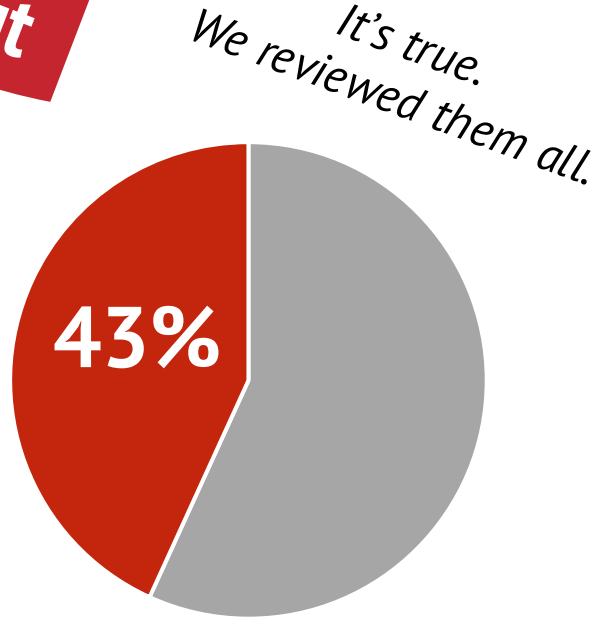
Principles and Methodologies for Serial Performance Optimization

