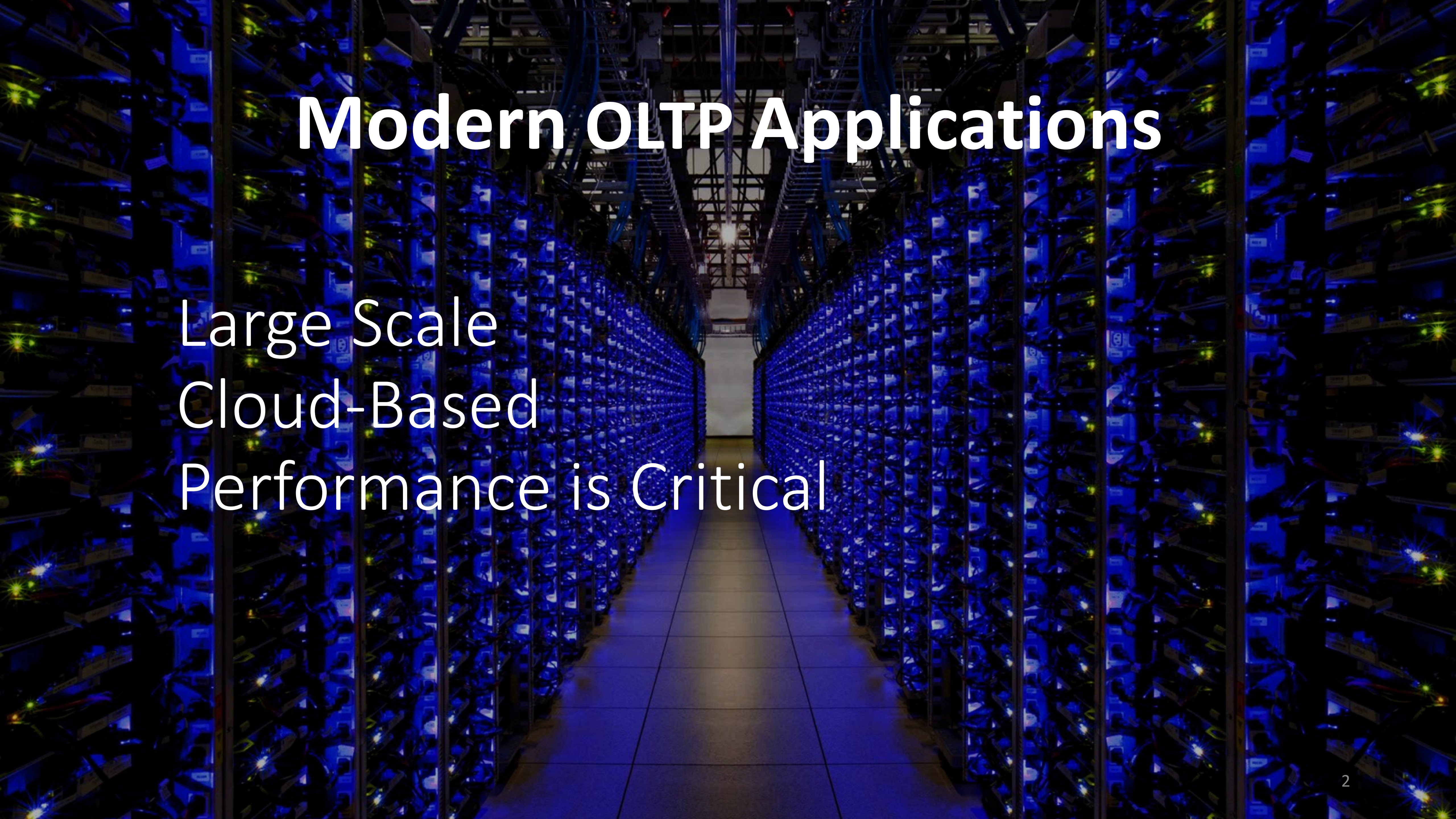
Predictive Elastic Database Systems

Rebecca Taft – becca@cockroachlabs.com

HPTS 2017



Modern OLTP Applications

Large Scale
Cloud-Based
Performance is Critical

Challenges to transaction performance: skew and workload variation

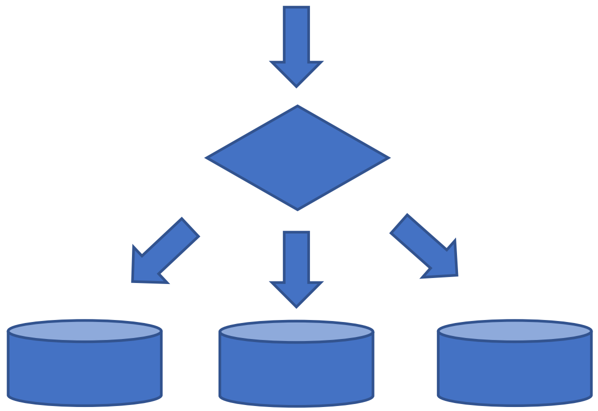
Database Elasticity

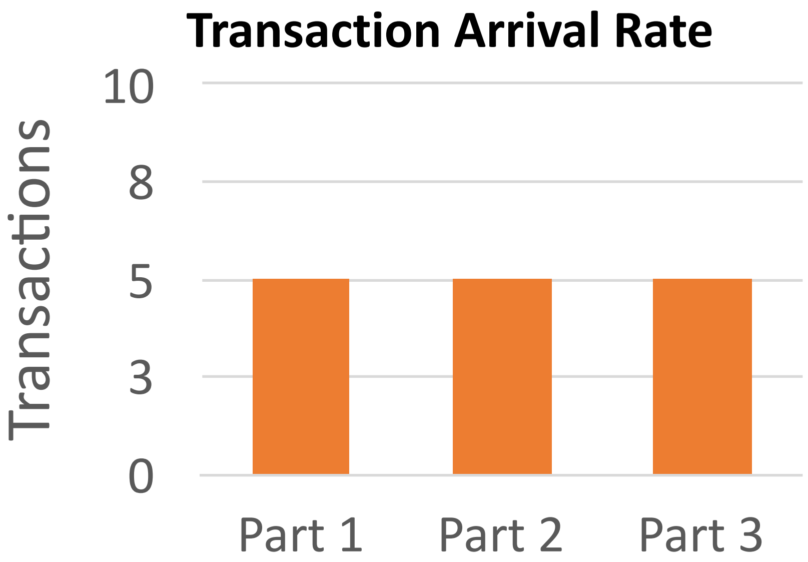
E-Store – manage skew and react to variation

P-Store – predictive modeling for time variation

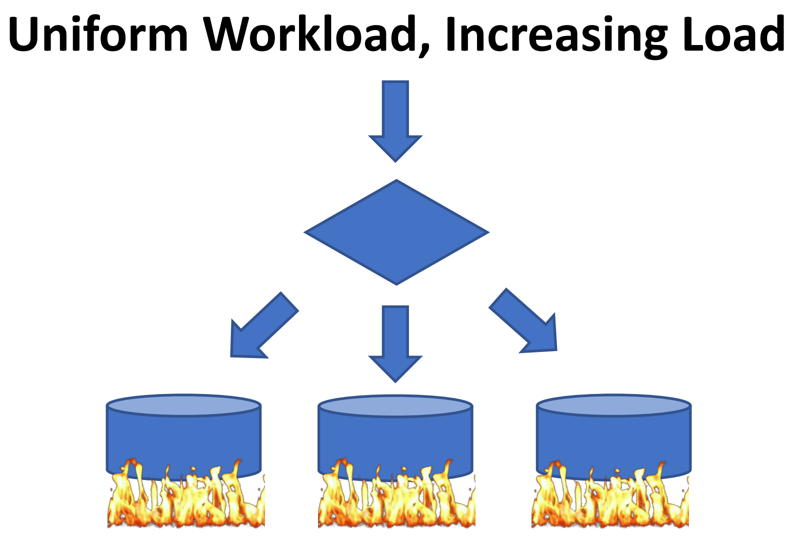
E-Store: Elastic Scaling to Adapt to Workload

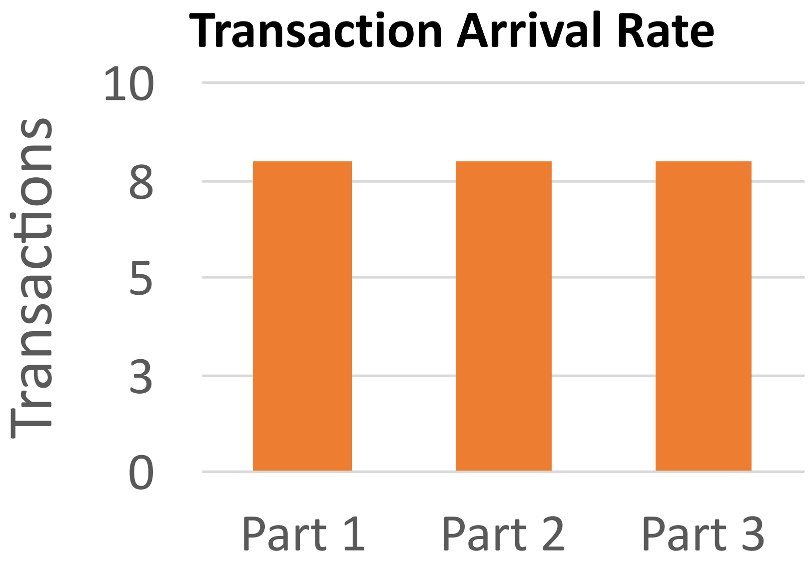
Uniform Workload, Increasing Load





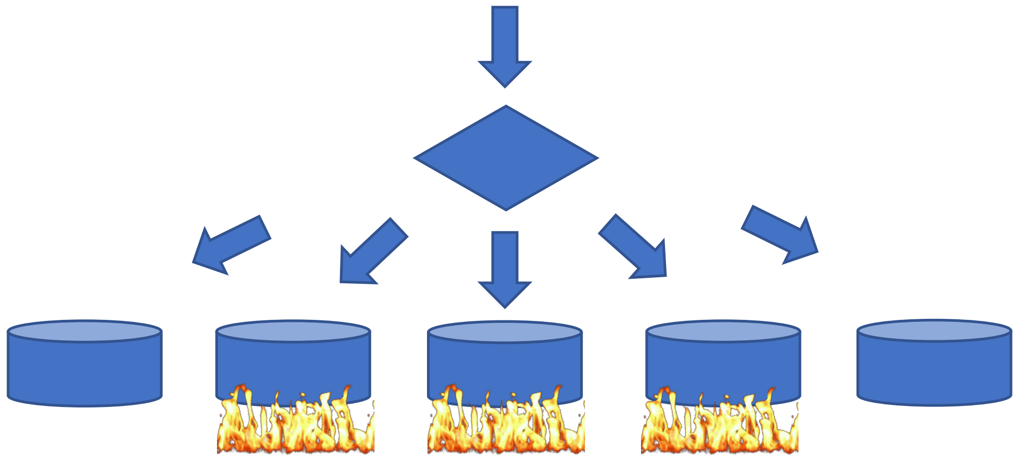
E-Store: Elastic Scaling to Adapt to Workload



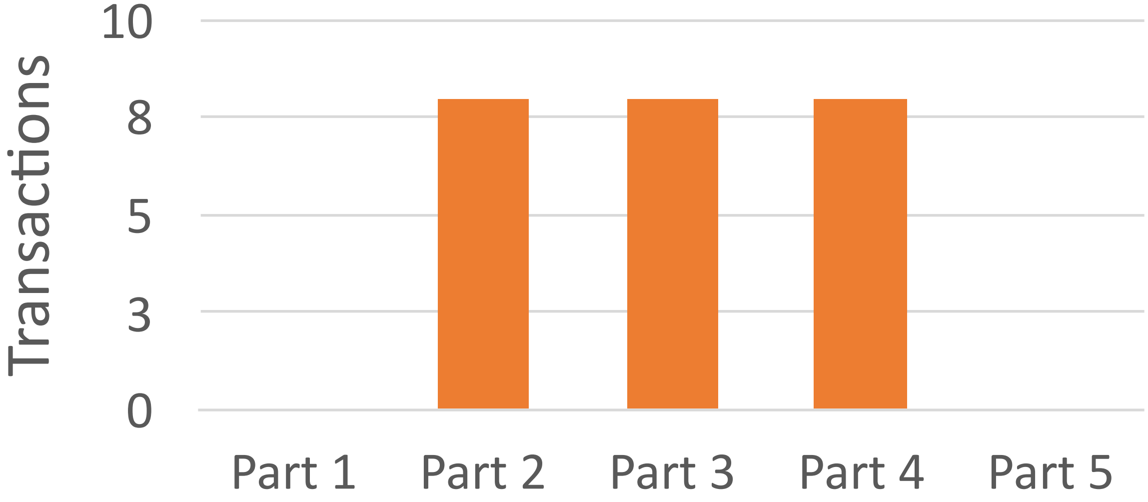


E-Store: Elastic Scaling to Adapt to Workload

Uniform Workload, Increasing Load

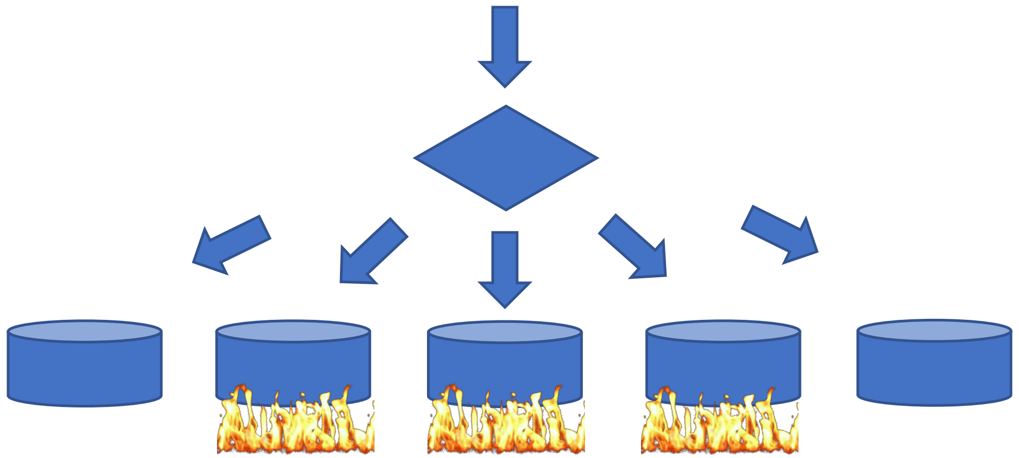


Transaction Arrival Rate

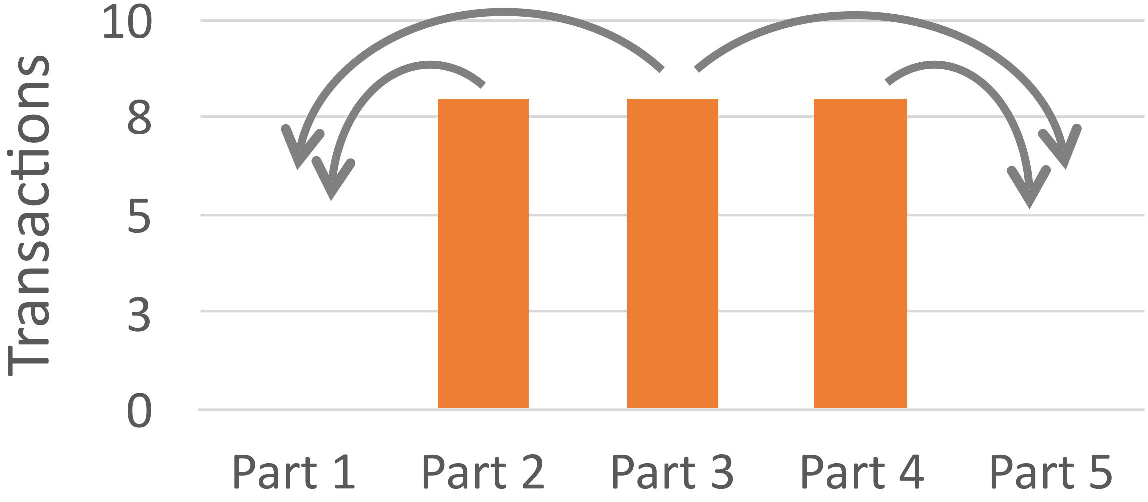


E-Store: Elastic Scaling to Adapt to Workload

Uniform Workload, Increasing Load

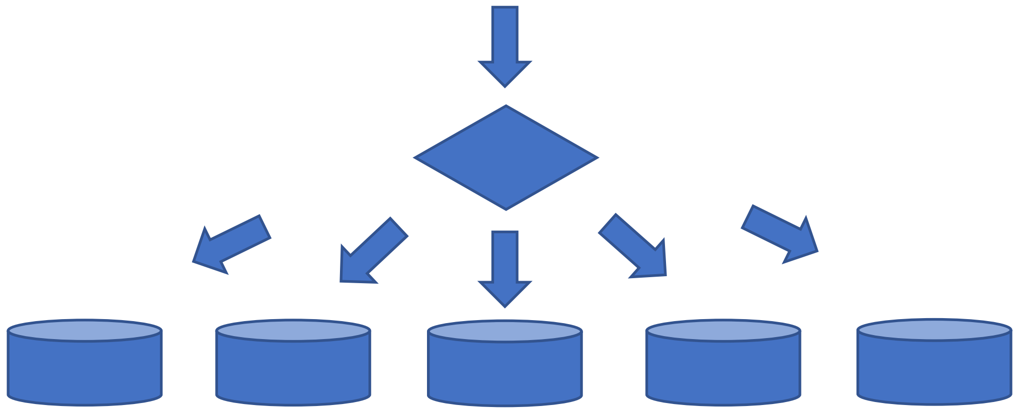


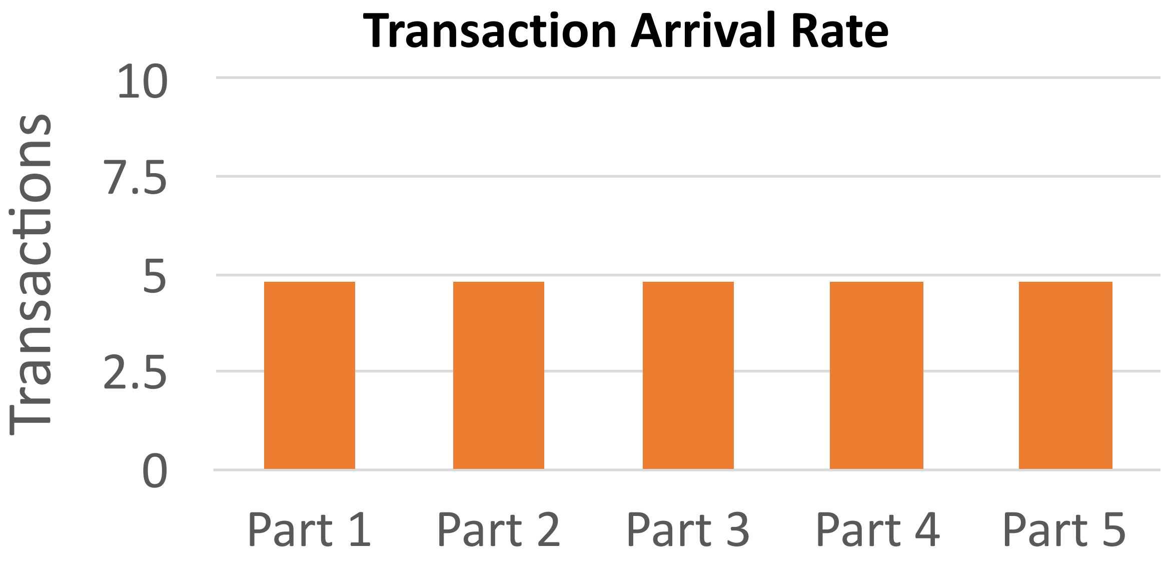
Transaction Arrival Rate



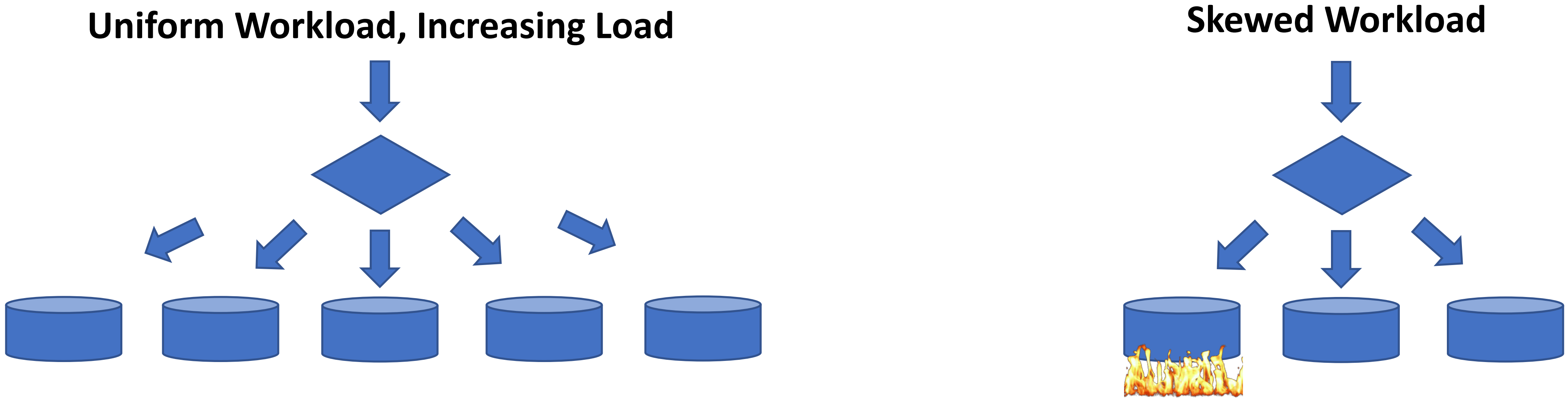
E-Store: Elastic Scaling to Adapt to Workload

