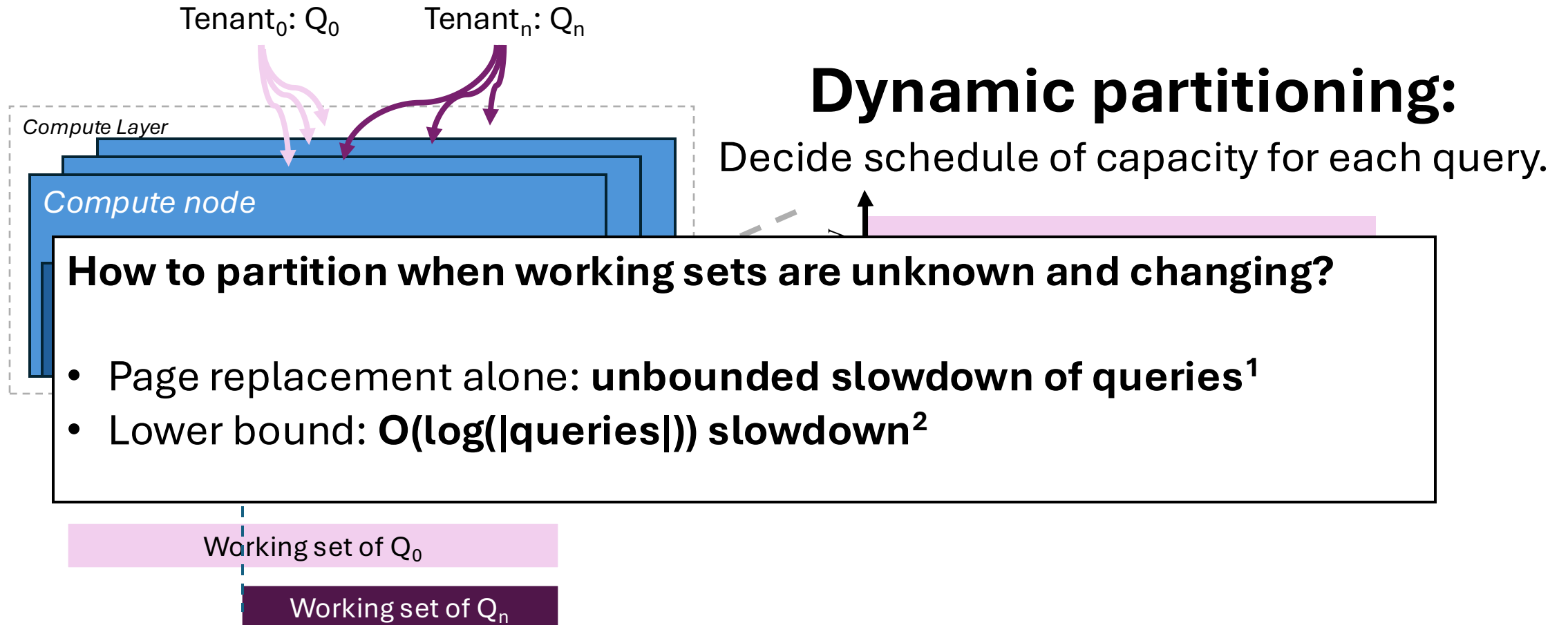
# Insight 1) Two subproblems in parallel paging: Page replacement + Dynamic partitioning