Sujin Park Mingyu Guan Xiang Cheng Taesoo Kim

Georgia Institute of Technology

Georgia Tech

Performance: long-standing goal in systems community

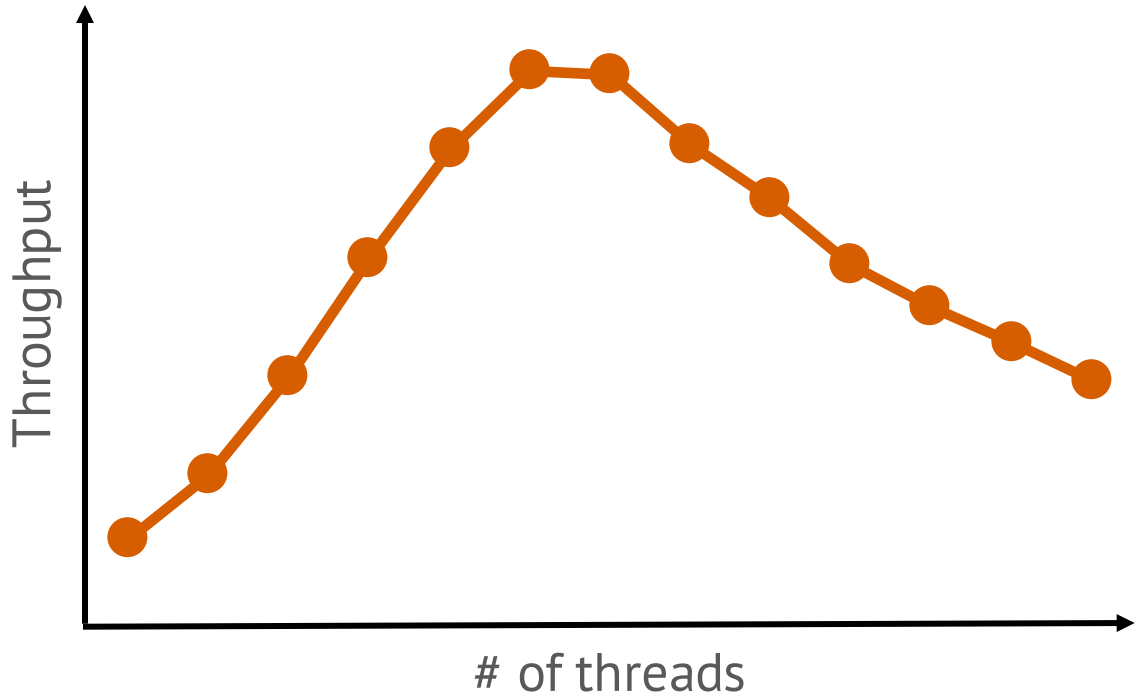


10 Years of OSDI/SOSP papers

Why this paper may feel different

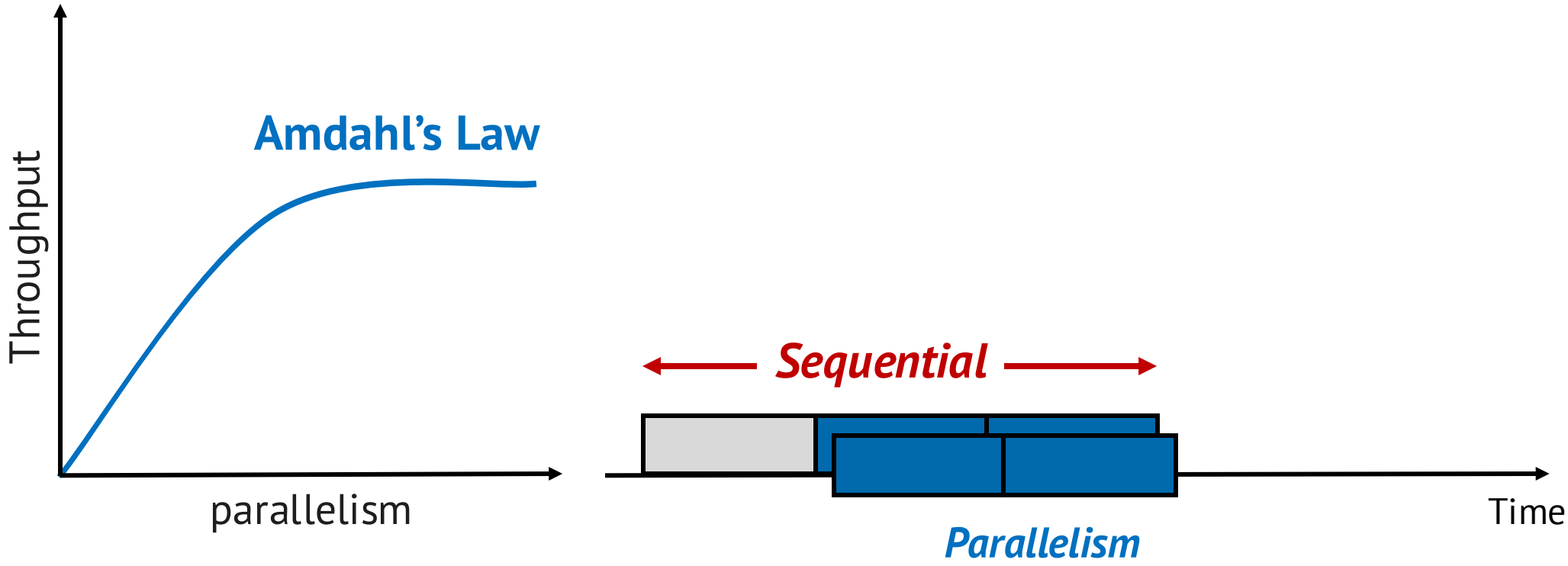
- This is ***not*** a paper about building a new system
- Instead, we ask a meta question:
  - ***"Can we systematize how sequential performance is optimized in practice?"***
- We reviewed 477 OSDI/SOSP papers over the past decade
- Distilled a **unified framework** of recurring optimization methodologies
- It's not a solution to one problem – It's a guide to solve many

You found a bottleneck. Now what?