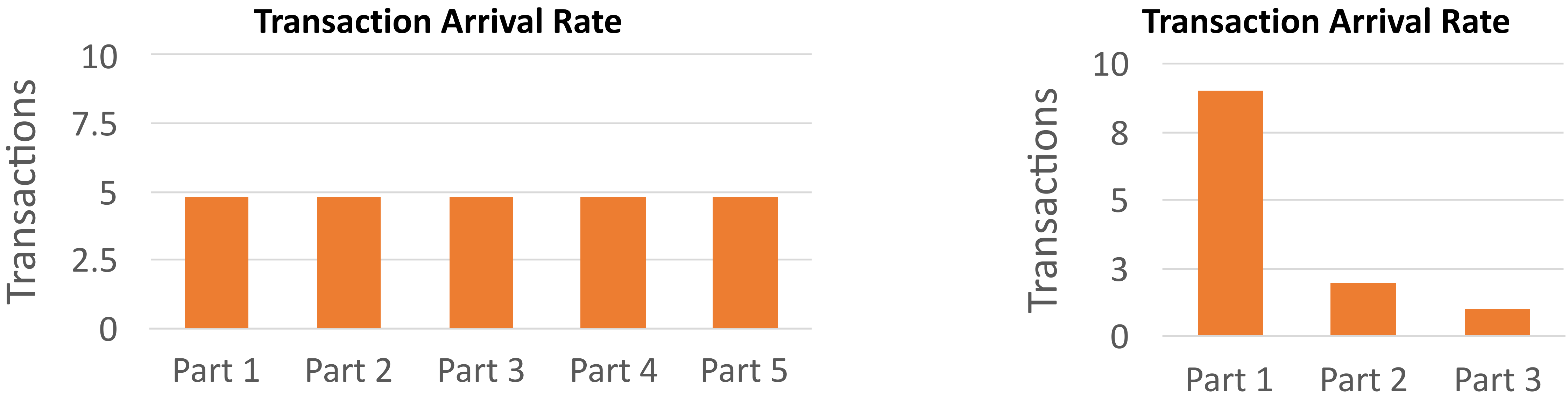
Uniform Workload, Increasing Load





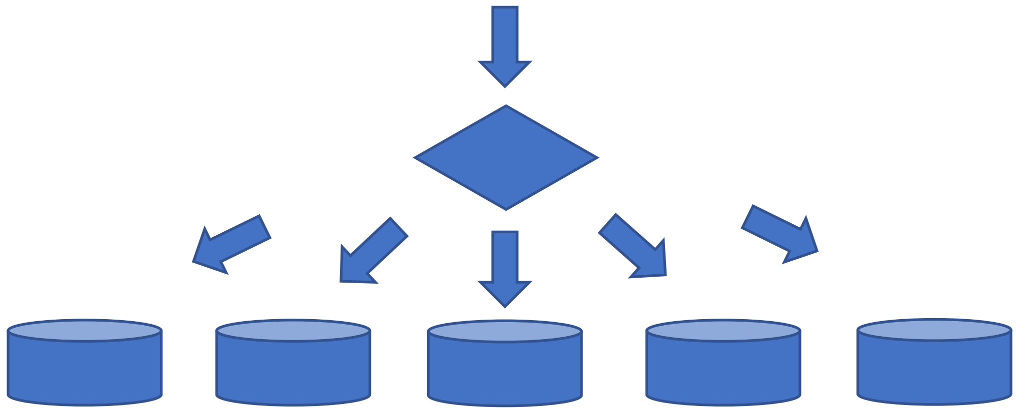
E-Store: Elastic Scaling to Adapt to Workload



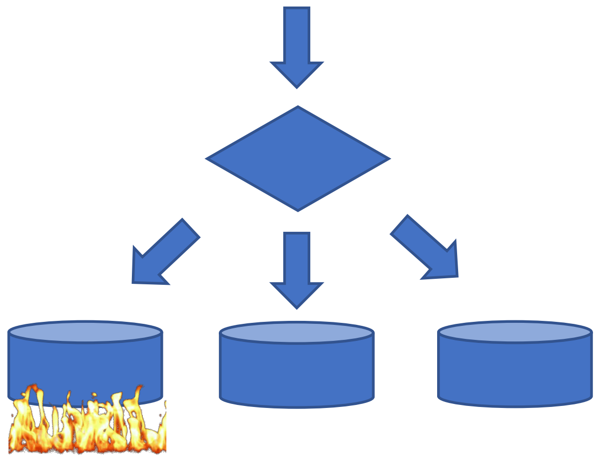


E-Store: Elastic Scaling to Adapt to Workload

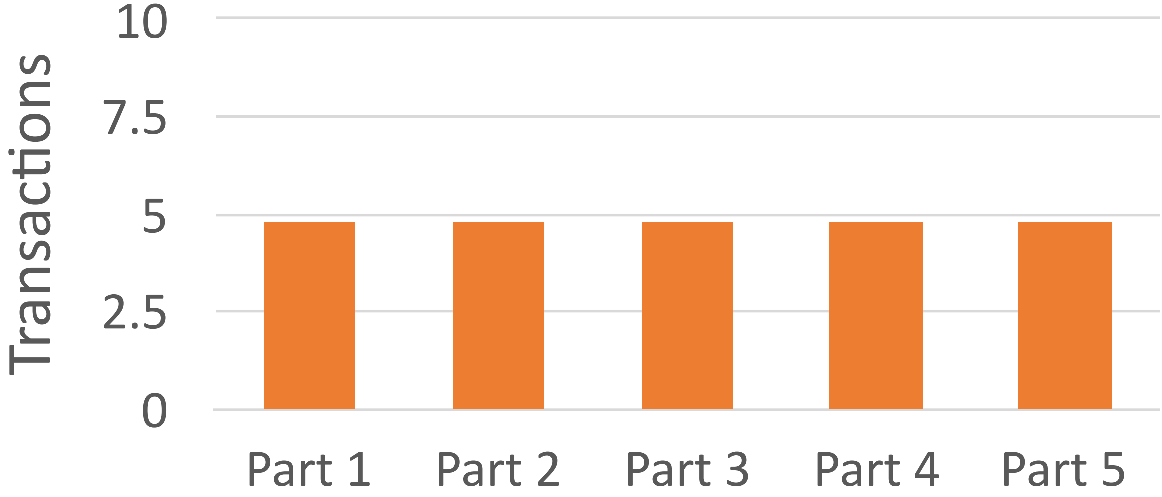
Uniform Workload, Increasing Load



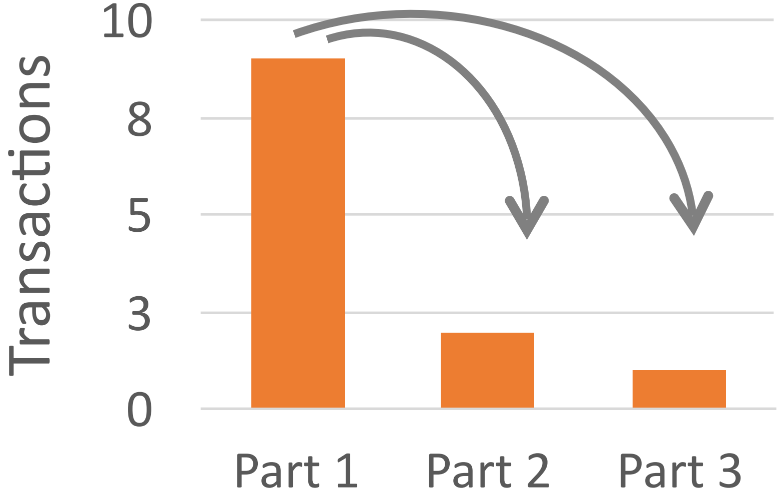
Skewed Workload



Transaction Arrival Rate

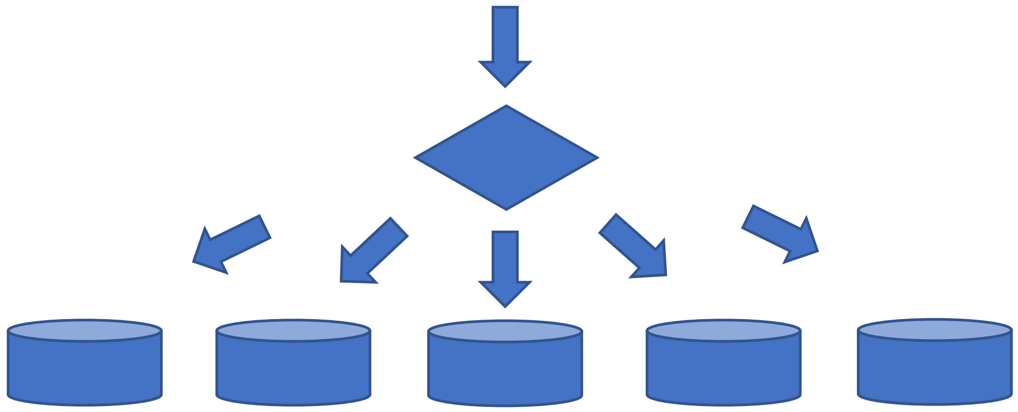


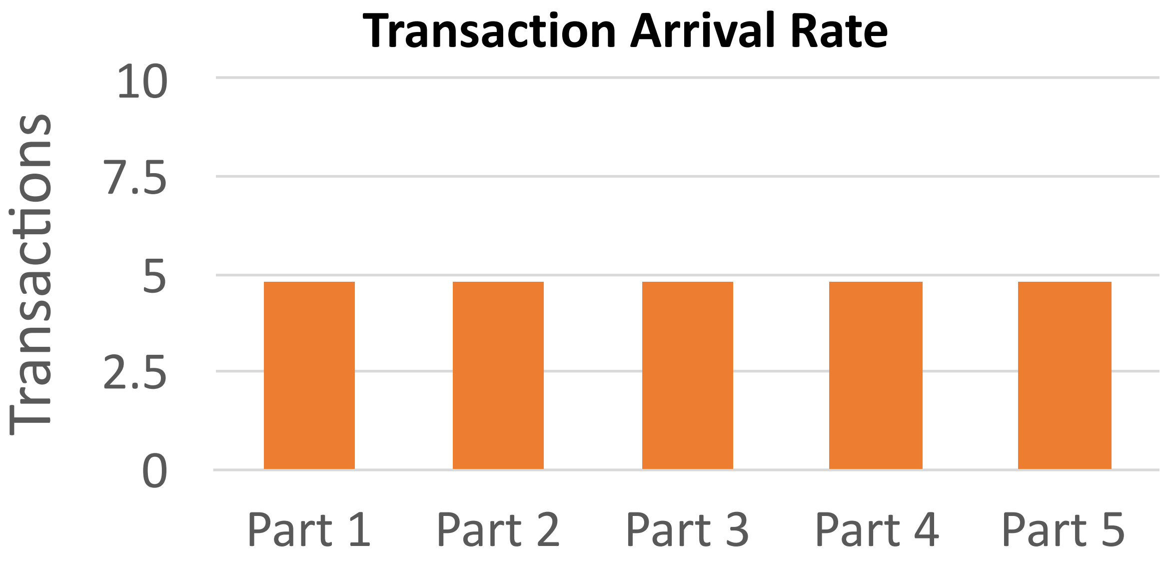
Transaction Arrival Rate



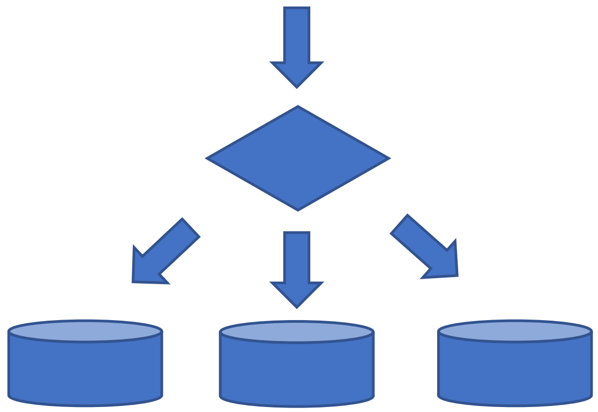
E-Store: Elastic Scaling to Adapt to Workload