# Insight 1) Two subproblems in parallel paging: Page replacement + Dynamic partitioning



# Insight 1) Two subproblems in parallel paging: Page replacement + Dynamic partitioning



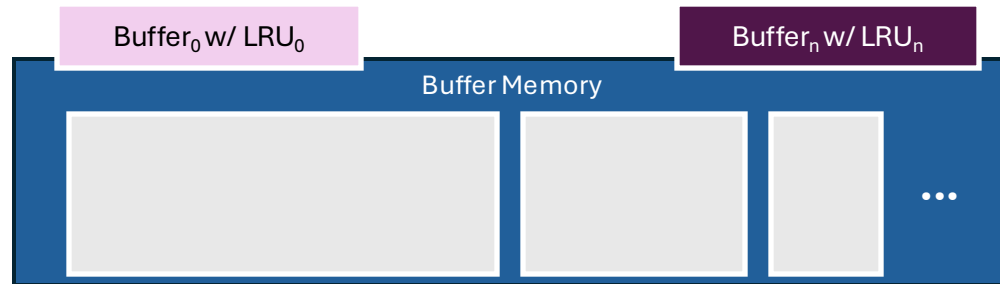
[1] S. Kamali and H. Xu, "Multicore Paging Algorithms Cannot Be Competitive," in *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

[2] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

Insight 2) Best Paper of SPAA'22 by Agrawal et al.<sup>2</sup>:

# First Optimal Algorithm in 2022 by Agrawal et al.<sup>2</sup>

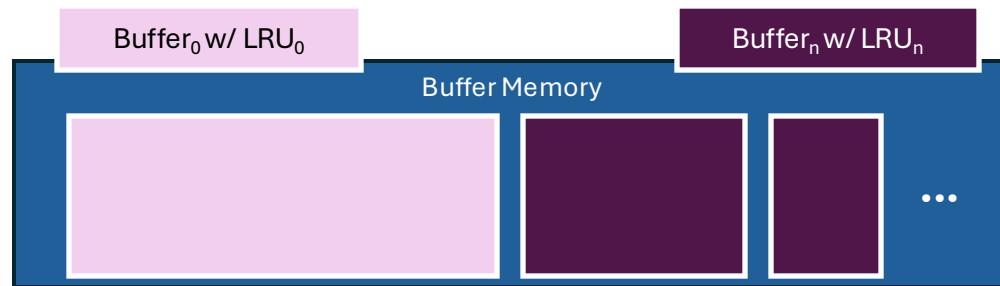
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

## Insight 2) **First Optimal** Algorithm in 2022 by Agrawal et al.<sup>1</sup>

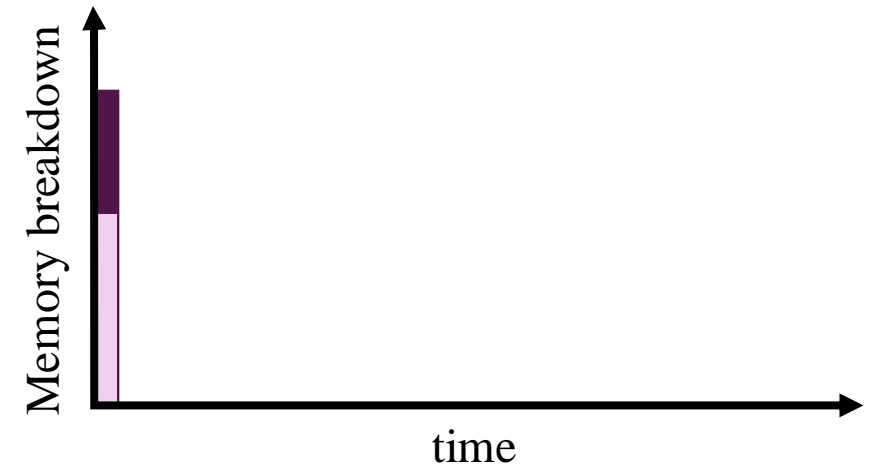
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

