STeP: Scalable Tenant Placement for Managing Database-as-a-Service Deployments

Rebecca Taft (MIT CSAIL), Willis Lang (Microsoft), Jennie Duggan (Northwestern University), Aaron J. Elmore (University of Chicago), Michael Stonebraker (MIT CSAIL), David DeWitt (Microsoft)

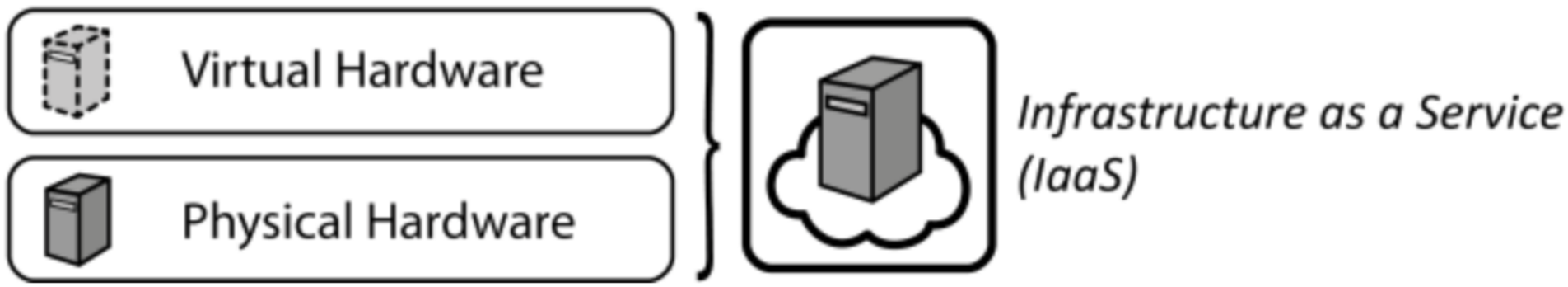
Presented by **Jayasi Mehar**

This paper addresses the question of how to assign DBaaS workloads to hardware resources and examines this issue in the context of Microsoft's Azure SQL Database

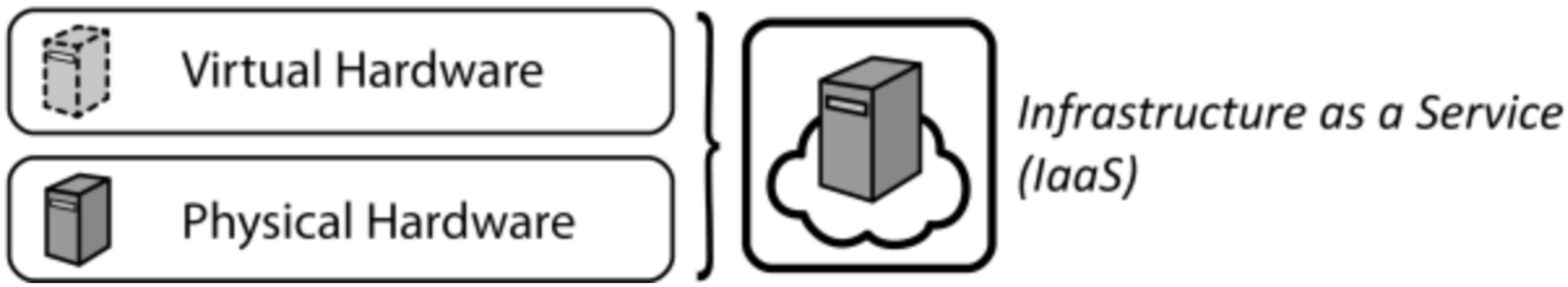
Key Ideas

- Characterizing tenants for placement by analyzing real production scale trace
- Different aspects of the workload:
  - Summary statistics
  - Time series shapelets
  - FFT coefficients
  - Dynamicity
- Two comprehensive cost models and penalties