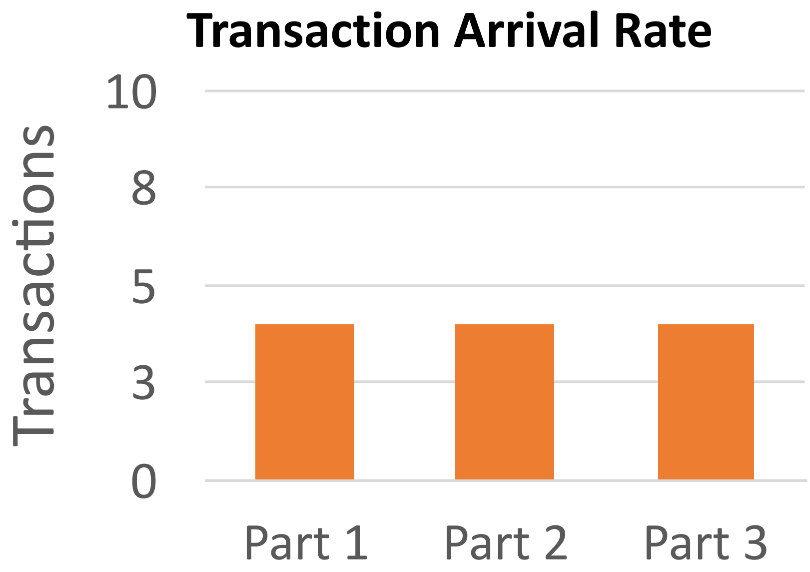
Uniform Workload, Increasing Load



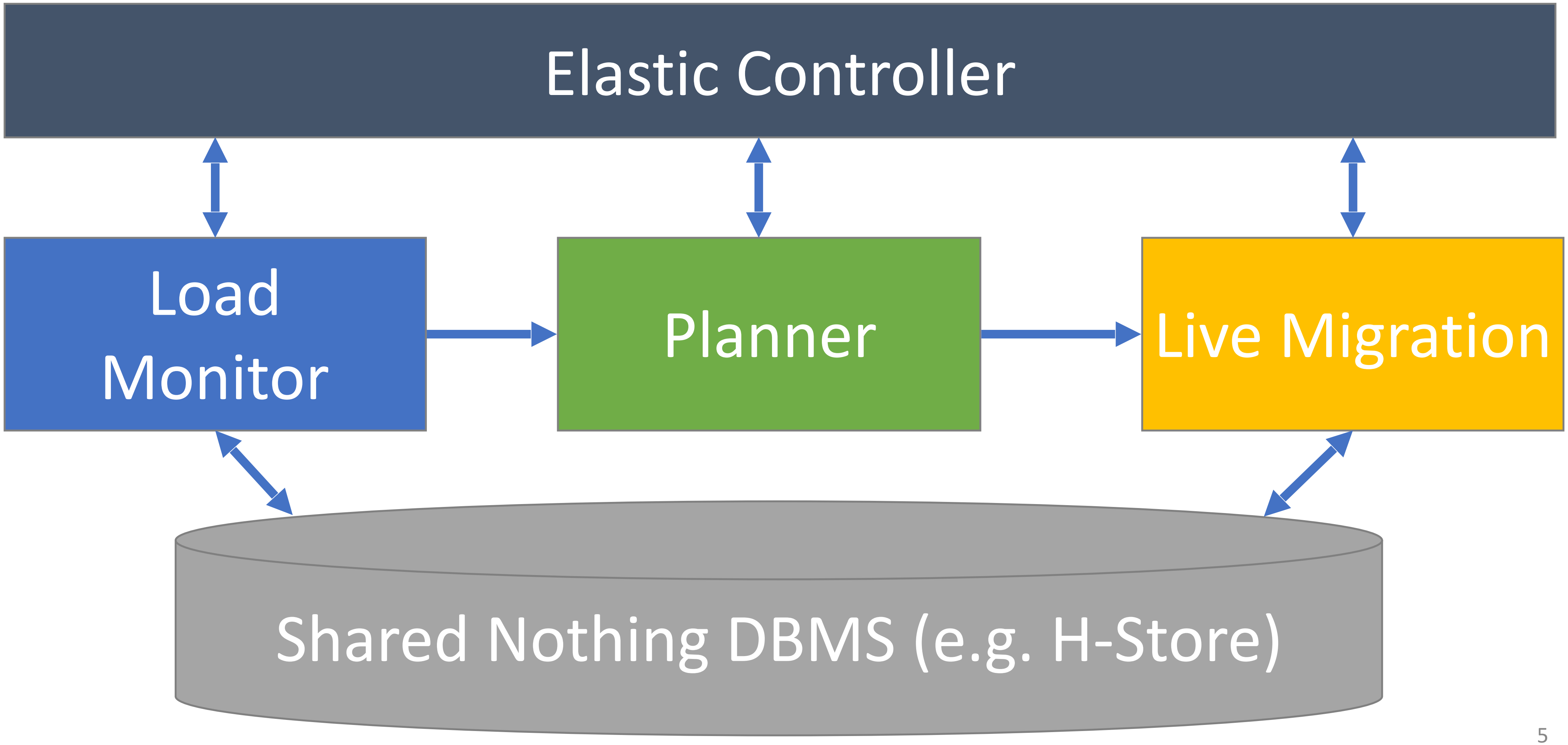


Skewed Workload



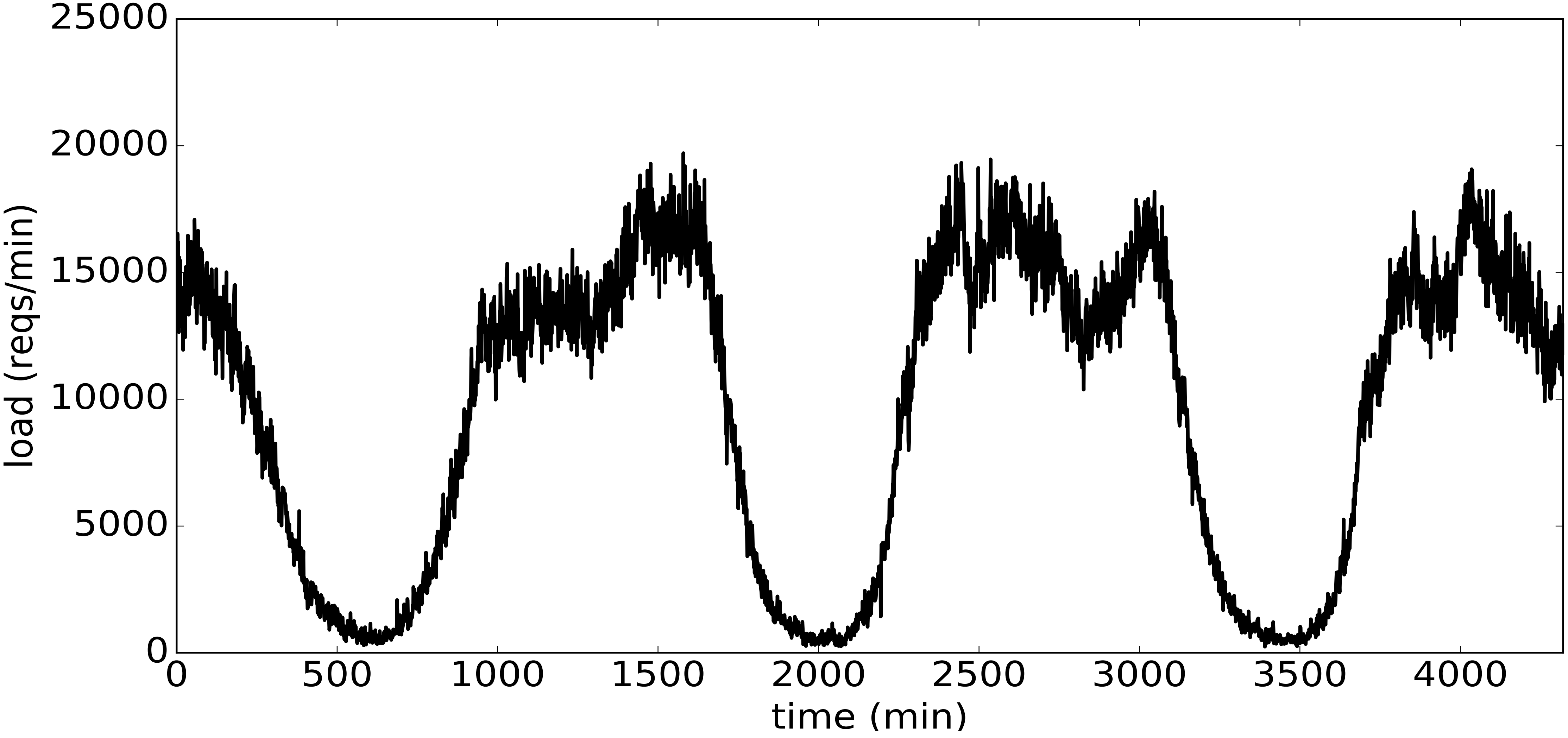


E-Store Elastic DBMS Architecture

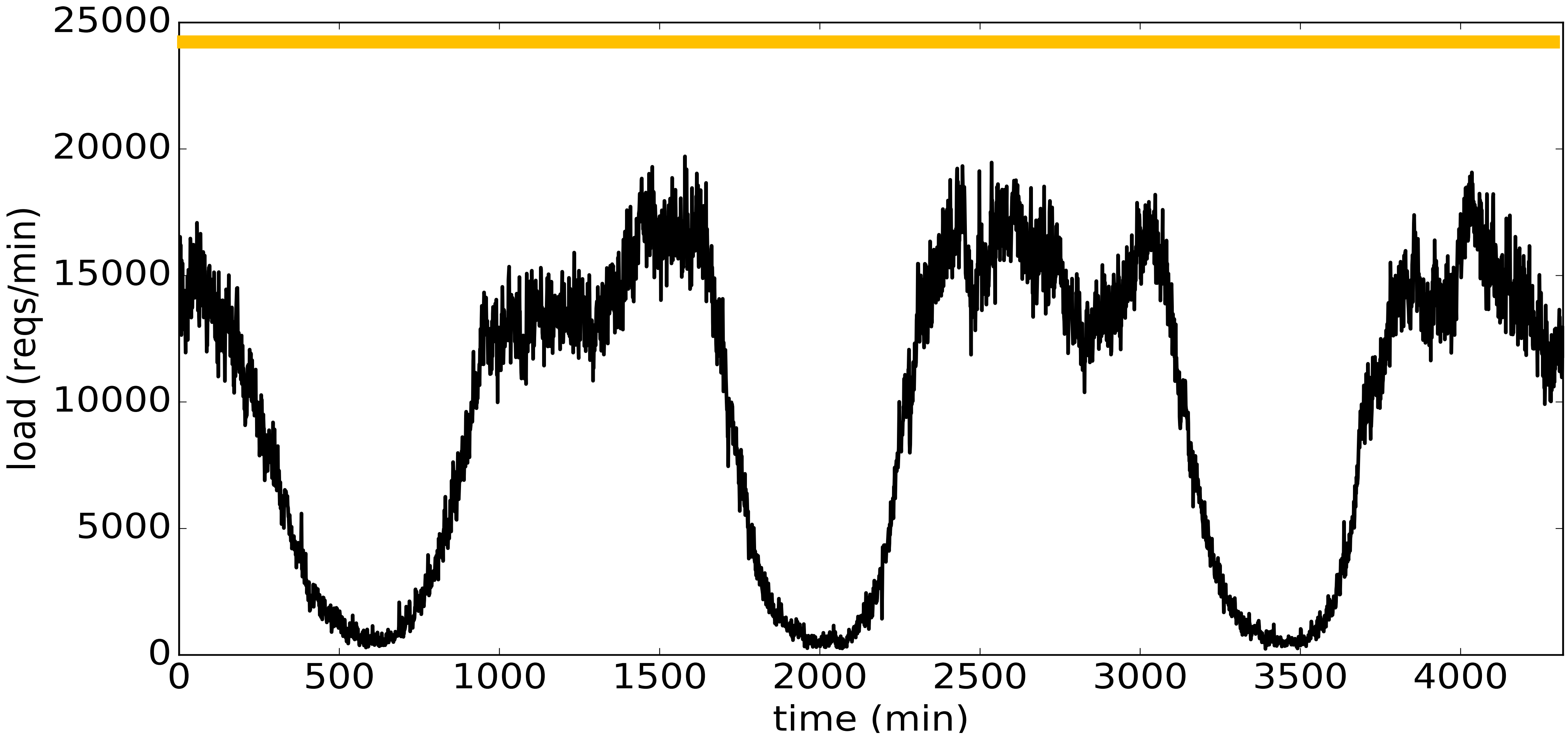


A Case Study: B2W Digital

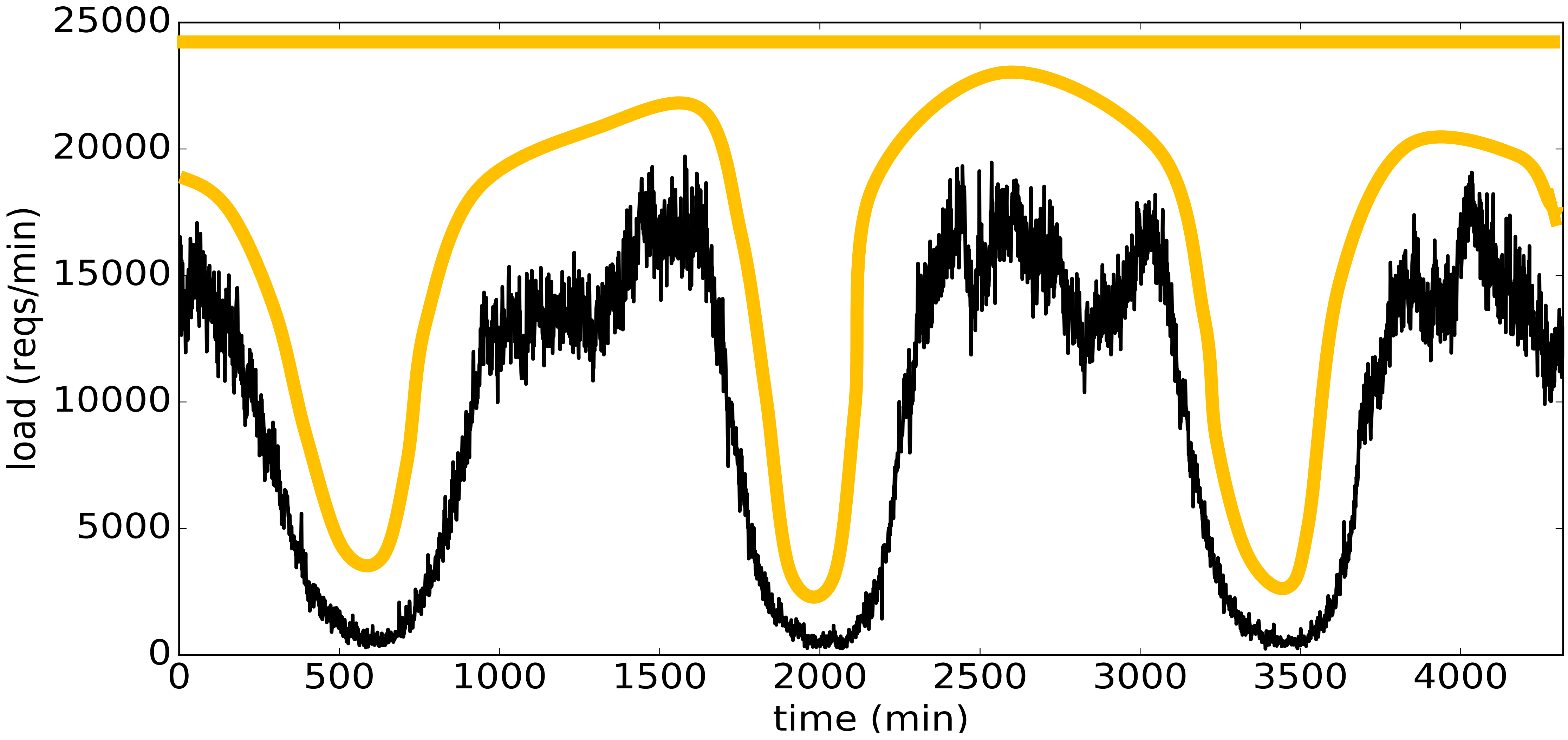
3-Day Online Retail Database Workload



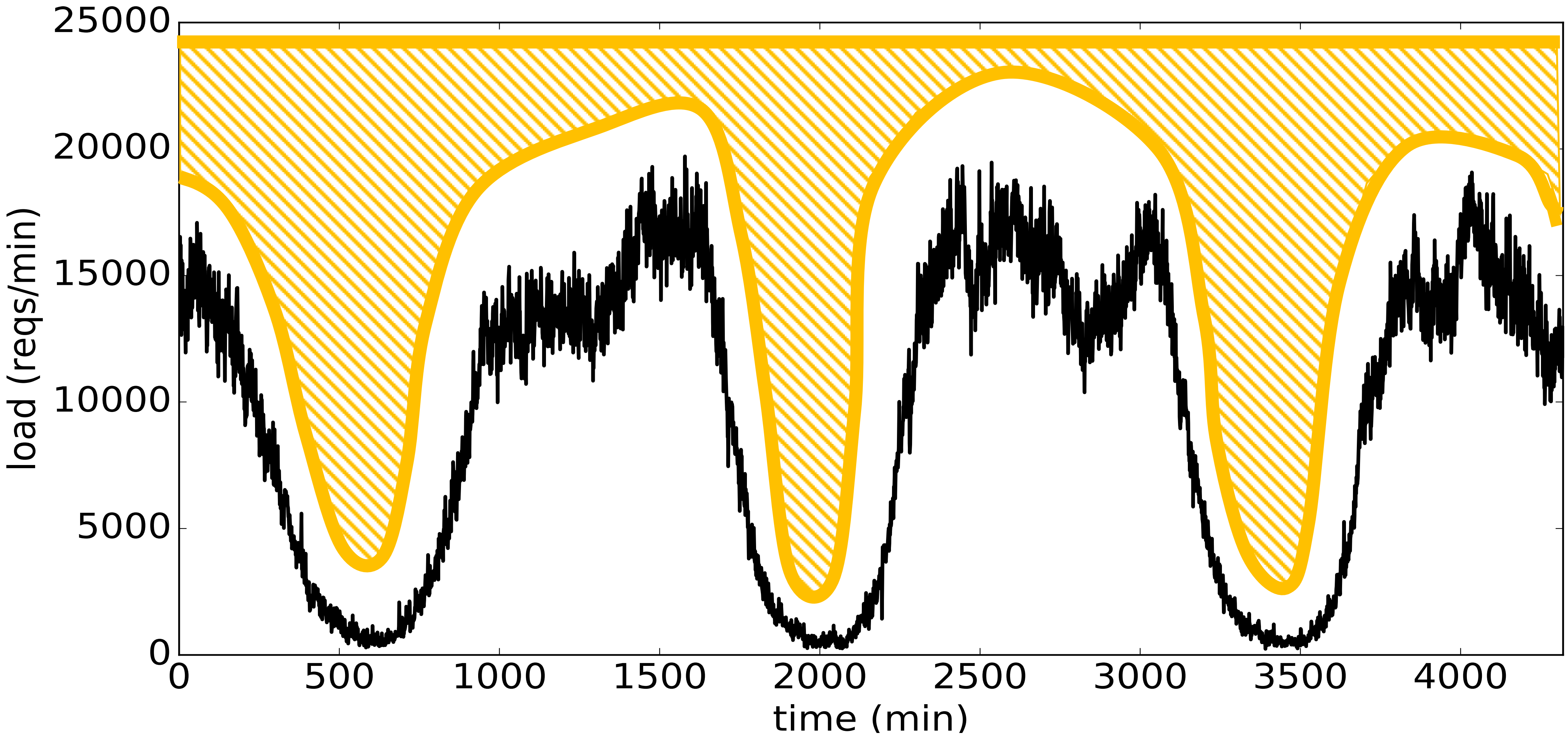
3-Day Online Retail Database Workload



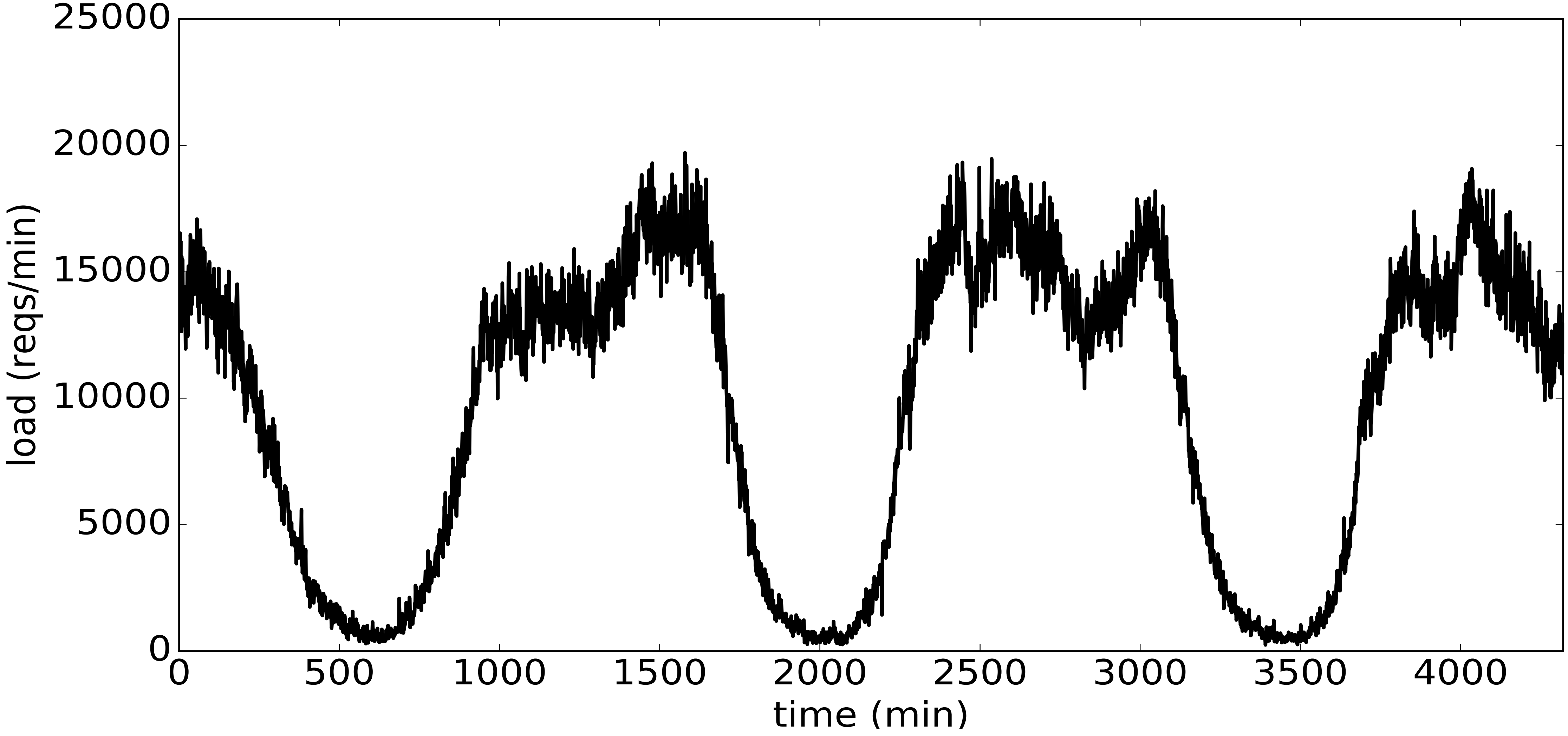