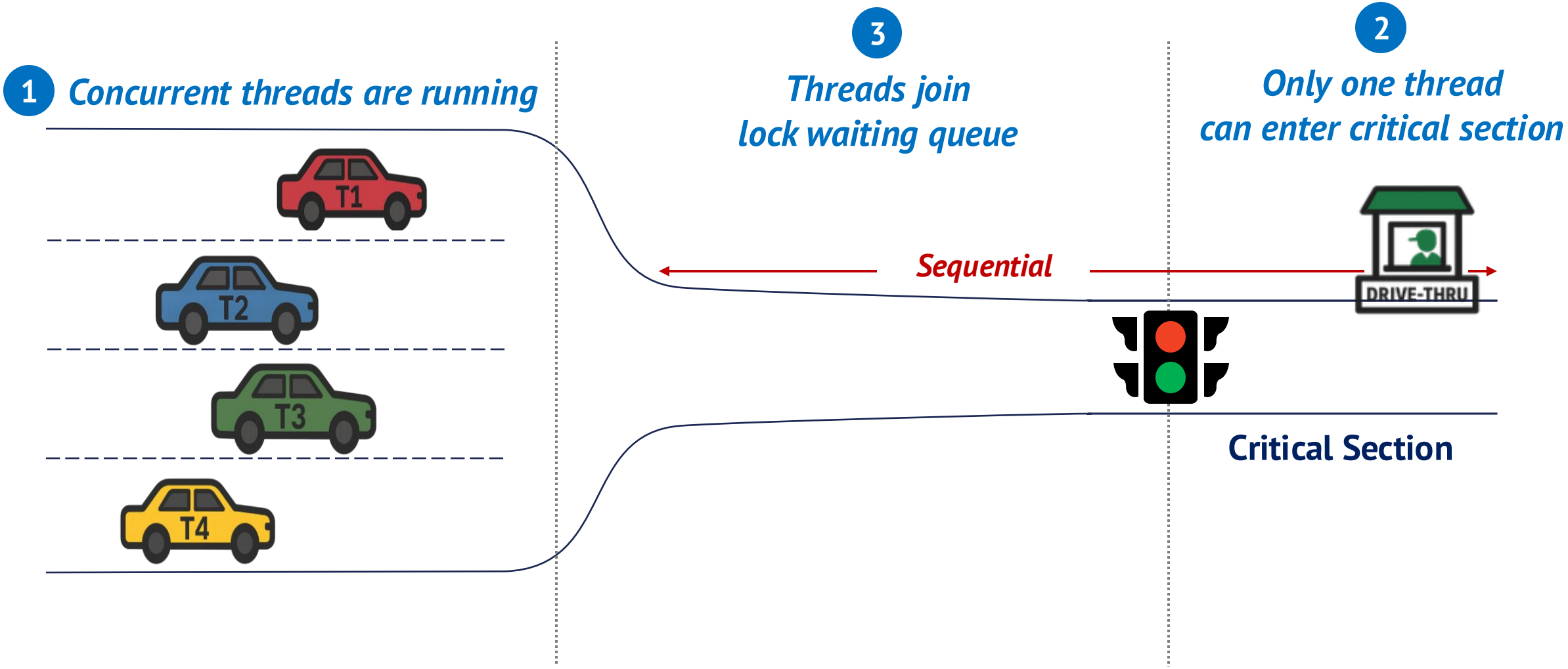
- Profiled to find a cause… and it turns out the bottleneck is lock contention!
- So how do you optimize this?

Parallelism helps… but not everything can be parallelized



The speedup is limited by the **sequential portion of the program**.

Sequential optimization remains crucial

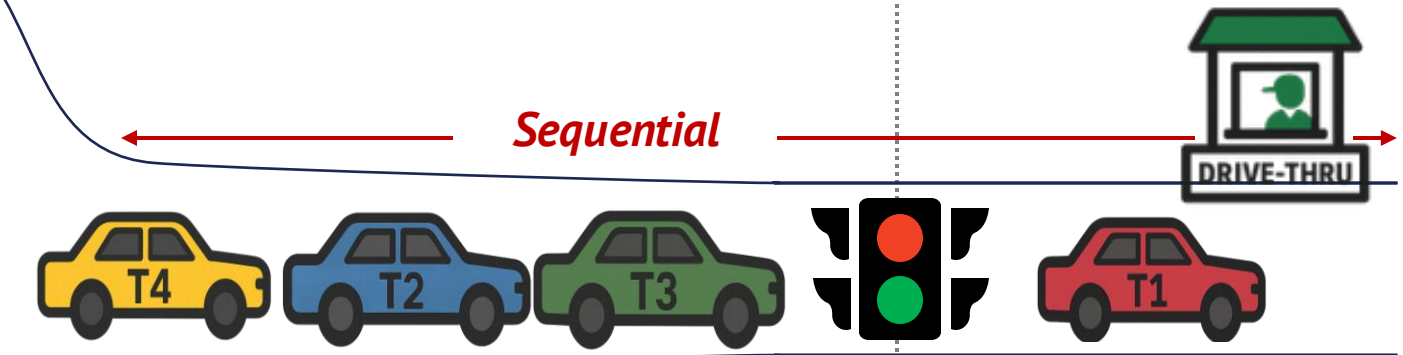


Sequential optimization remains crucial

1 *Concurrent threads are running*

3 *Threads join lock waiting queue*

2 *Only one thread can enter critical section*



Critical Section

To achieve higher throughput :

- **Reorder** waiting queue for faster progress (ShflLock, SOSP'19)
- **Batch** threads to better utilize underlying **cache** (CNA, EuroSys'19)
- Allow **dynamic** custom policy for reordering (SynCord, OSDI'22)

How to optimize sequential execution

🎯 Our questions

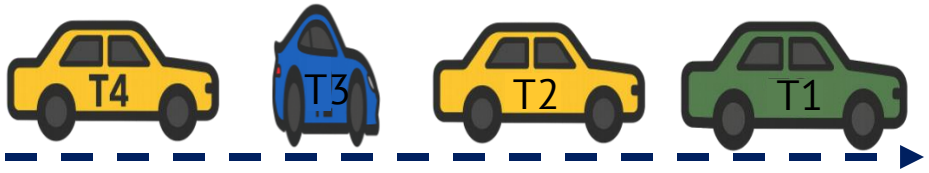
1. Can we **systematize** sequential optimization?
2. **How many** distinct approaches?
3. **When** to use each of them?