**Assign boxes to buffer managers for  $z_i$  time units**

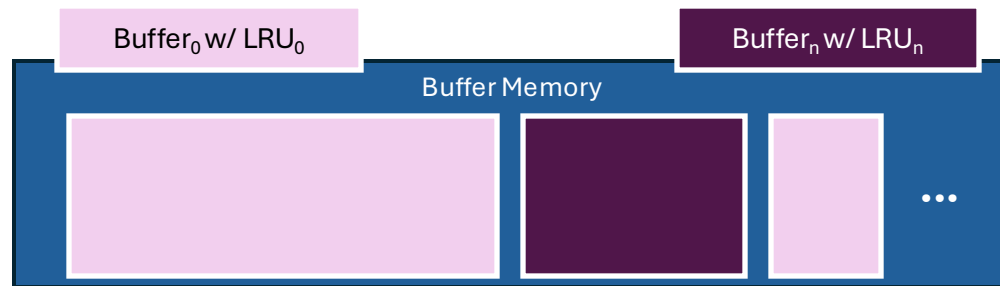
**Allocation schedule**



[1] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

## Insight 2) **First Optimal** Algorithm in 2022 by Agrawal et al.<sup>1</sup>

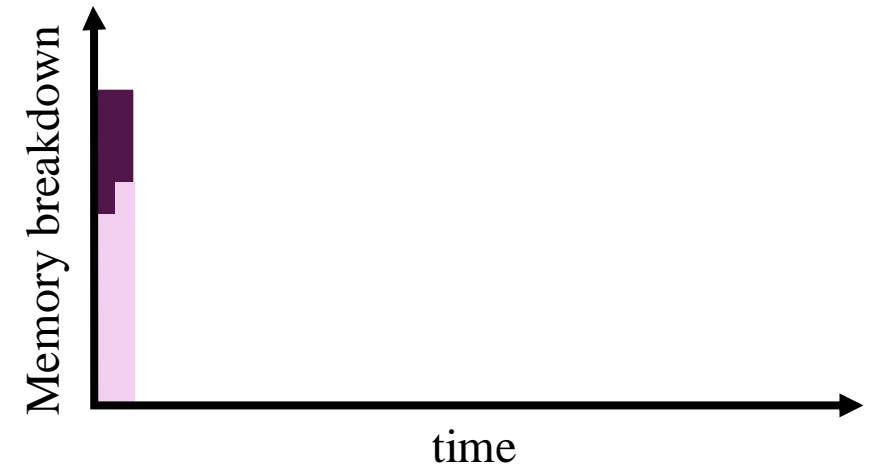
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