3-Day Online Retail Database Workload



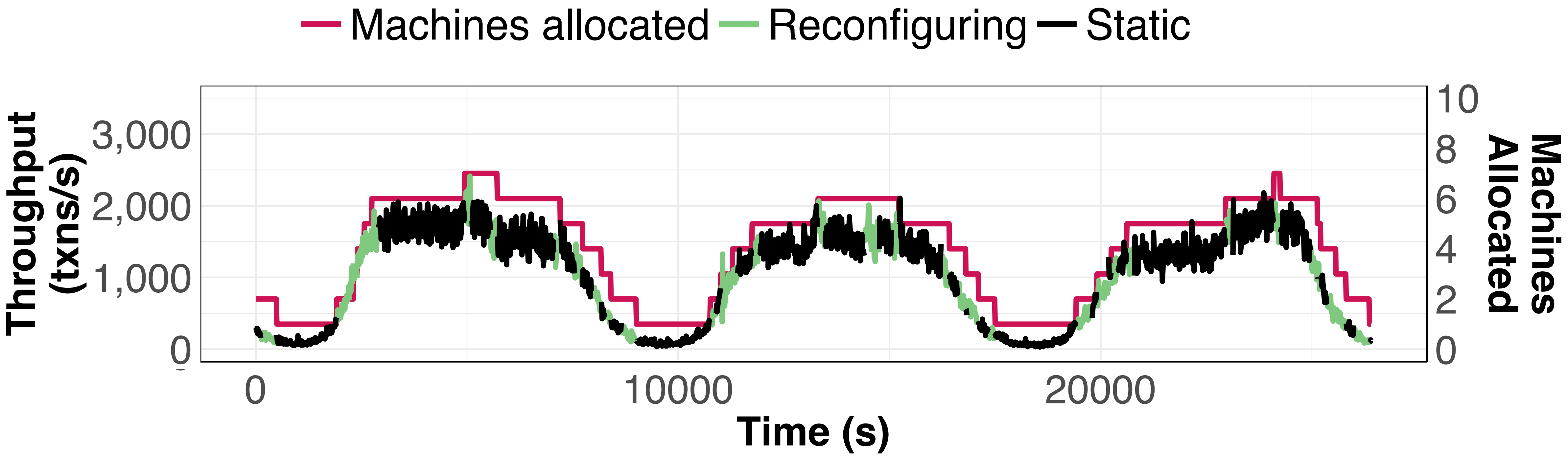
3-Day Online Retail Database Workload



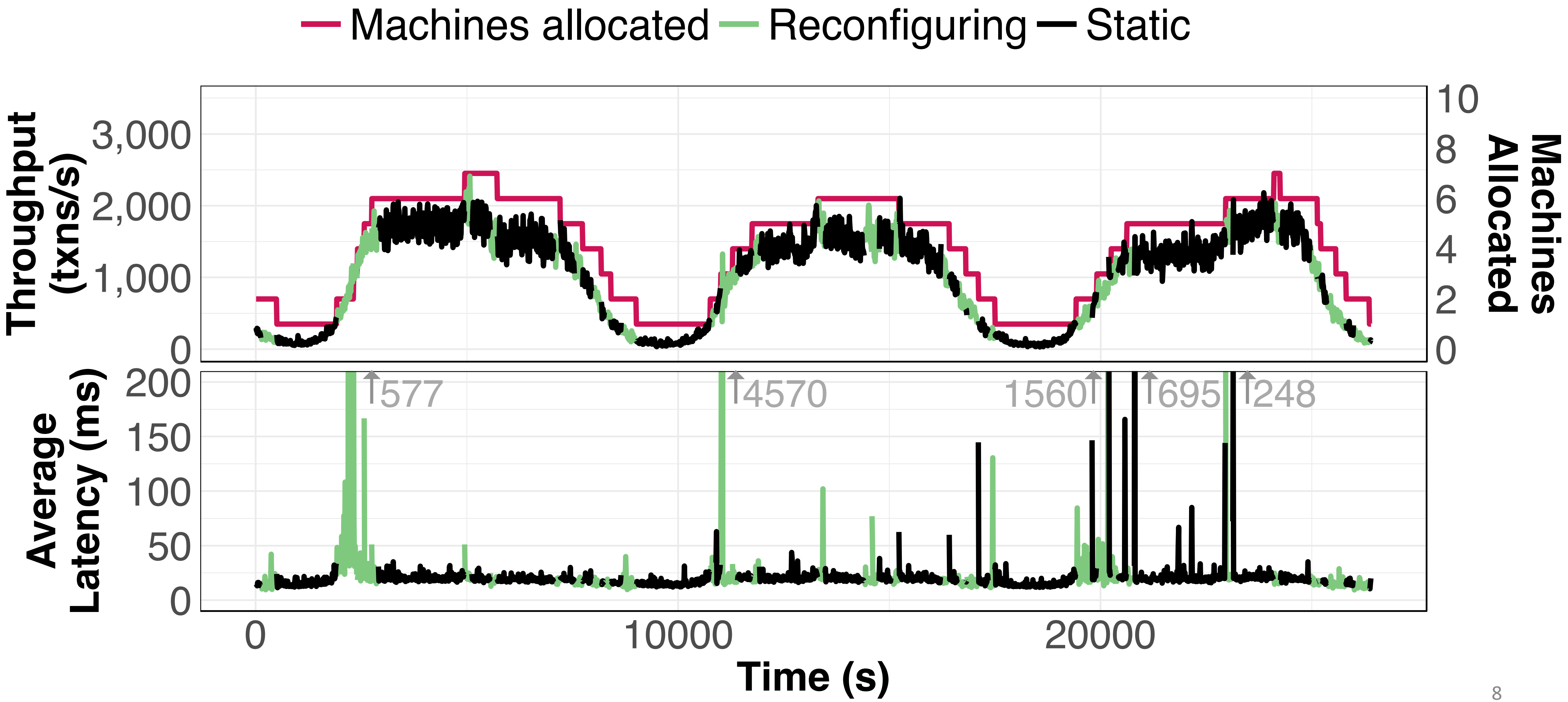
3-Day Online Retail Database Workload



Elastic Scaling Adapts to Workload

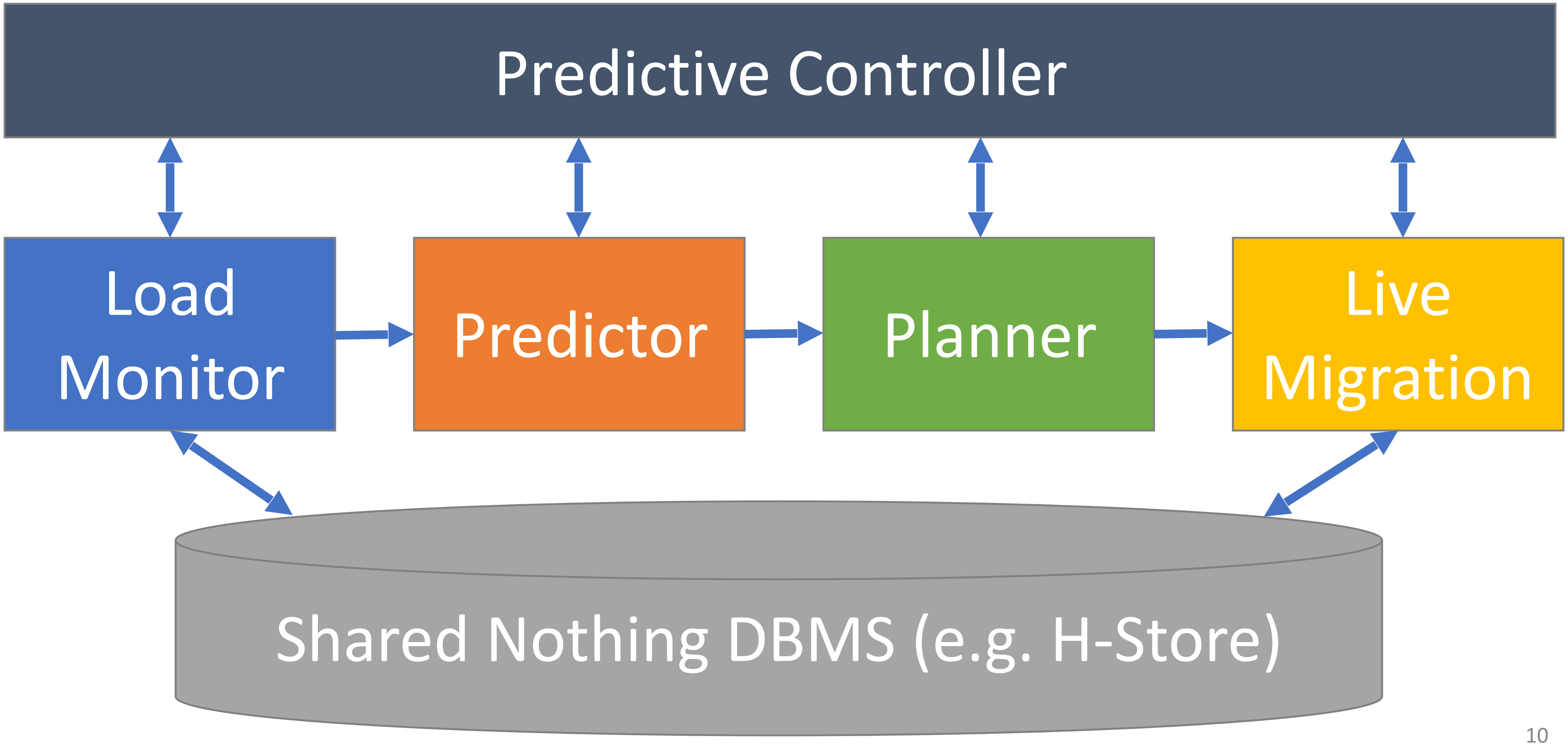


Reactive Scaling Causes Latency Spikes

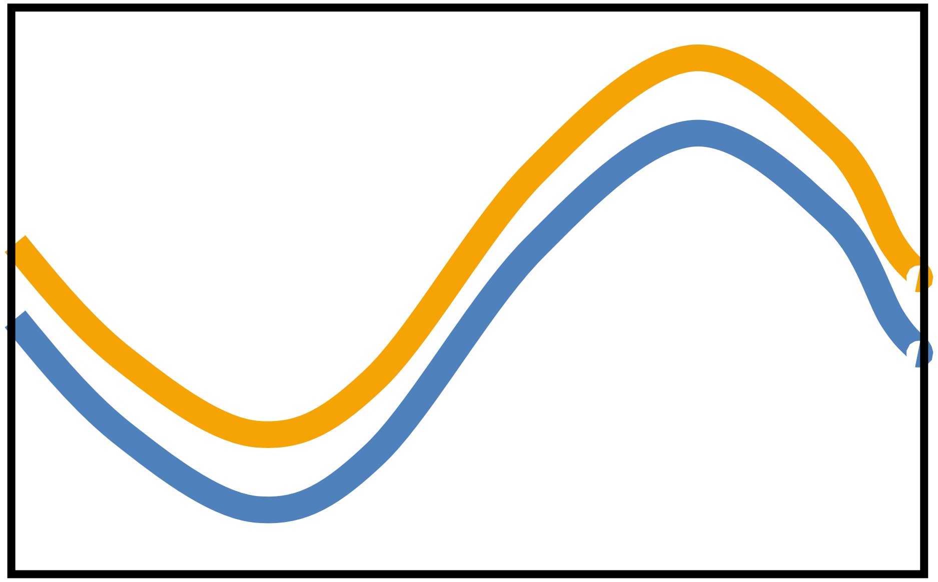


P-Store
A Predictive Elastic DBMS

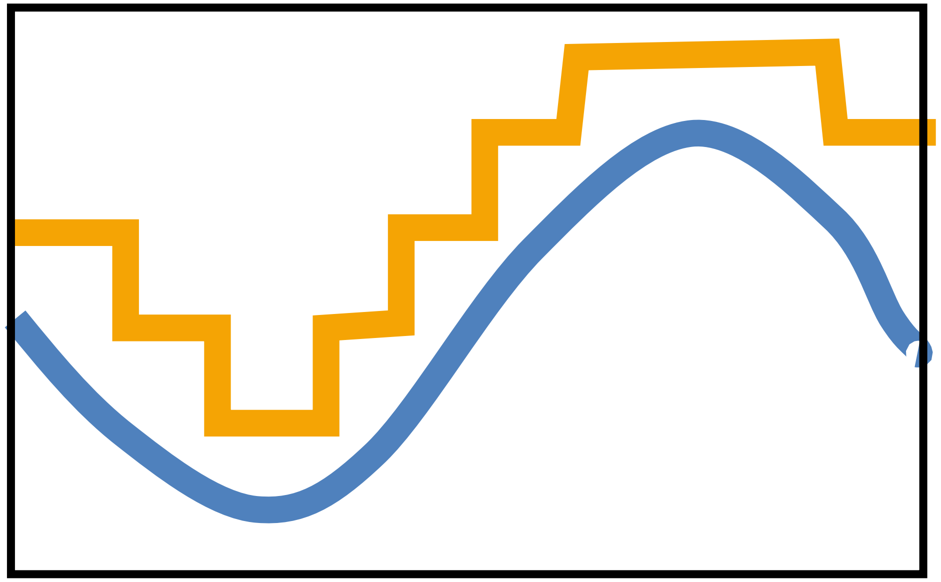
P-Store Architecture



Ideal Capacity