Three principles of sequential execution

To optimize existing sequence,

1. **Remove** a task

2. **Replace** with a faster one



3. **Reorder** tasks for better locality

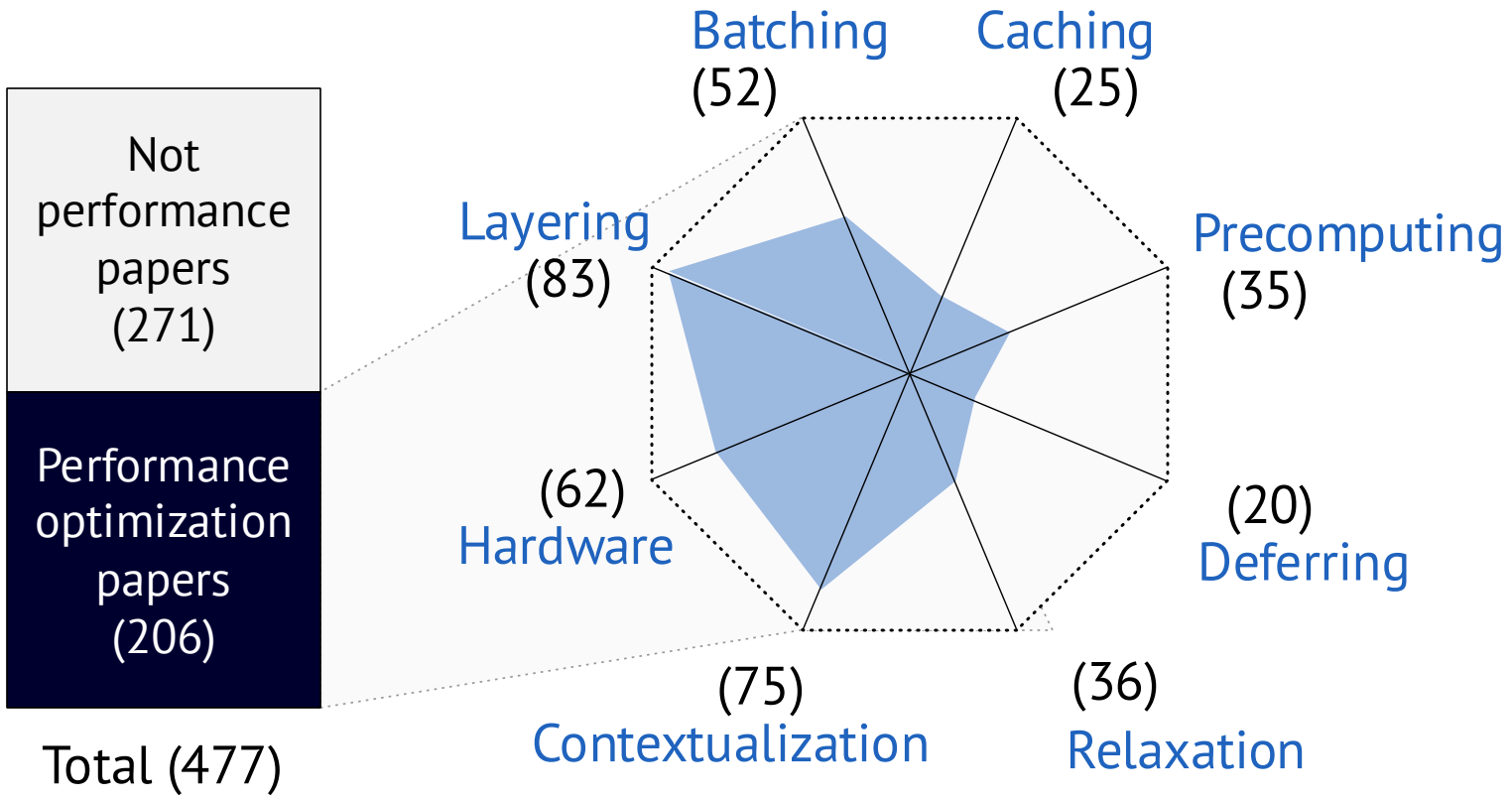
From principles to practice: The eight methodologies