**Assign boxes to buffer managers for  $z_i$  time units**

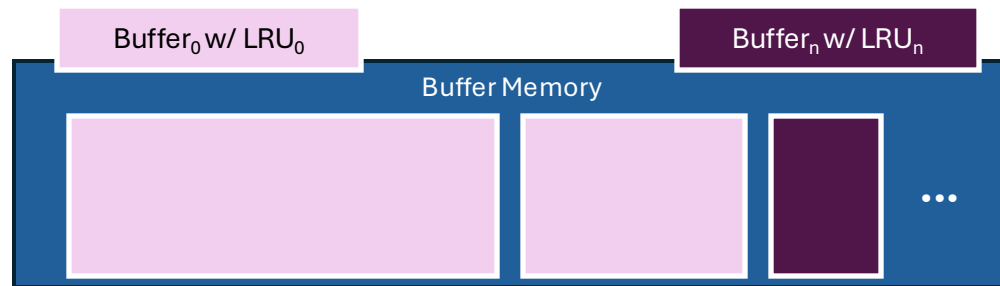
**Allocation schedule**



[1] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

## Insight 2) **First Optimal** Algorithm in 2022 by Agrawal et al.<sup>1</sup>

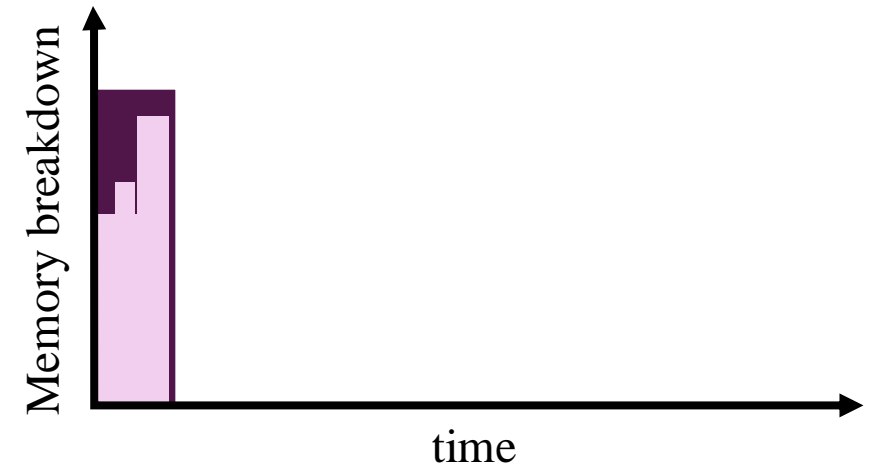
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

**Assign boxes to buffer managers for  $z_i$  time units**

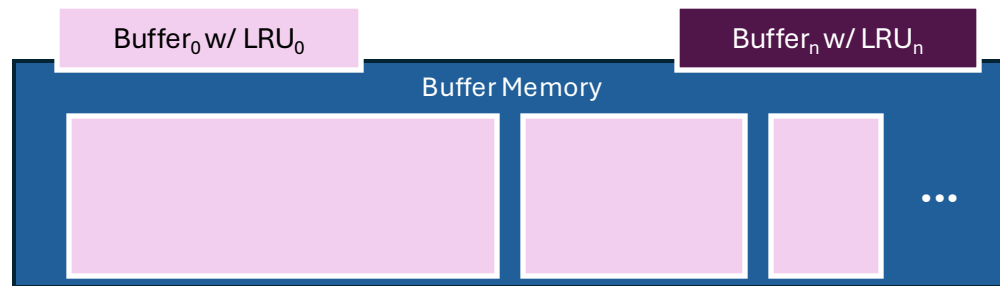
**Allocation schedule**



[1] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

## Insight 2) **First Optimal** Algorithm in 2022 by Agrawal et al.<sup>1</sup>

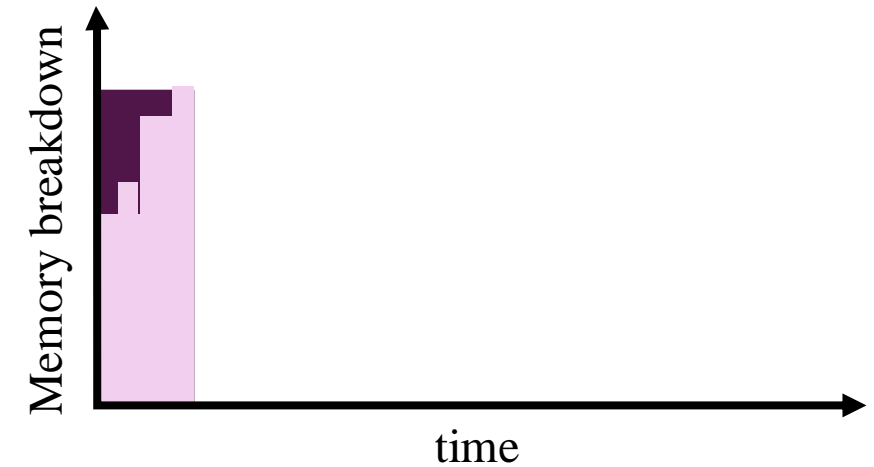
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