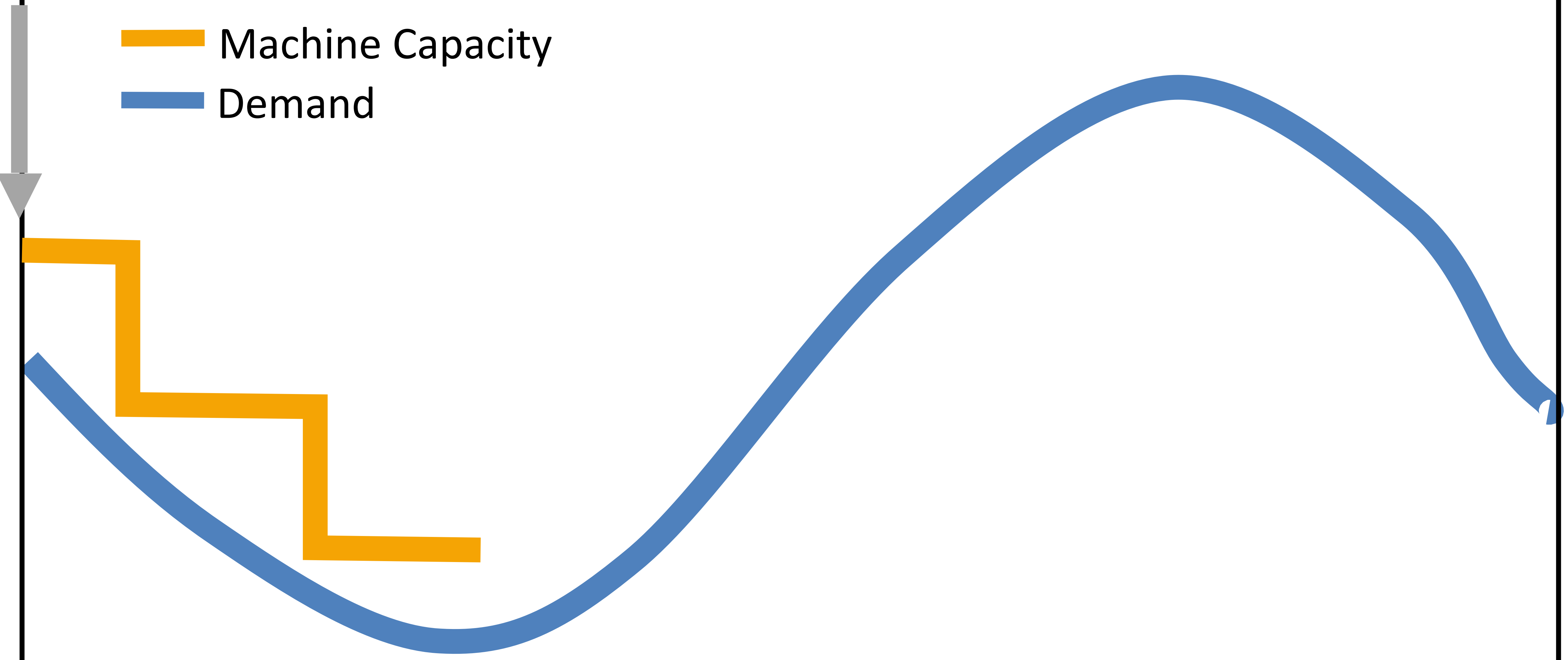


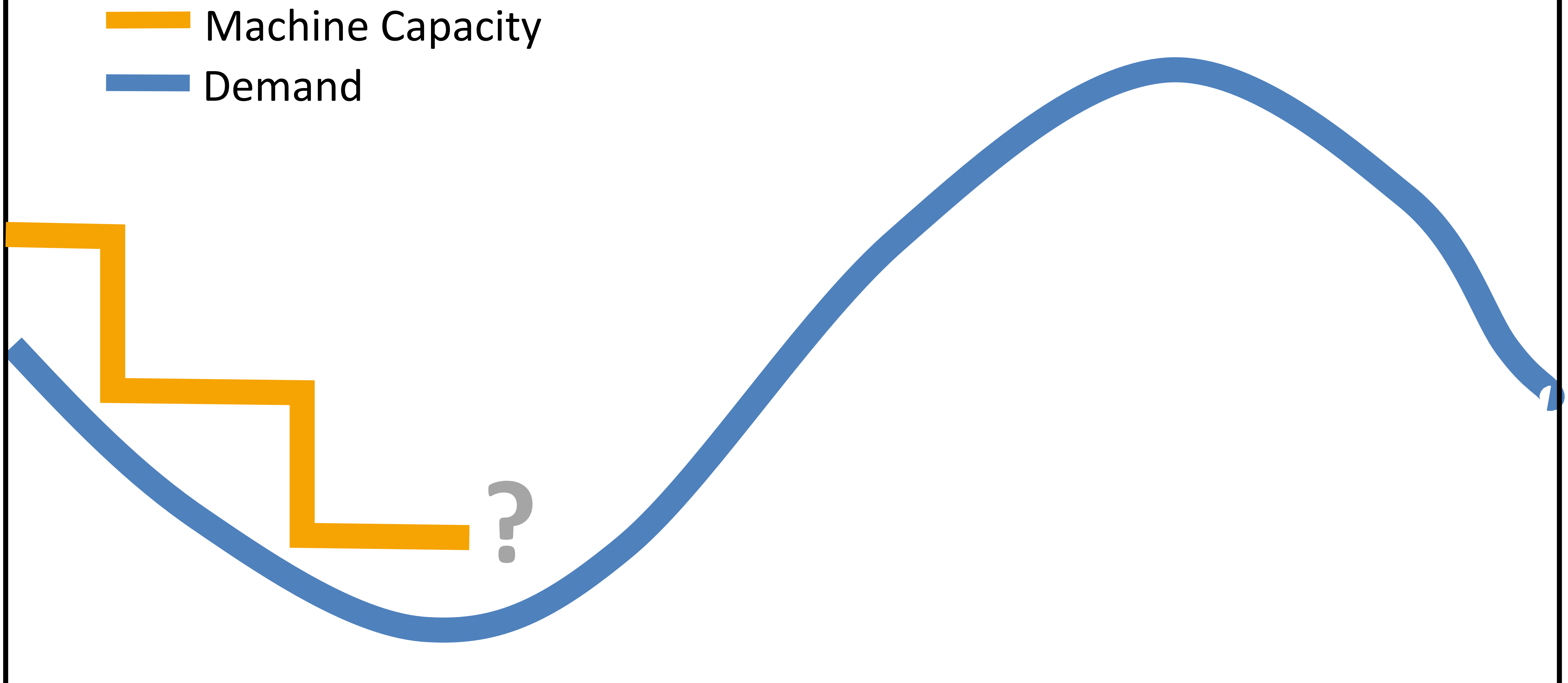
Actual Servers Allocated

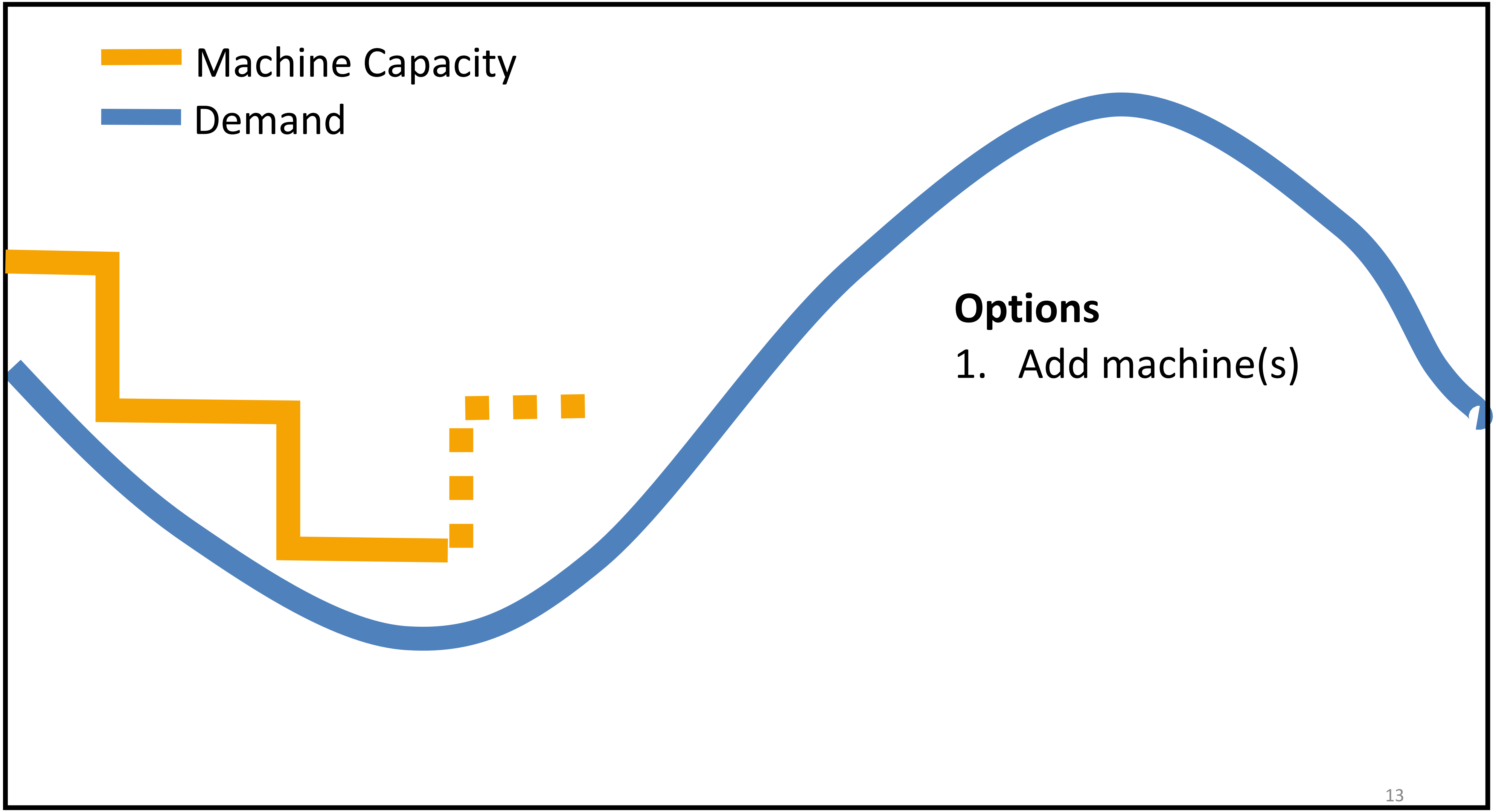


Machine Capacity

Demand

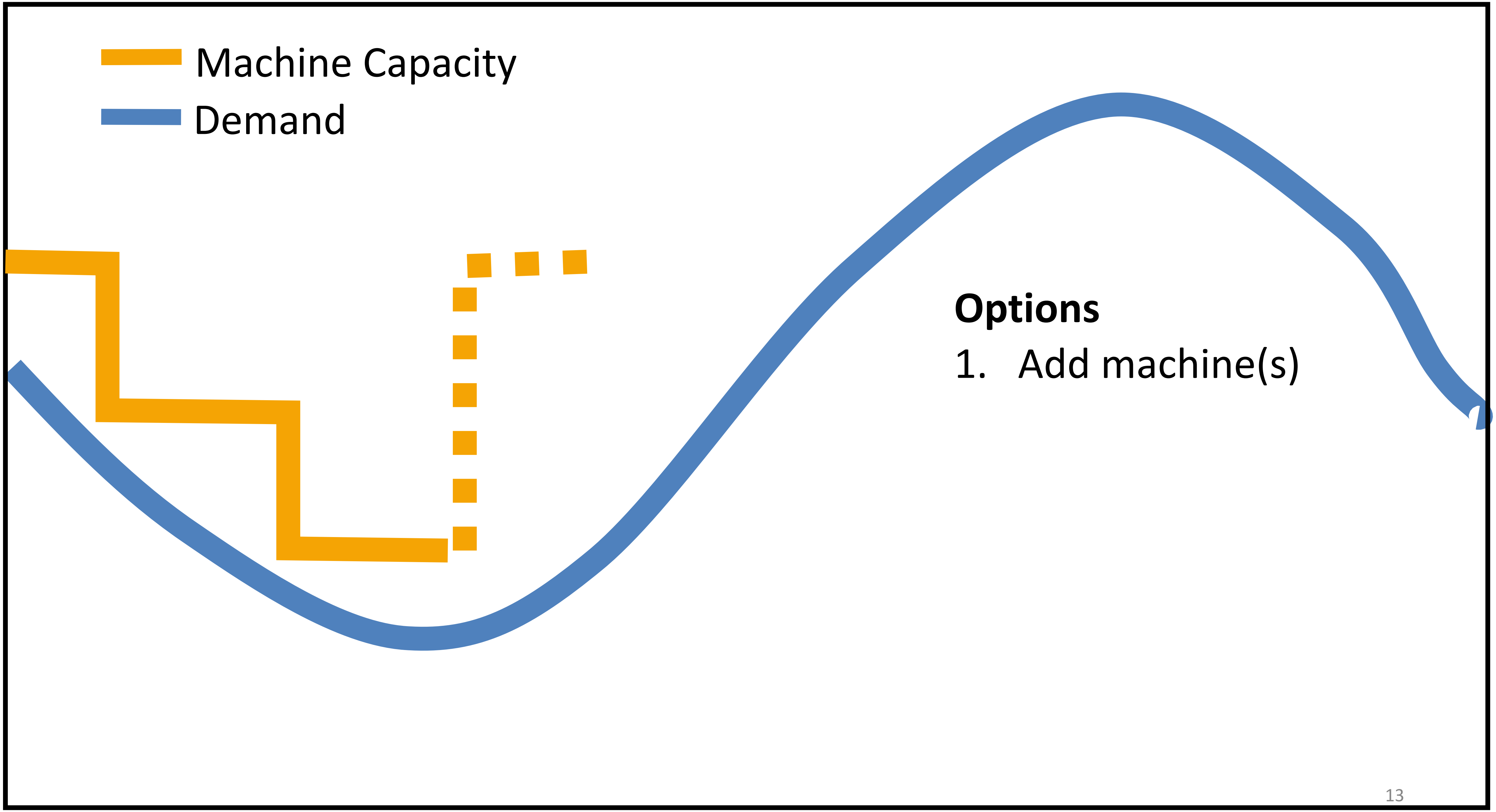

Machine Capacity
Demand

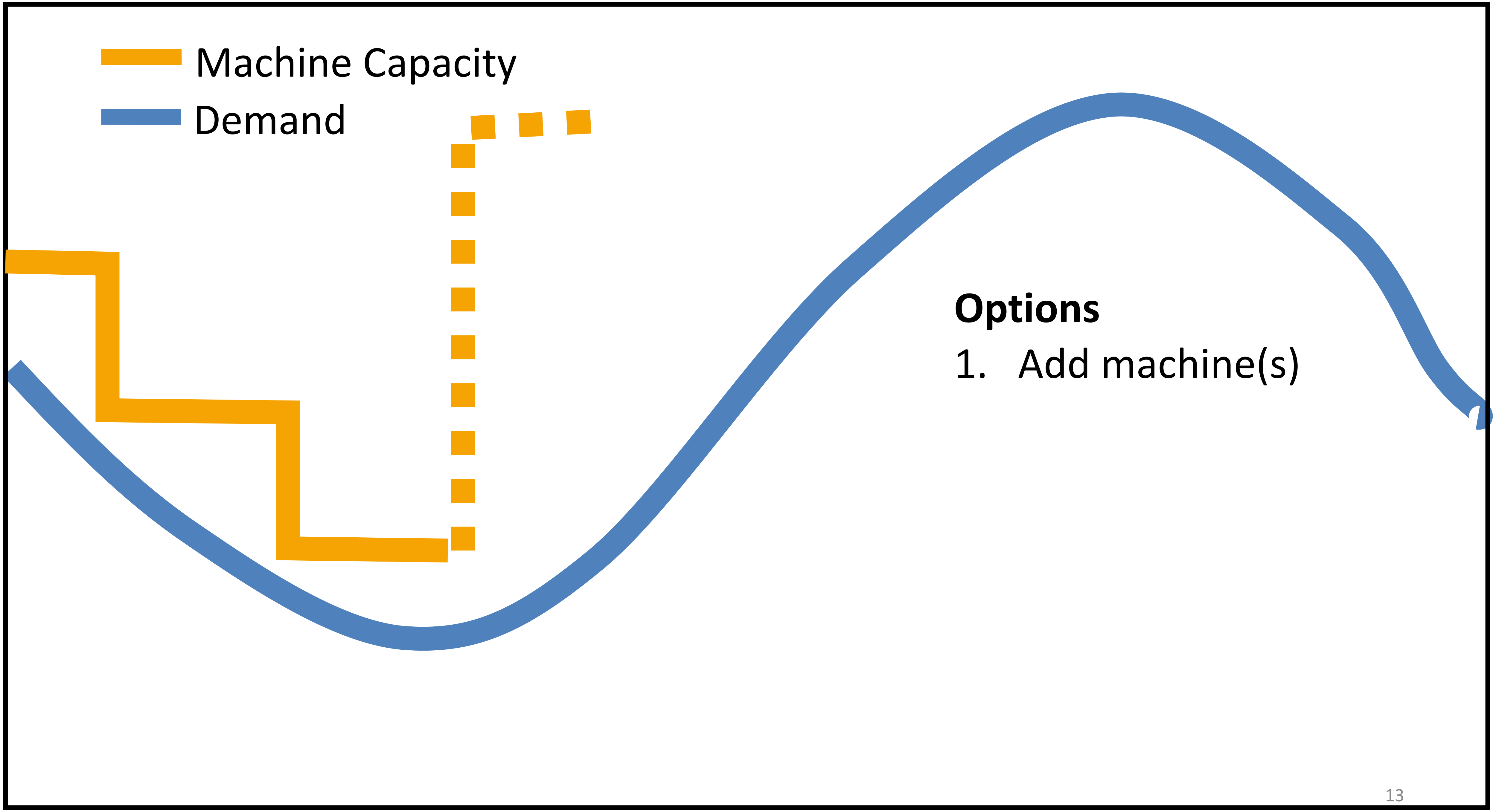

Machine Capacity
Demand
?



Options

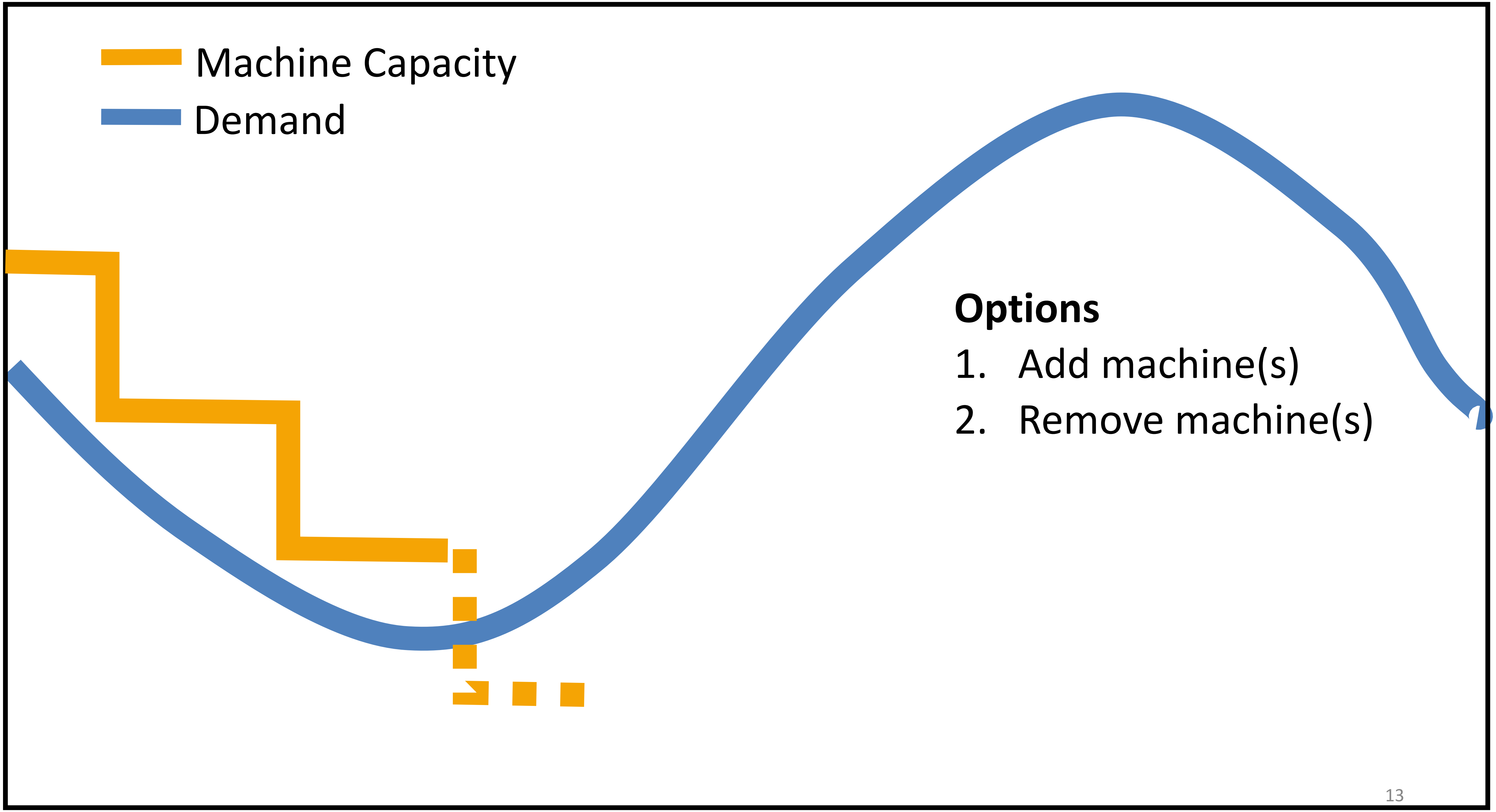
1. Add machine(s)

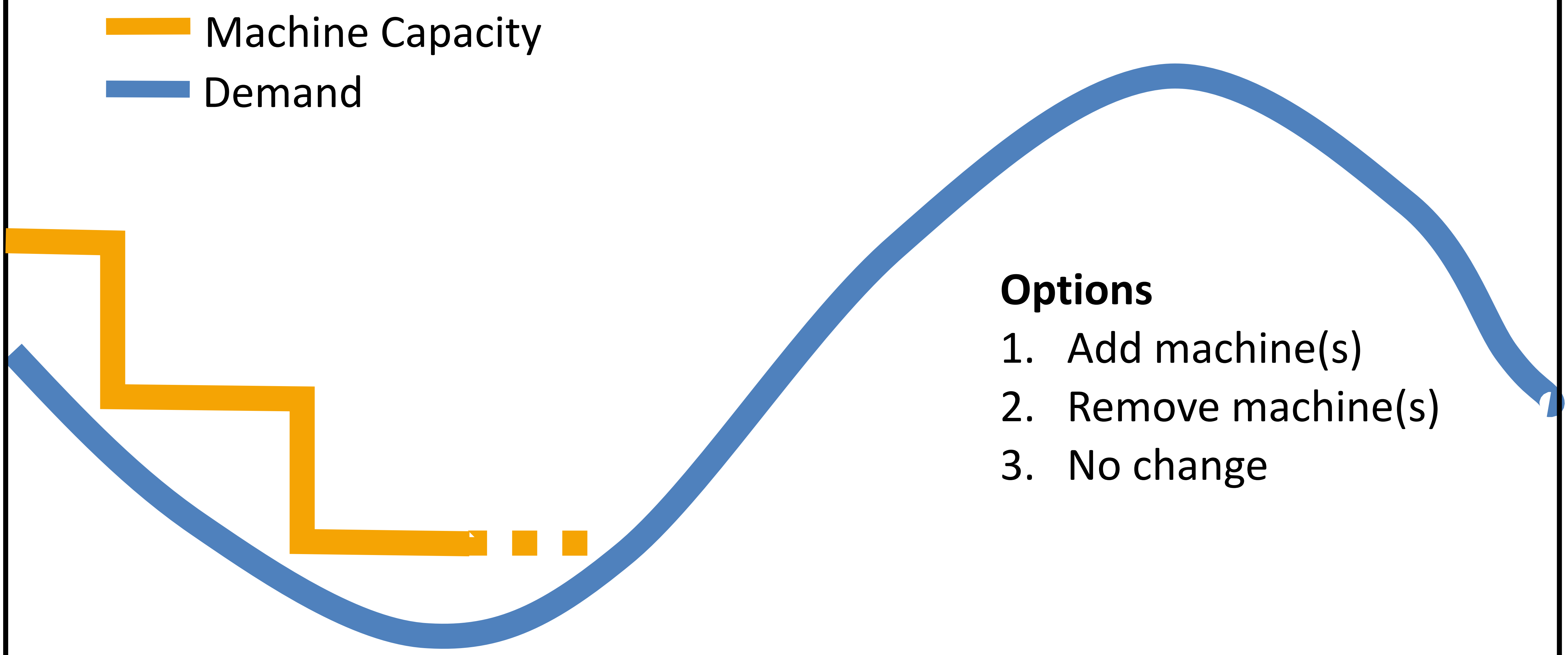

Machine Capacity
Demand
Options
1. Add machine(s)