Batching	**Caching**	**Precomputing**	**Deferring**
Relaxation	**Contextualization**	**Hardware**	**Layering**

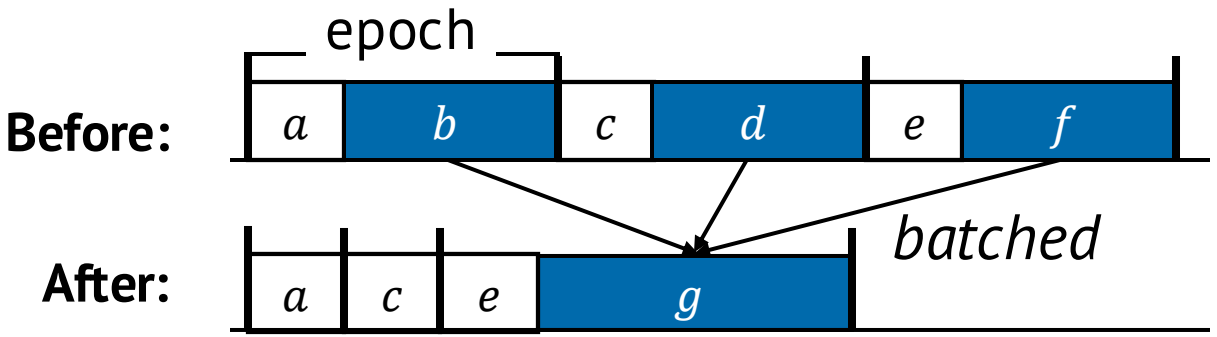
- **A set of common optimization patterns** for sequential performance
- Each methodology is derived from the three principles
- **Together, a unified way to understand and explore sequential optimization**
- Check our paper for definitions, visualizations, and real examples from OSDI/SOSP papers

Are they enough to cover common patterns?

Yes, we manually reviewed **10 years of OSDI & SOSP papers**
(477 papers)



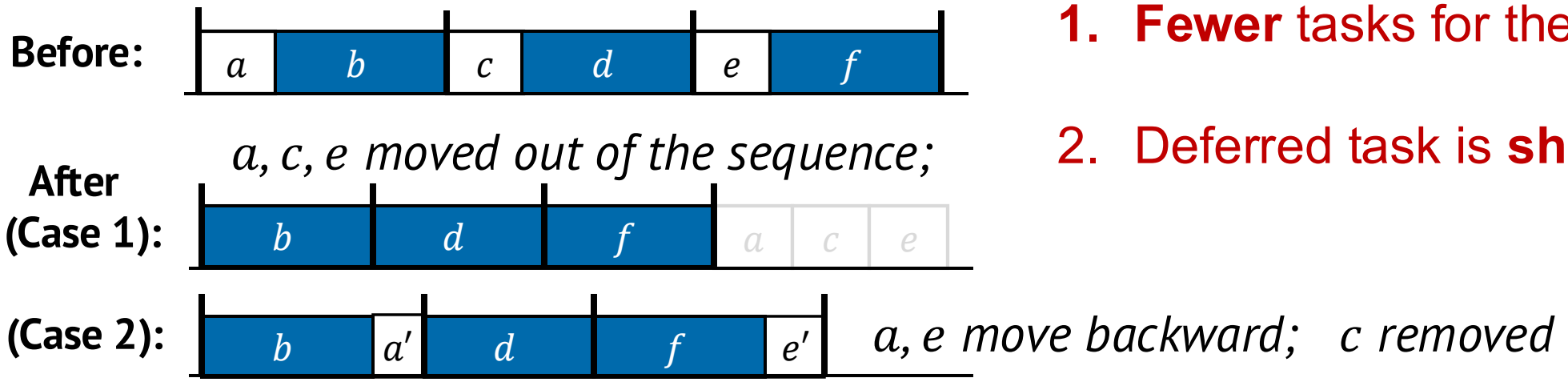
Batching



1. **Fewer** tasks after batched
2. Batched task is **shorter**

1. Coalesced calls — remove, replace
   - When each task incurs expensive cost every time it's called
2. Discard stale tasks at the time of batched request — remove
   - Batching inherently defers earlier tasks to make a batch
   - E.g., group commit, write buffer
3. Maximize spatial and temporal locality — reorder