**Assign boxes to buffer managers for  $z_i$  time units**

**Allocation schedule**

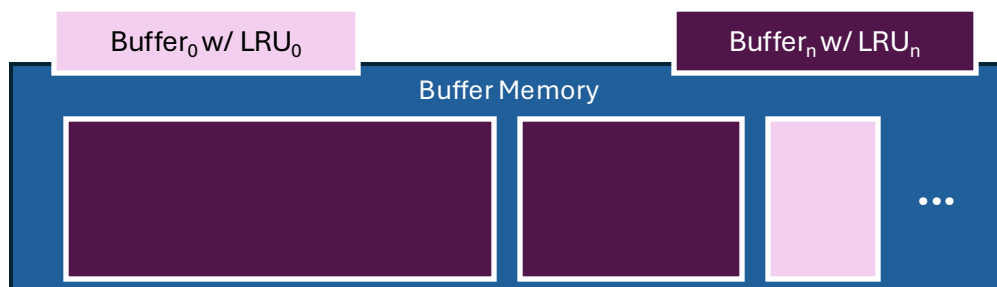


[1] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020



## Insight 2) **First Optimal** Algorithm in 2022 by Agrawal et al.<sup>1</sup>

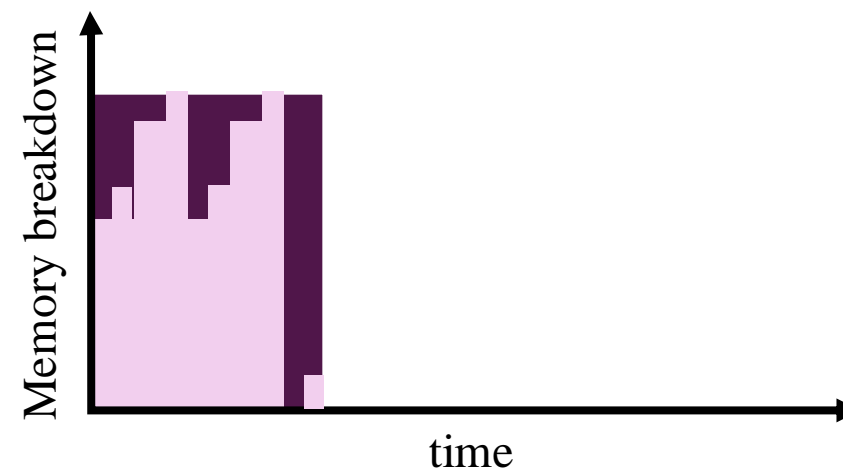
Variable-size buffer managers per query processor



Memory as boxes of size  $z_i = k, k/2, \dots$

Assign boxes to buffer managers for  $z_i$  time units

Allocation schedule



- **At most  $O(\log(|queries|))$  slower than OPT, no more thrashing!**
- **Compatible with every page replacement algorithm!**

[1] K. Agrawal, et al., "Online Parallel Paging with Optimal Makespan," *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

# Much to learn from this paper and caching theory...<sup>1,2</sup>

- **Memory-efficient (green) paging** as starting point
- Optimal random algorithm for parallel paging
- **Balancing memory- and time-efficiency** via well-roundedness
- **Optimal deterministic algorithm**
- ...

[1] S. Kamali and H. Xu, “Multicore Paging Algorithms Cannot Be Competitive,” in *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

[2] K. Agrawal, et al., “Online Parallel Paging with Optimal Makespan,” *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020

# More co-design required:

## **Implementation:**

- How to implement the partition schedule, for single and multi-node caches?

## **Benchmarking:**

- What is the performance for real workloads, with respect to under resource augmentation?

## **Extension:**

- How to address shared working sets and performance variability?

# Conclusion

LLMs make theory very accessible!

Let's expand the success story of co-design with theory to disaggregated data systems

Concrete Opportunities

- **Optimal caching**: no thrashing, coupling of partitioning + paging
- Sound coupling of workload forecasting and provisioning

# Resources

- [1] S. Kamali and H. Xu, “Multicore Paging Algorithms Cannot Be Competitive,” in *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020
- [2] K. Agrawal, et al., “Online Parallel Paging with Optimal Makespan,” *ACM Symposium on Parallelism in Algorithms and Architectures*, 2022