Options

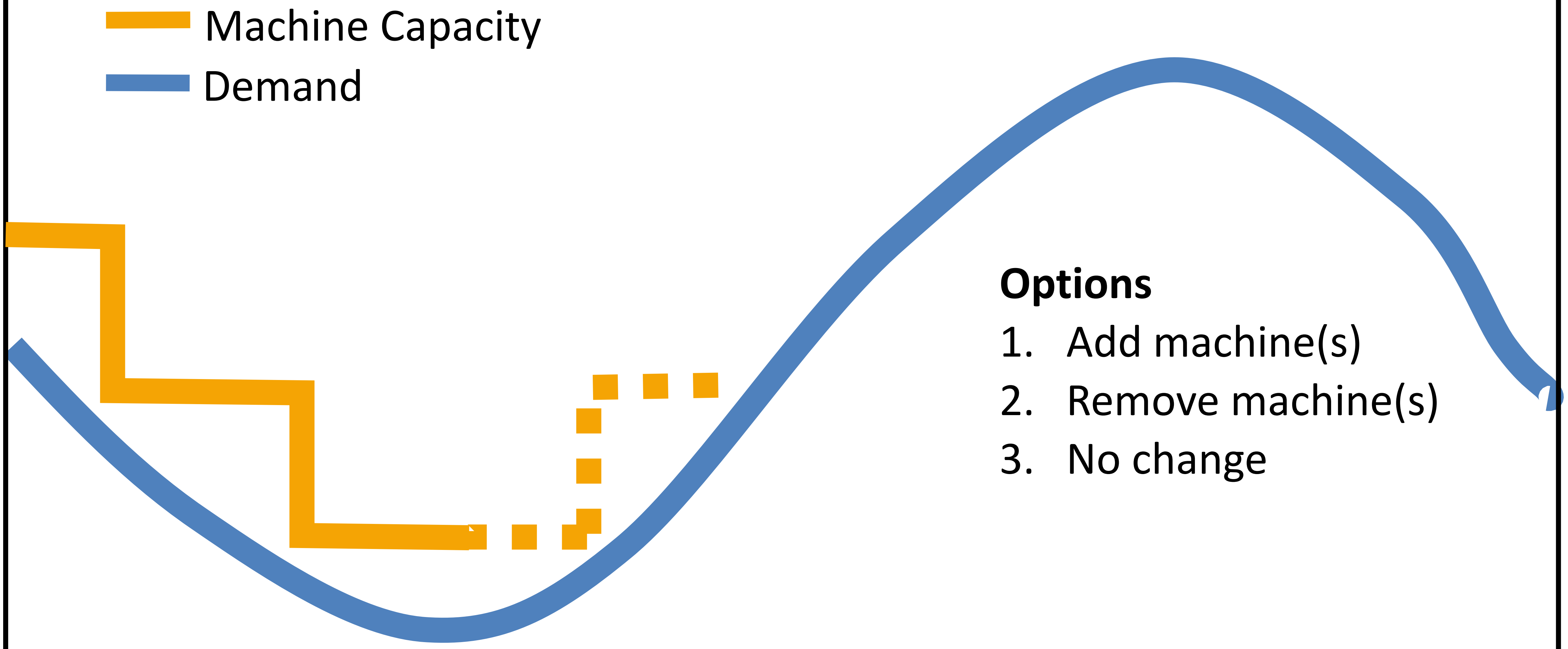
1. Add machine(s)


Machine Capacity
Demand
Options
1. Add machine(s)
2. Remove machine(s)



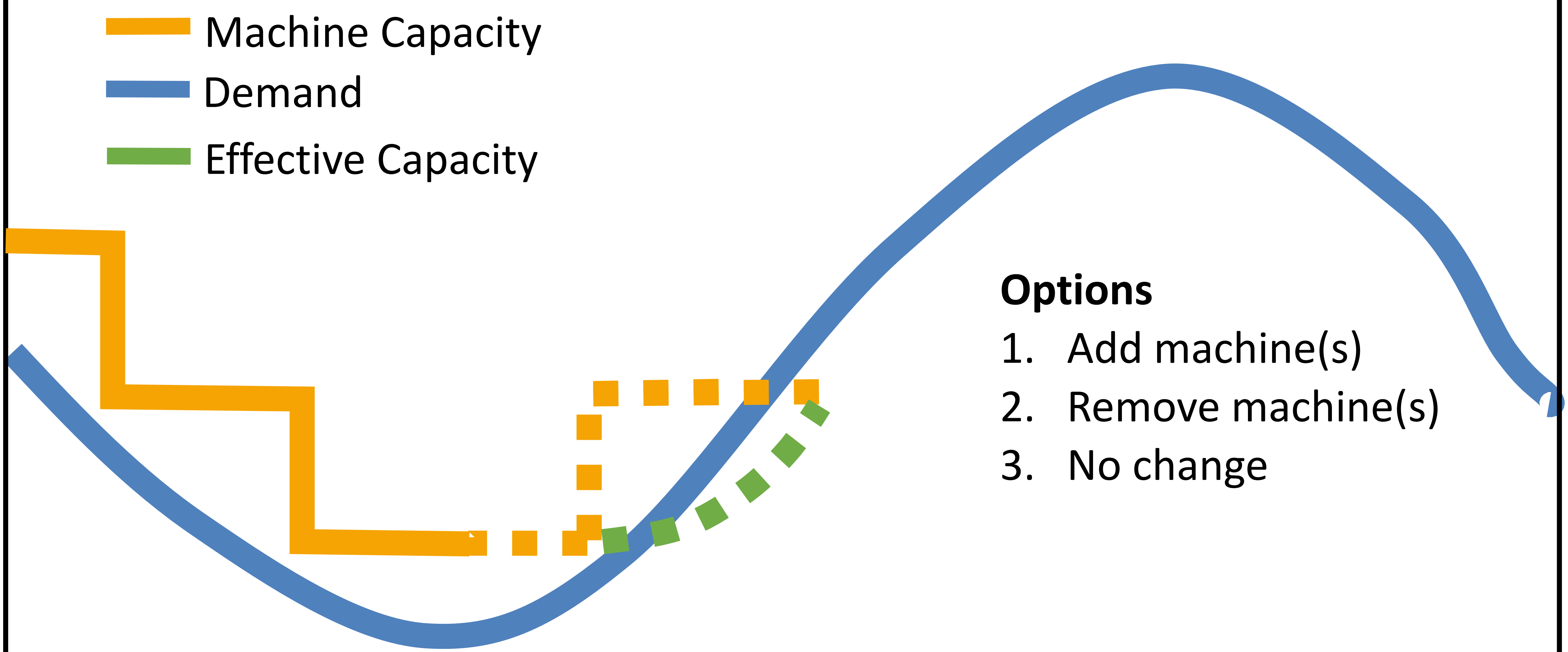
Options

1. Add machine(s)
2. Remove machine(s)
3. No change



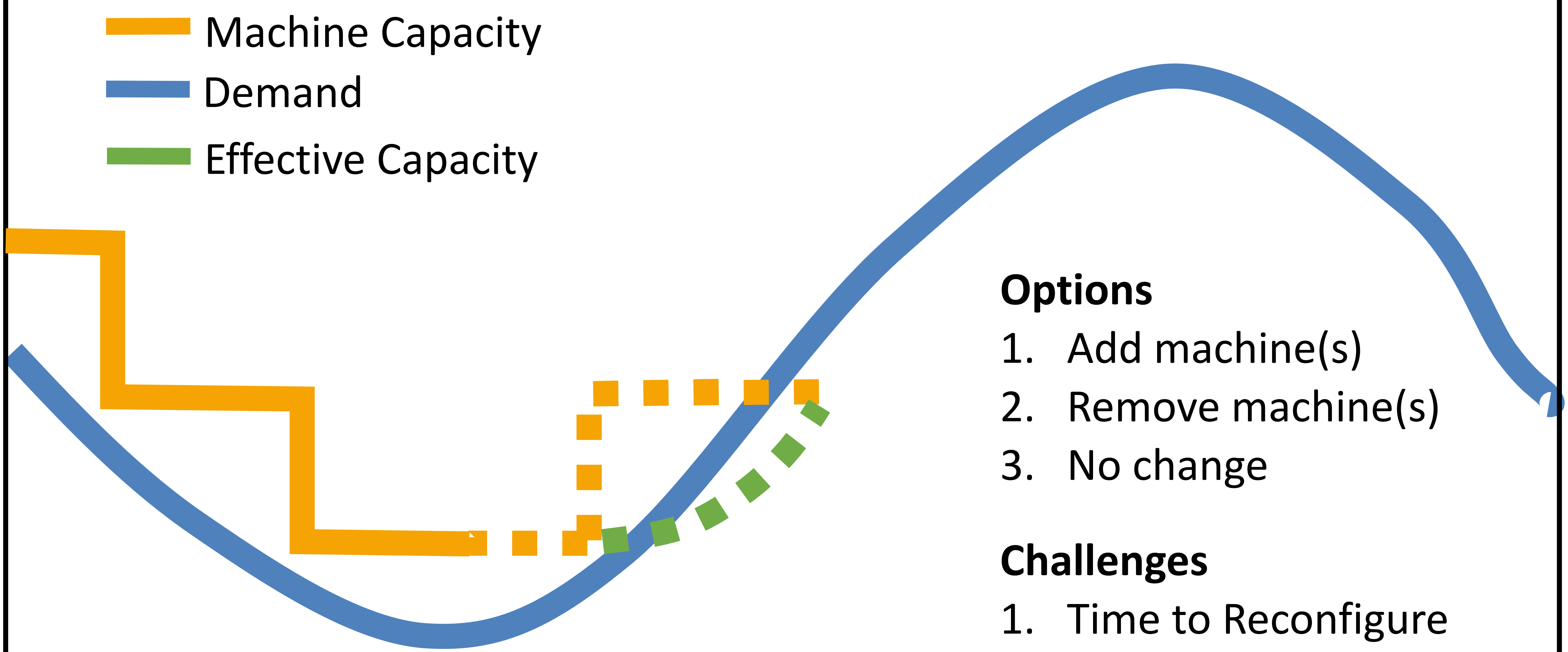
Options

1. Add machine(s)
2. Remove machine(s)
3. No change



Options

1. Add machine(s)
2. Remove machine(s)
3. No change



Options

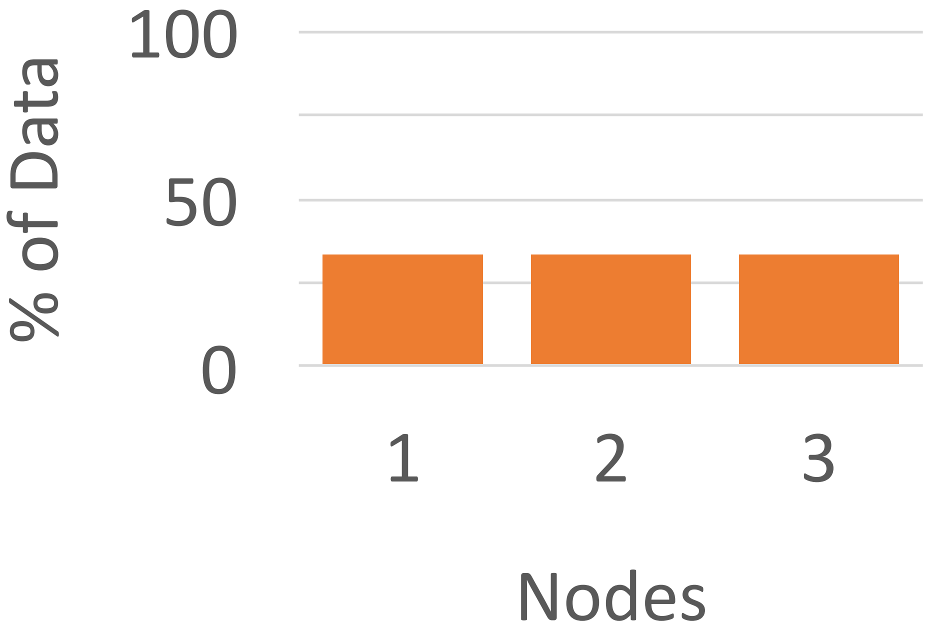
1. Add machine(s)
2. Remove machine(s)
3. No change

Challenges

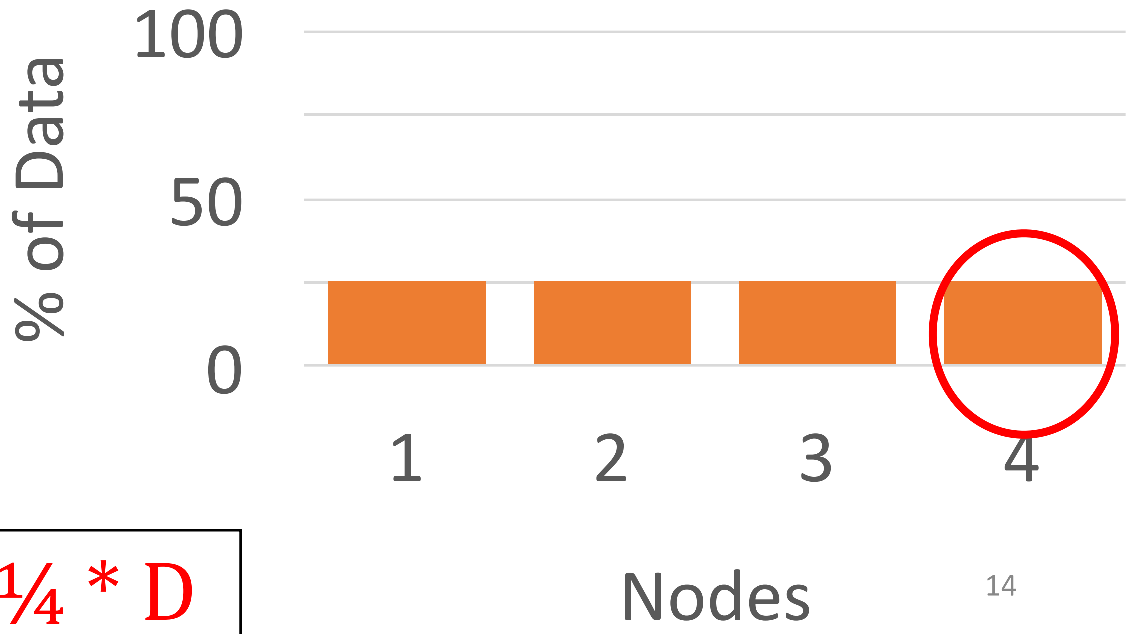
1. Time to Reconfigure
2. Effective Capacity
3. Cost

Time to Reconfigure

- Rate control: small chunks of data, spaced apart
- ∃ some time *D* – minimum time to move entire database w/ no performance impact