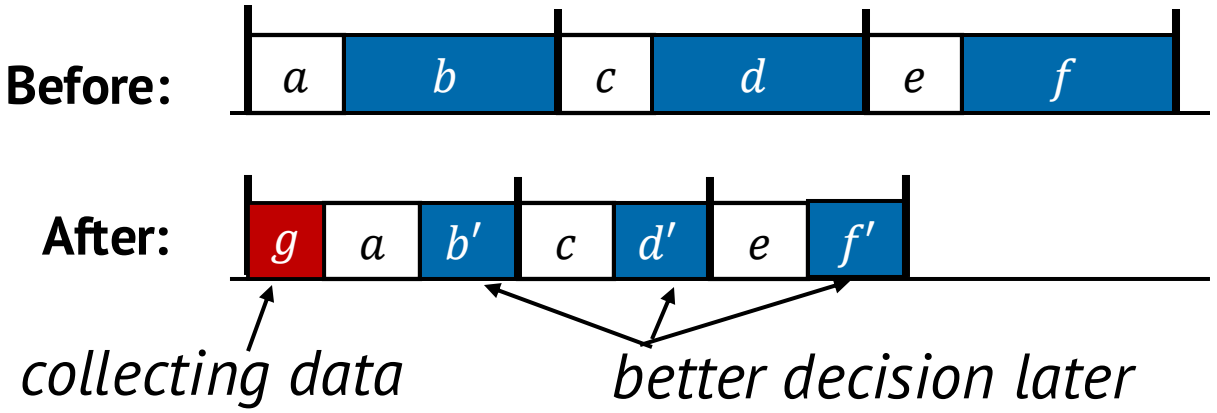
Deferring



1. **Fewer** tasks for the sequence
2. Deferred task is **shorter**

1. Move tasks out of a sequence and use later resources — remove
2. Reorder tasks within a sequence for better decision or locality — reorder
3. Based on two types of speculation
   - (1) the task might not be needed later, (2) it might be handled more efficiently if delayed