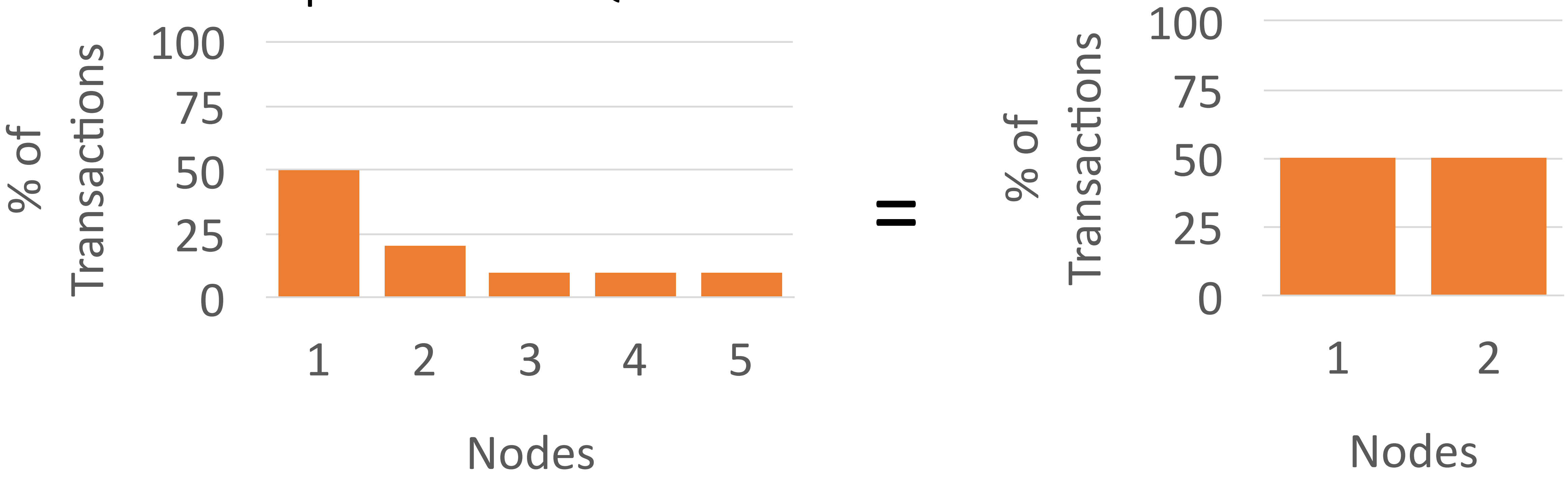
→



$\boxed{\text{Time} = \frac{1}{4} * D}$

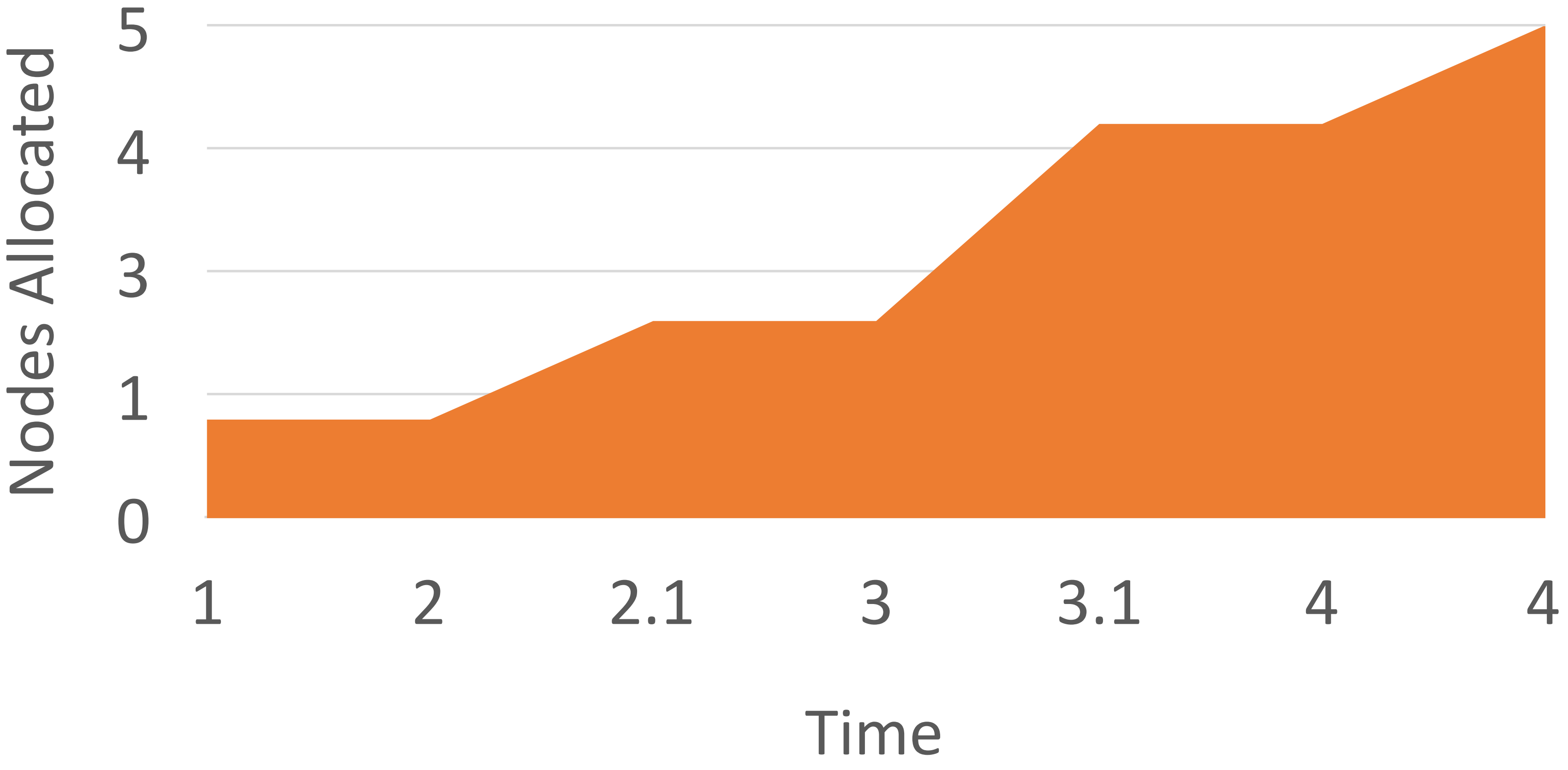
Effective Capacity

- What transaction rate can the system support?
- Max rate per machine: Q

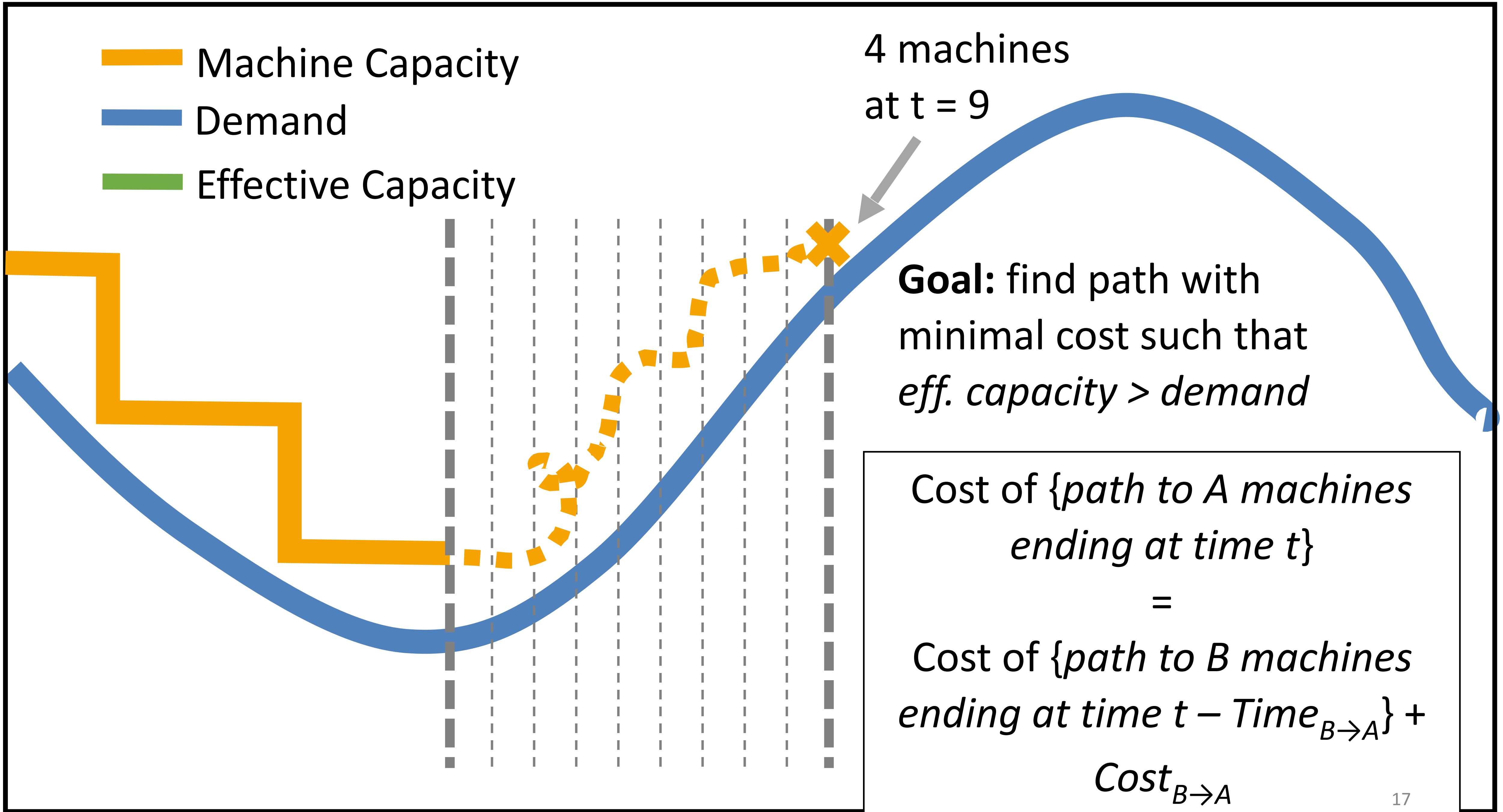


$$\textbf{Effective Capacity} = \frac{Q}{0.5} = 2 * Q$$

Cost



$\boxed{\text{Cost} = 7}$



Goal: find path with minimal cost such that *eff. capacity > demand*

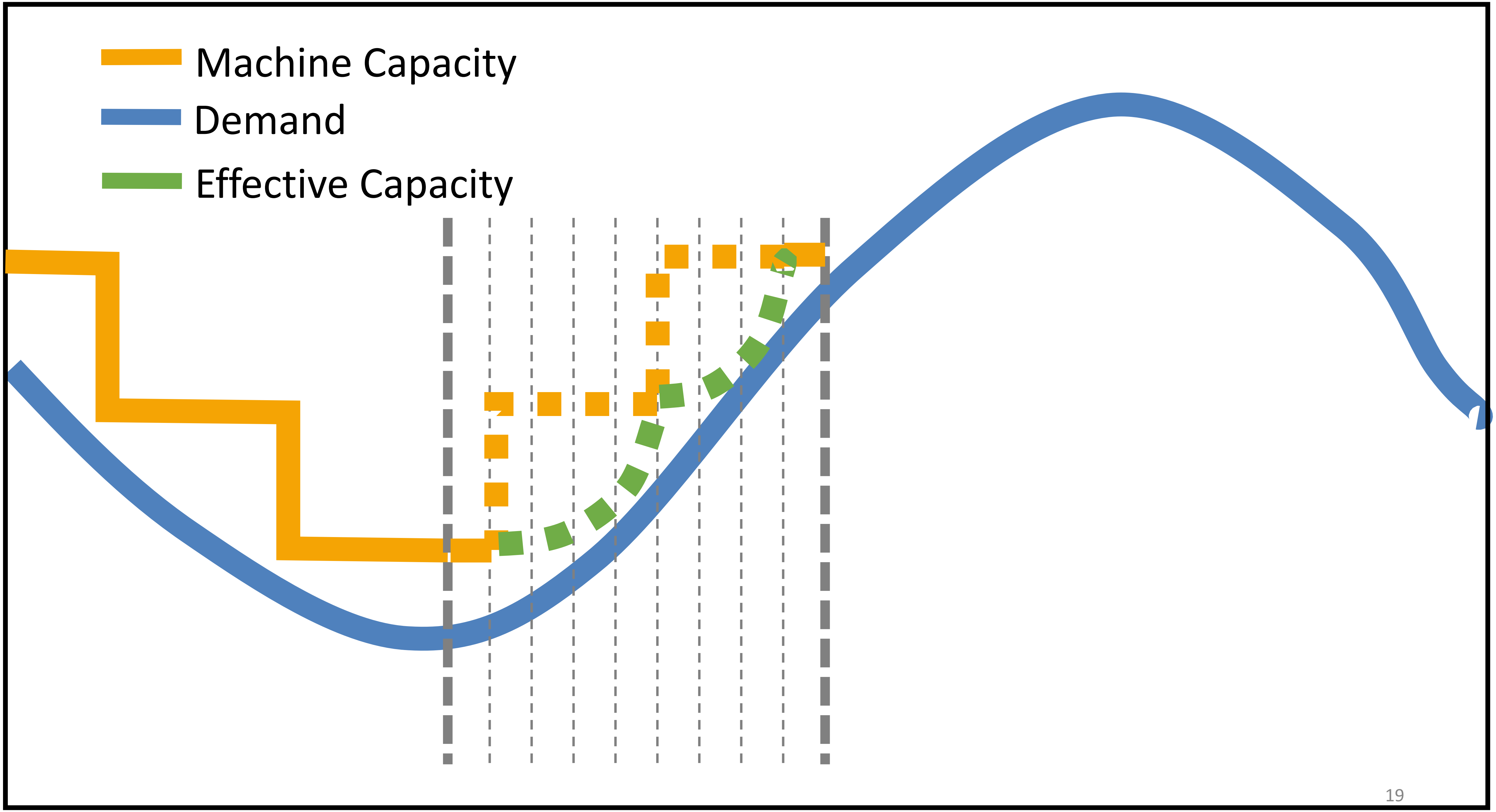
Cost of {*path to A machines ending at time t*}

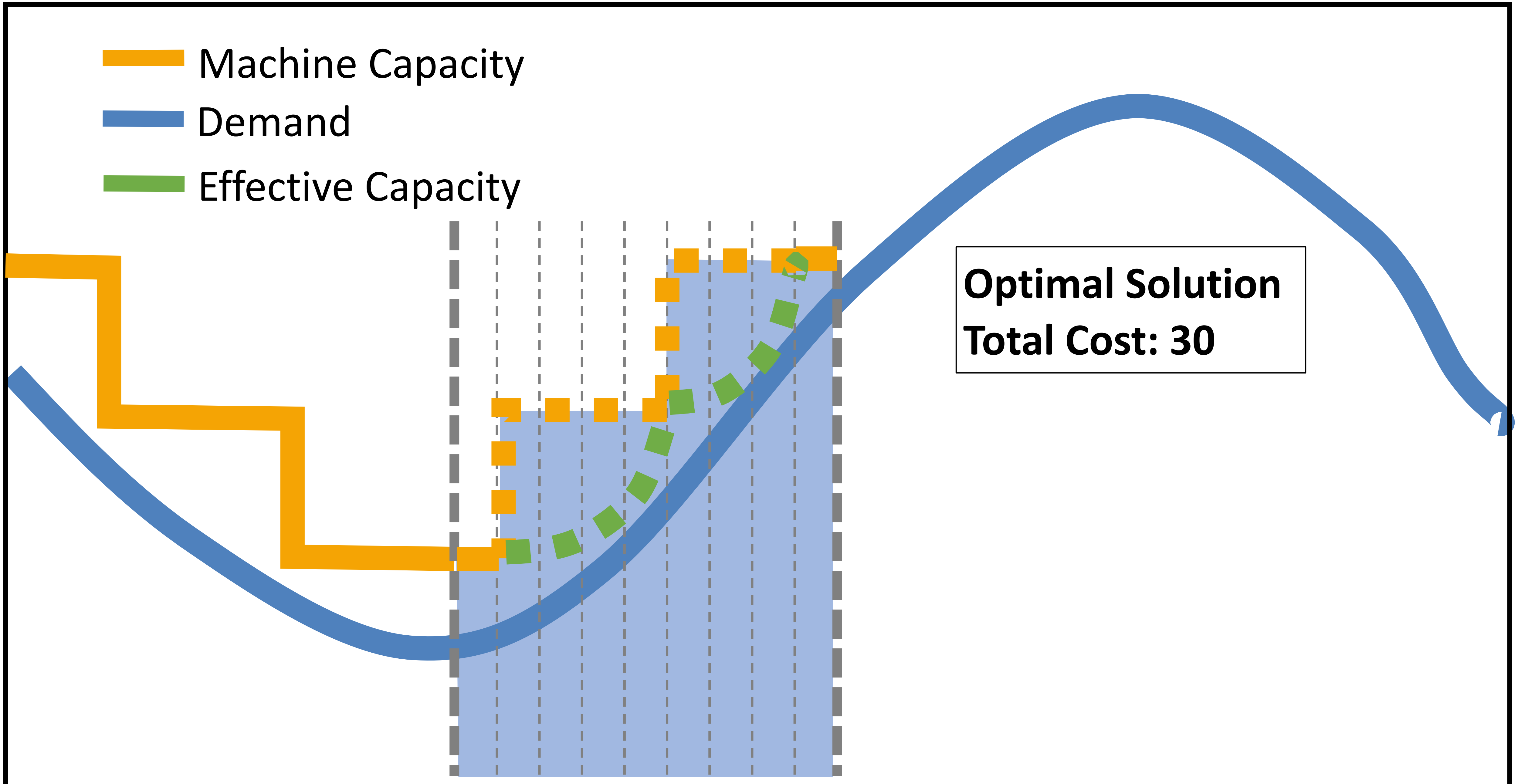
=

Cost of {*path to B machines ending at time* $t - Time_{B \to A}$} + $Cost_{B \to A}$

Dynamic Programming Algorithm for Planning Reconfigurations

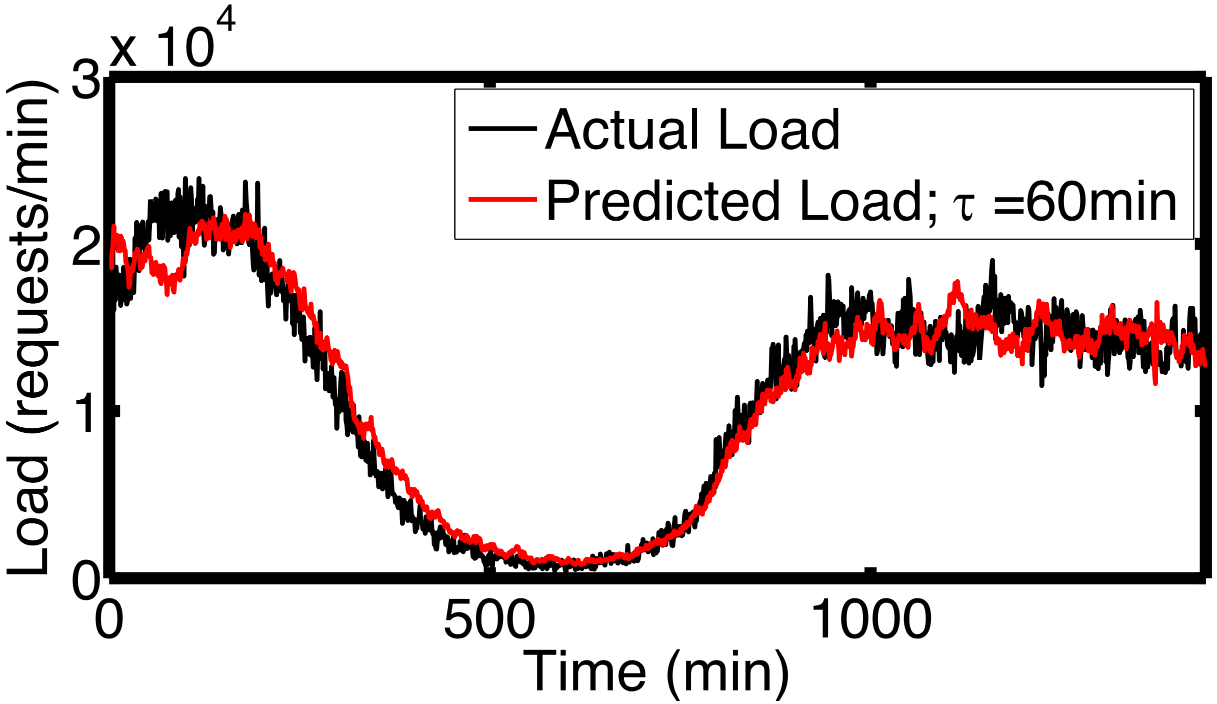
- Complexity $\propto$ # time steps and max number of machines needed
- In practice: runtime < 1 second
- Greedy alternatives don't work


Machine Capacity
Demand
Effective Capacity


Machine Capacity
Demand
Effective Capacity
Optimal Solution
Total Cost: 30

Time Series Prediction

- Diurnal, periodic workload, some variation day to day
- Prediction must complete in seconds-to-minutes
- We use Sparse Periodic Auto-Regression (SPAR) – Chen et al. NSDI ‘08



Evaluation

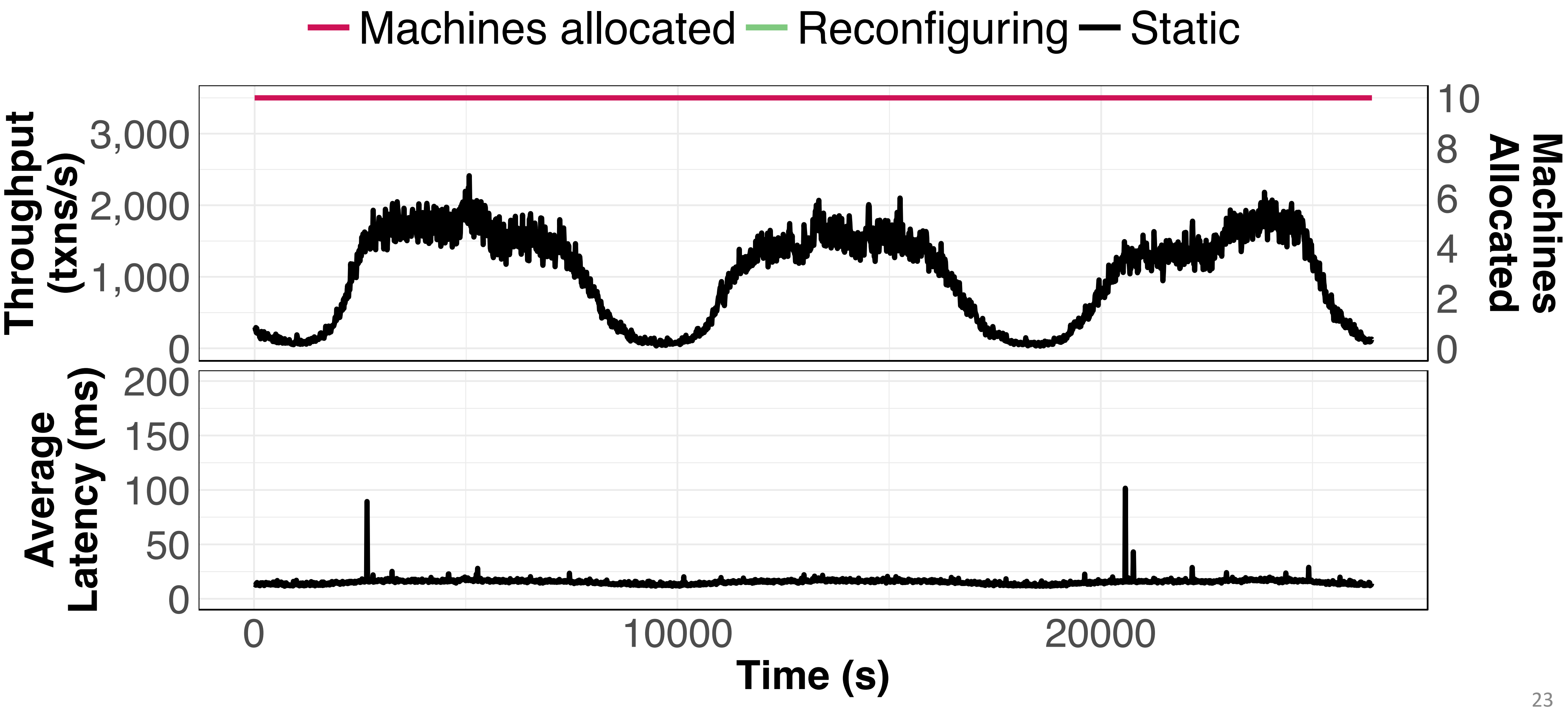
- Can we reduce resource usage?
- Can we prevent latency spikes?