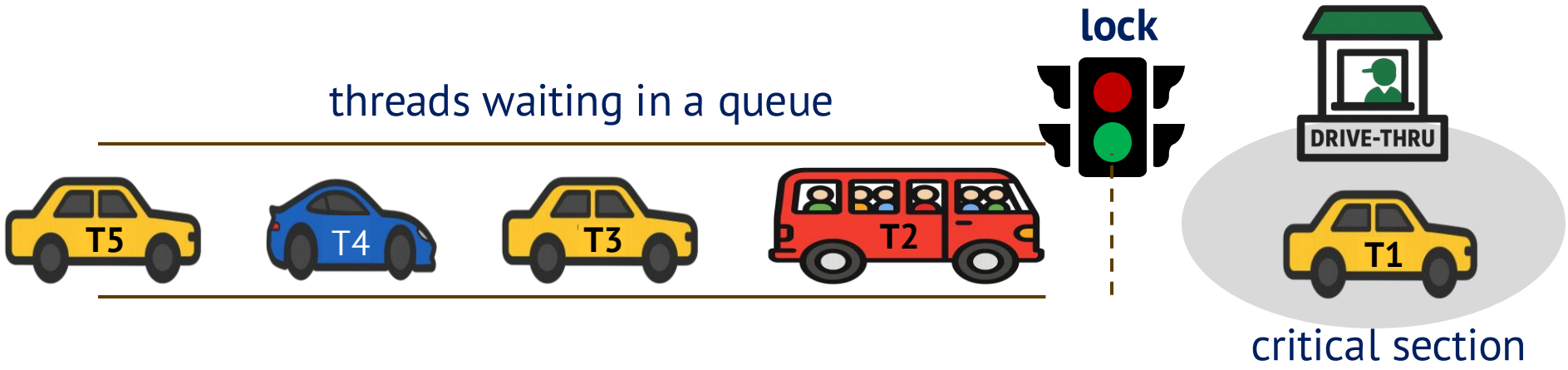
Contextualization



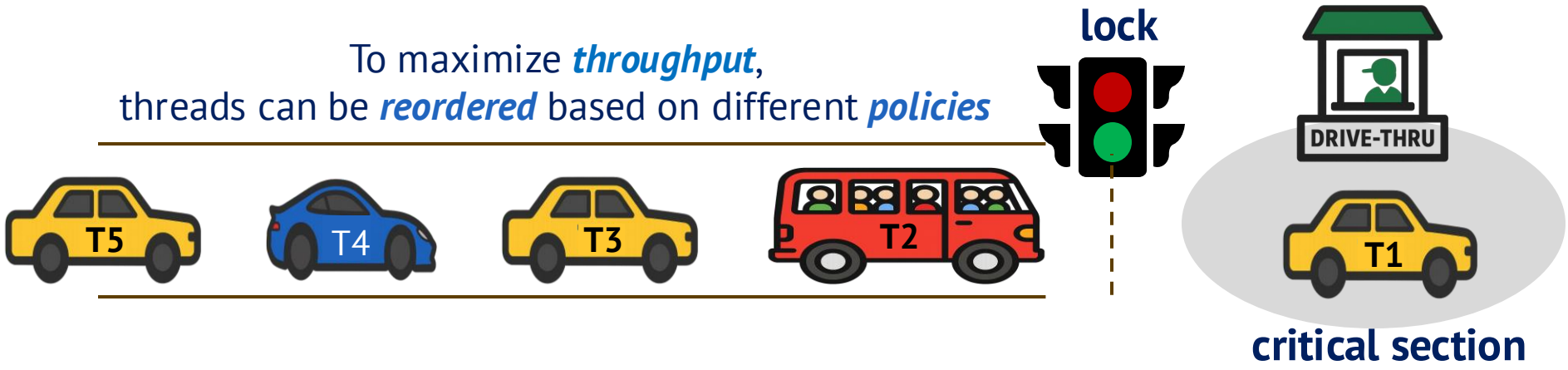
Overhead < **Later optimization**

1. Collect runtime context and make workload-specific decision replace
2. Move the runtime analysis off the critical path if possible

Applying methodologies to a real problem



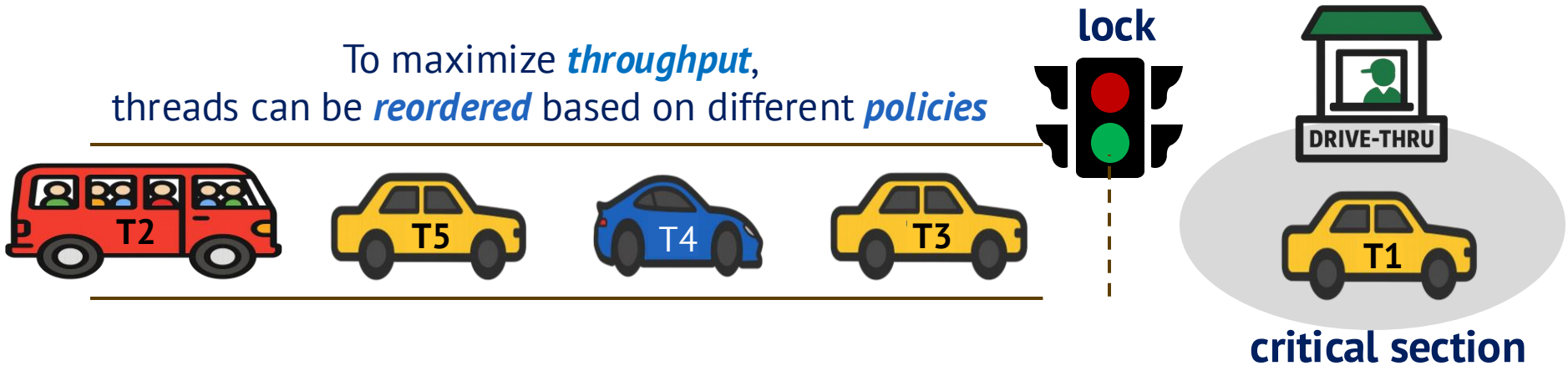
Applying methodologies to a real problem



SynCord (OSDI '22)