Experiments

- 3 Days of B2W workload run on H-Store
- Cluster of 10 servers, 6 partitions per server
- About 1 GB of shopping cart and checkout data
- Track machines allocated, throughput, latency, reconfiguration time periods

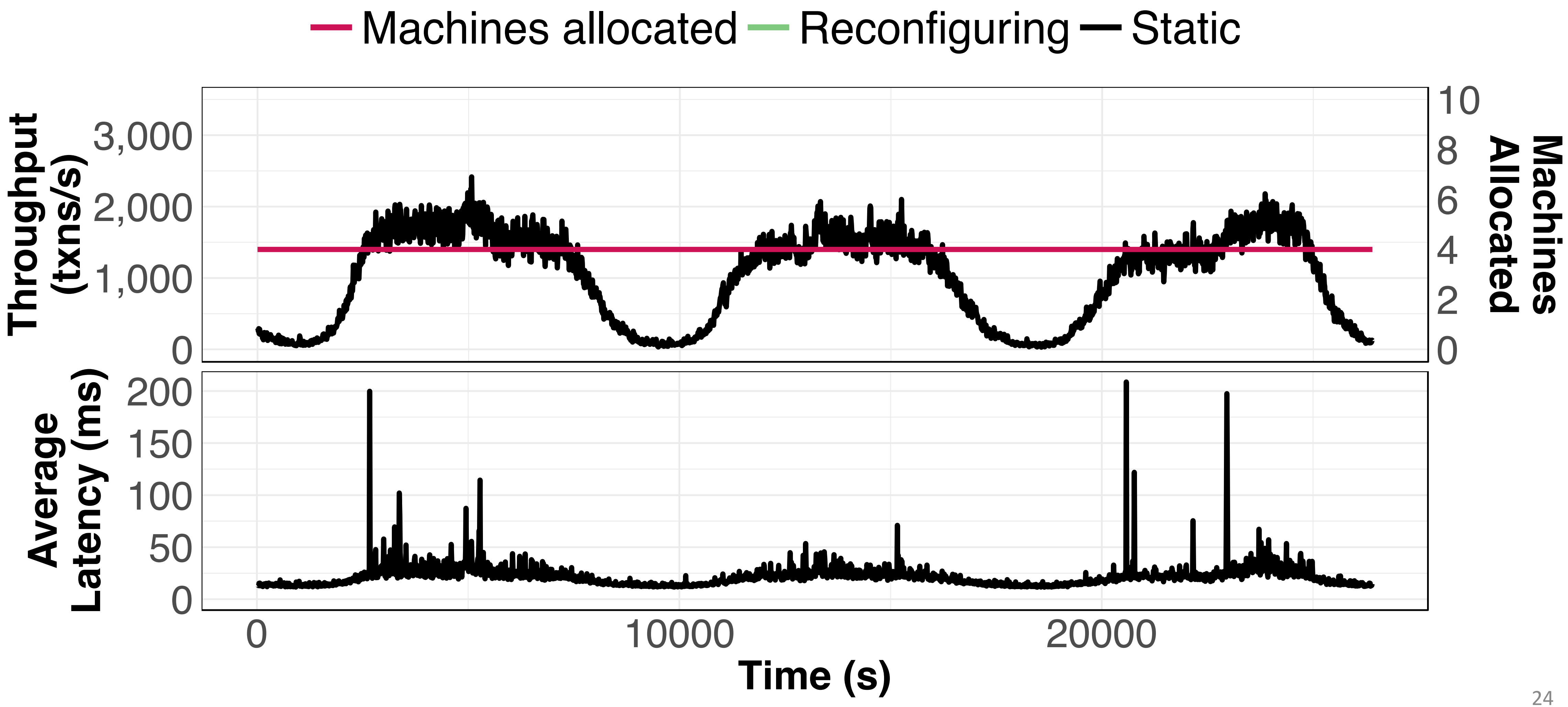
Results – Static, Peak Provisioning

Machines Used: 10



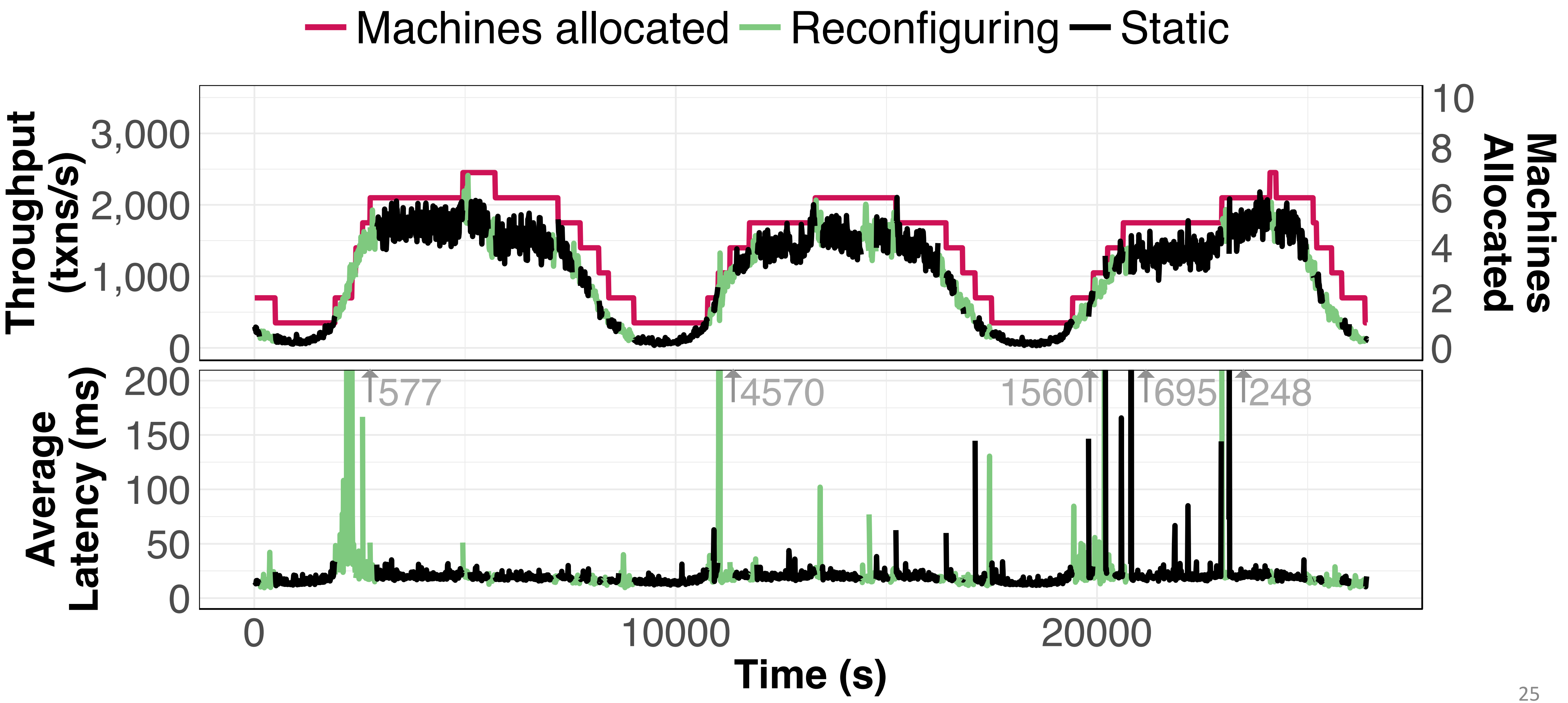
Results – Static, Average Provisioning

Machines Used: 4



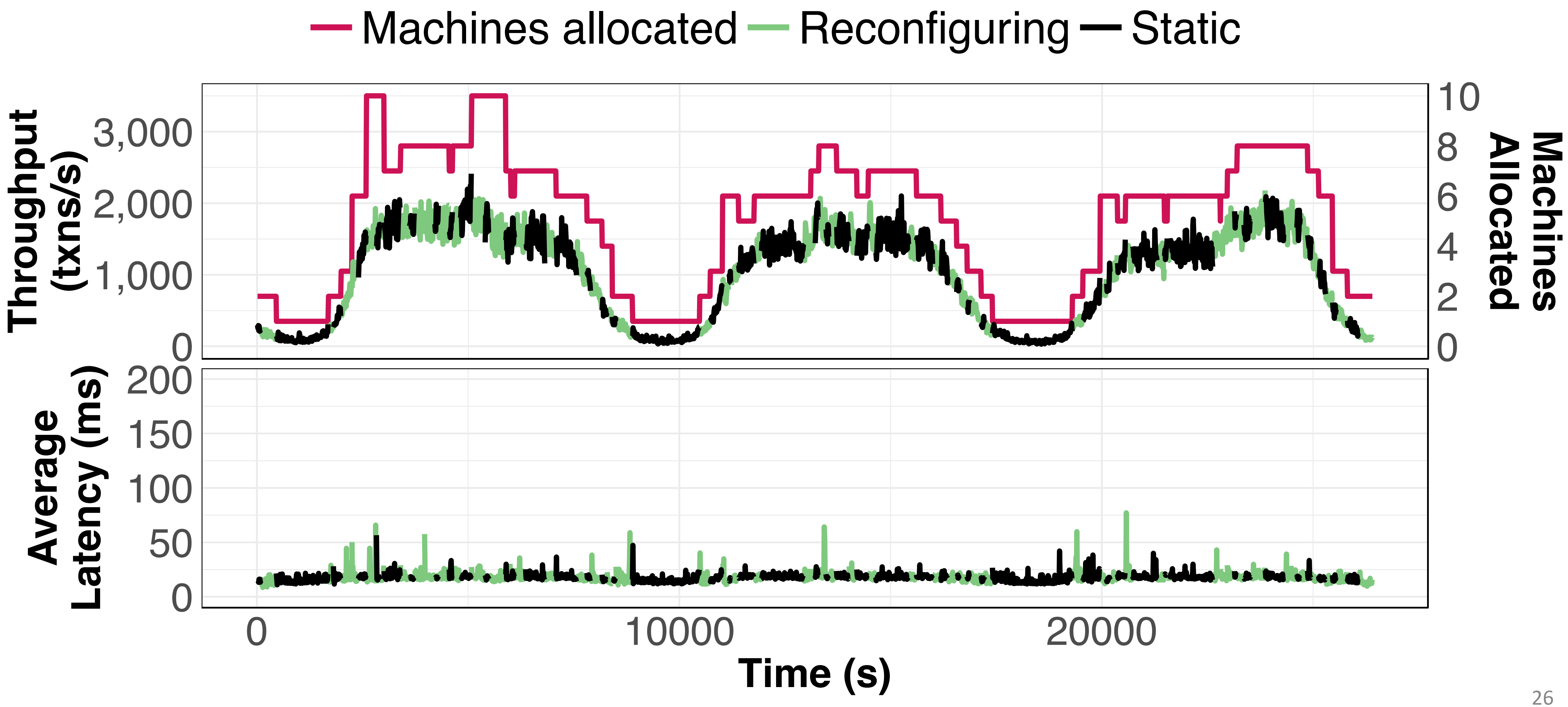
Results – Reactive Scaling

Avg. Machines Used: **4.02**



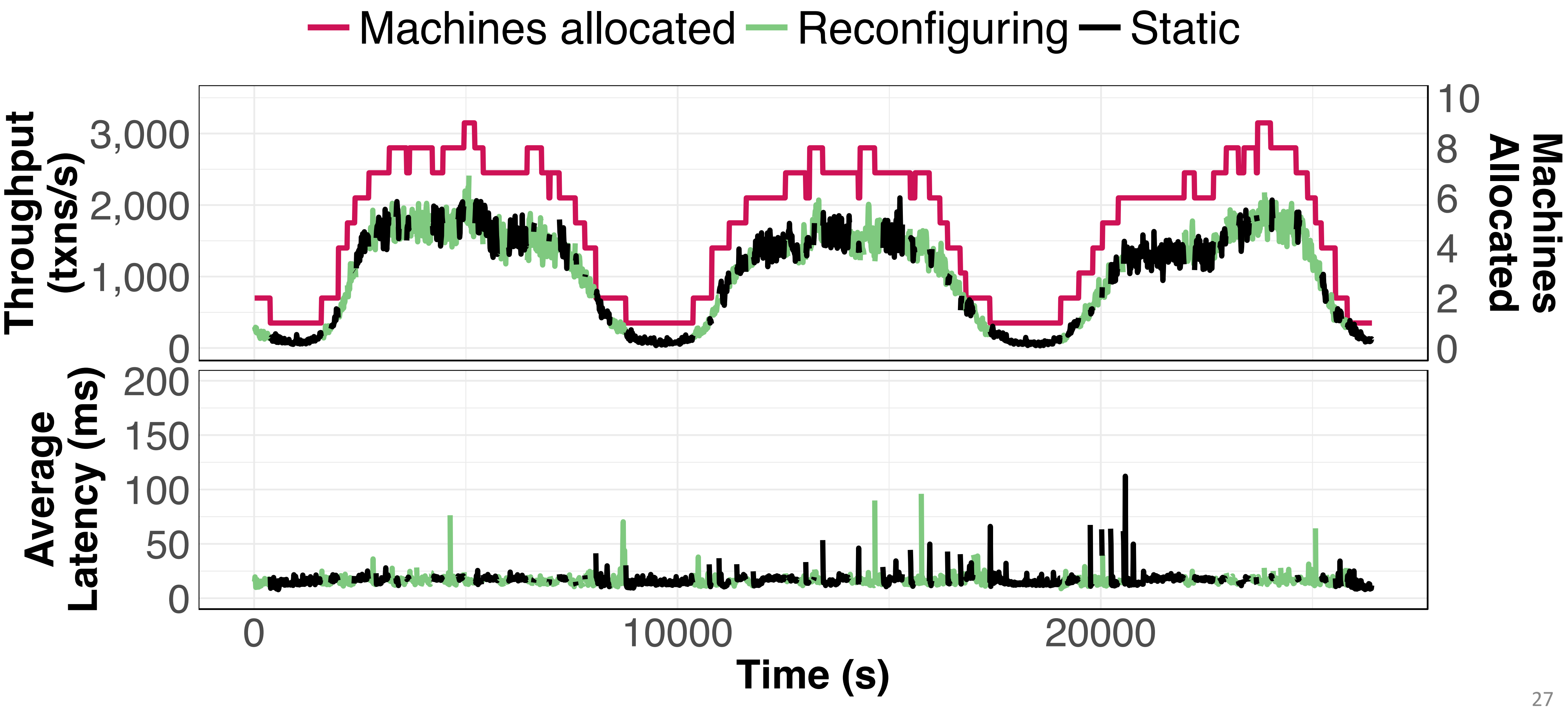
Results – P-Store with SPAR

Avg. Machines Used: **5.05**

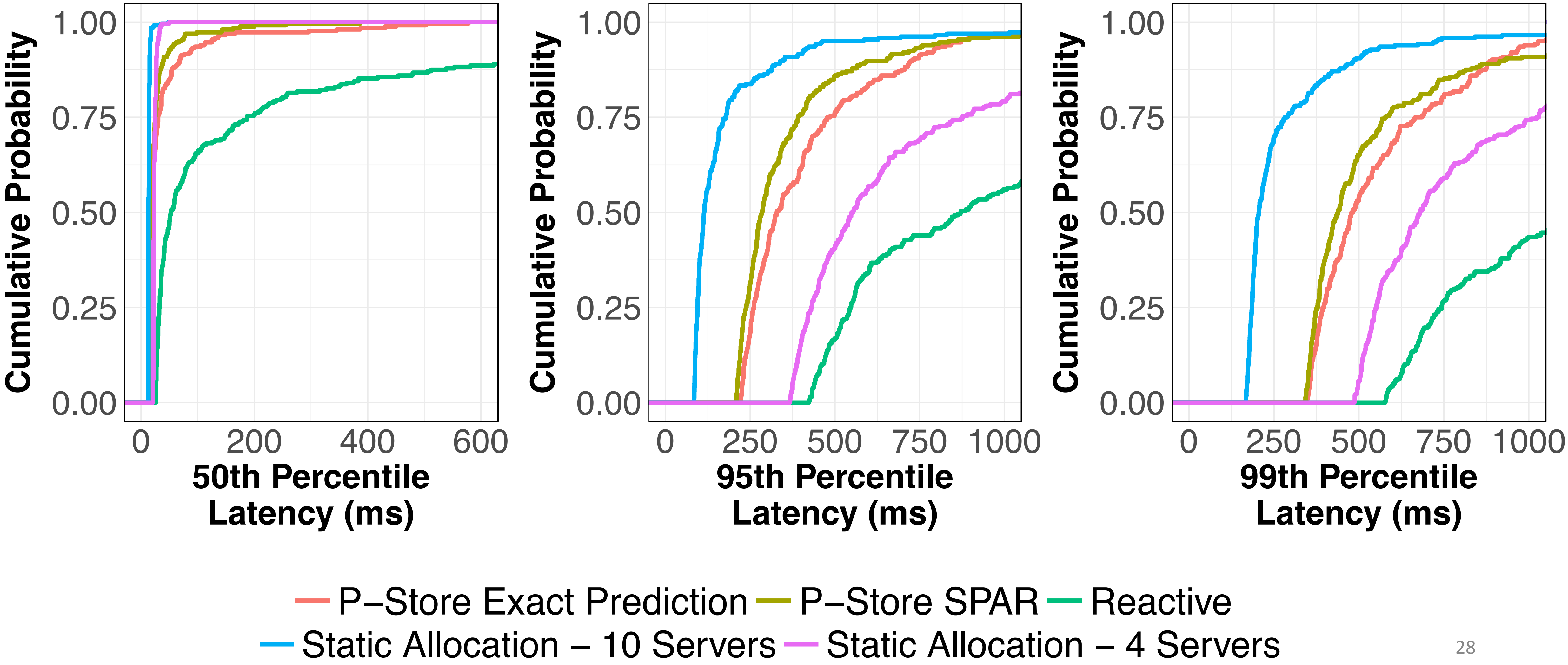


Results – P-Store with Exact Provisioning

Avg. Machines Used: **4.89**



Results Comparison: CDF of Top 1% of Latency



Summary: P-Store Evaluation

- Can we reduce resource usage?
  - Saves 50% of computing resources compared to static allocation
- Can we prevent latency spikes?
  - Superior performance compared to reactive approach

- **On a *real* workload, P-Store reduces resource usage while keeping latency within application requirements**

Summary

- ❖ Real database workloads are skewed and vary over time
- ❖ Elasticity enables management of skew and adaptation to load changes
- ❖ Predictive scaling improves performance during load changes compared to reactive scaling

Rebecca Taft

becca@cockroachlabs.com