Applying methodologies to a real problem

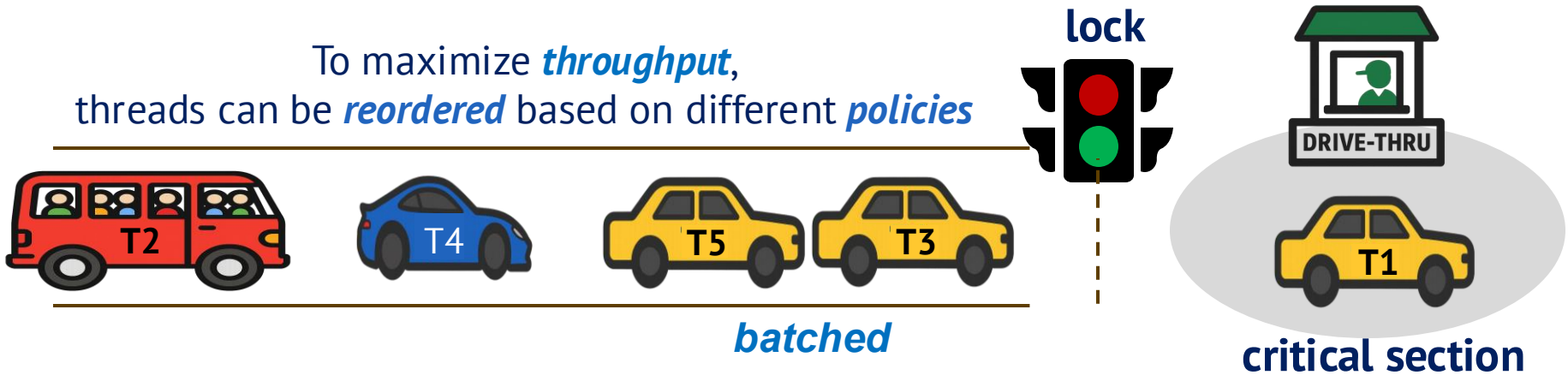
To maximize ***throughput***,
threads can be ***reordered*** based on different ***policies***



SynCord (OSDI '22)

- ***Deferring*** – *move T2 later in the queue*
- ***Contextualization*** – *collect runtime context*

Applying methodologies to a real problem

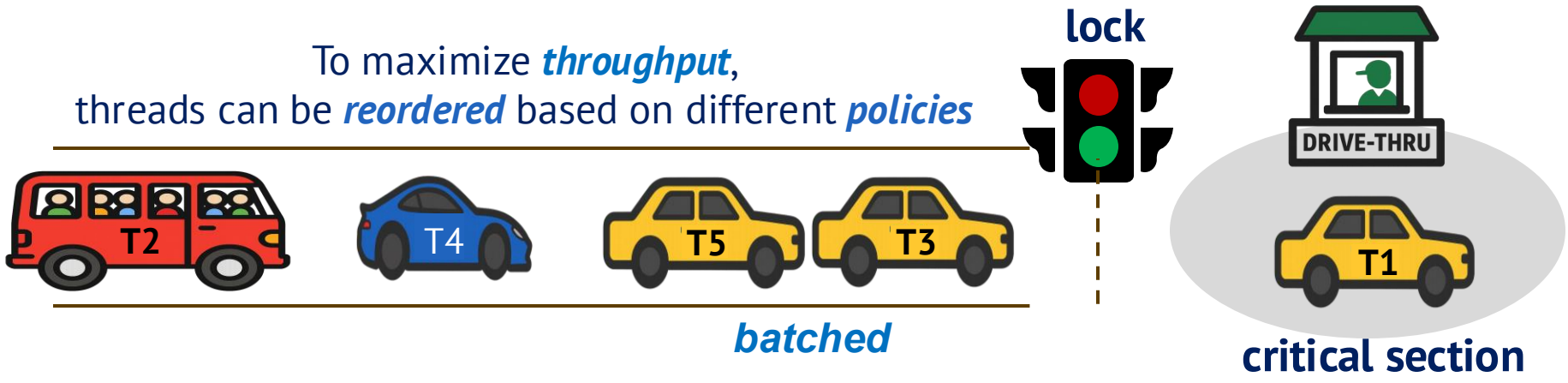


SynCord (OSDI '22)

- ***Deferring*** – *move T2 later in the queue*
- ***Contextualization*** – *collect runtime context*
- ***Batching*** – *group threads from same socket*

Applying methodologies to a real problem

To maximize ***throughput***,
threads can be ***reordered*** based on different ***policies***



SynCord (OSDI '22)

- ***Deferring*** – *move T2 later in the queue*
- ***Contextualization*** – *collect runtime context*
- ***Batching*** – *group threads from same socket*

More optimization!

- ***Caching*** ⟶ ***+23.8% throughput***
  cache auxiliary data into per-thread cache line
- ***Delayering*** ⟶ ***+14.4% throughput***
  Directly run policy code once it passes the verifier

Conclusion

- Sequential tasks are basic building blocks for performance optimization
- A systematic framework to guide sequential optimization

Human

https://persona0220.github.io/performance-book/

AI agent

https://github.com/sslab-gatech/SysGPT

It really captures ideas from the latest OSDI/SOSP papers

even those it was not trained on!

Questions