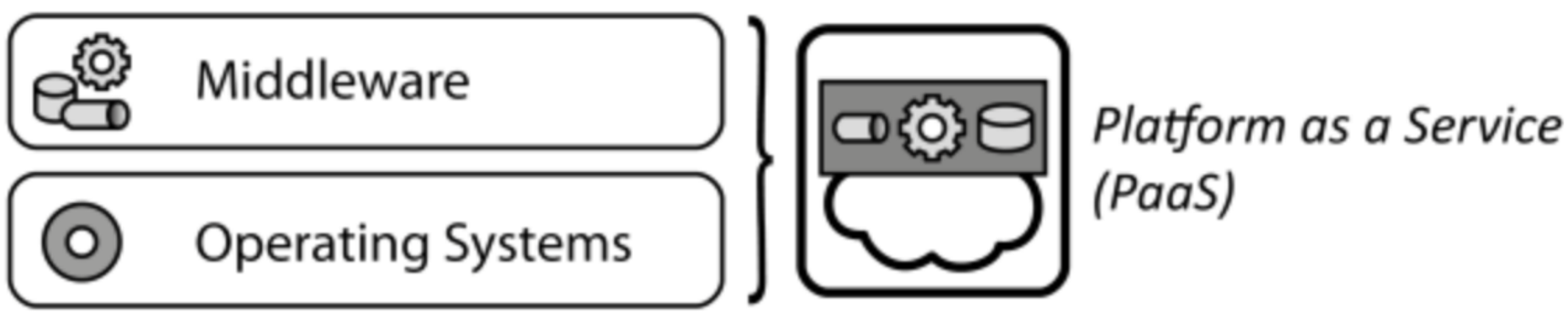
Infrastructure as a service



Infrastructure as a service



Platform as a service



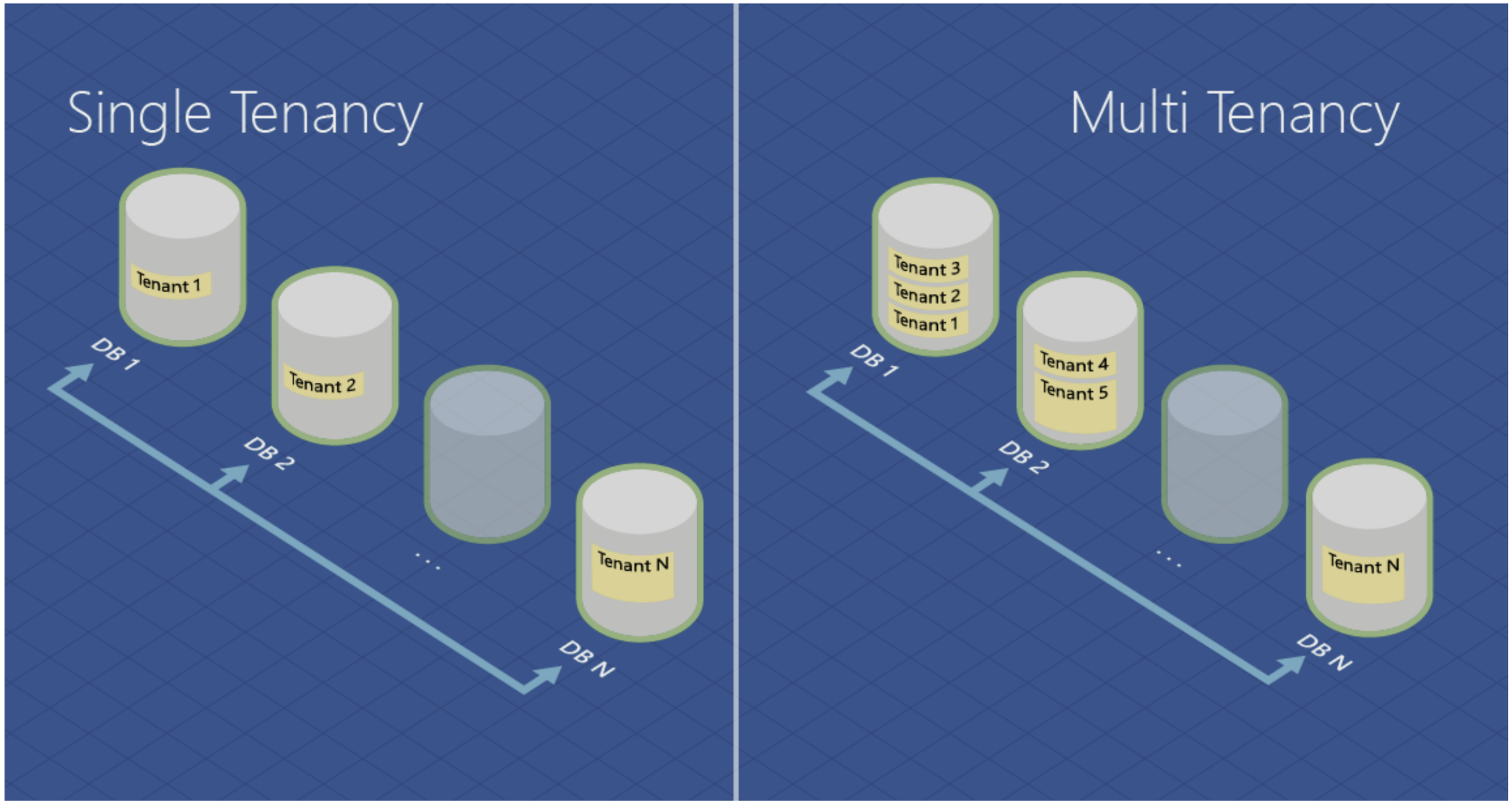
Database as a service

Database as a service

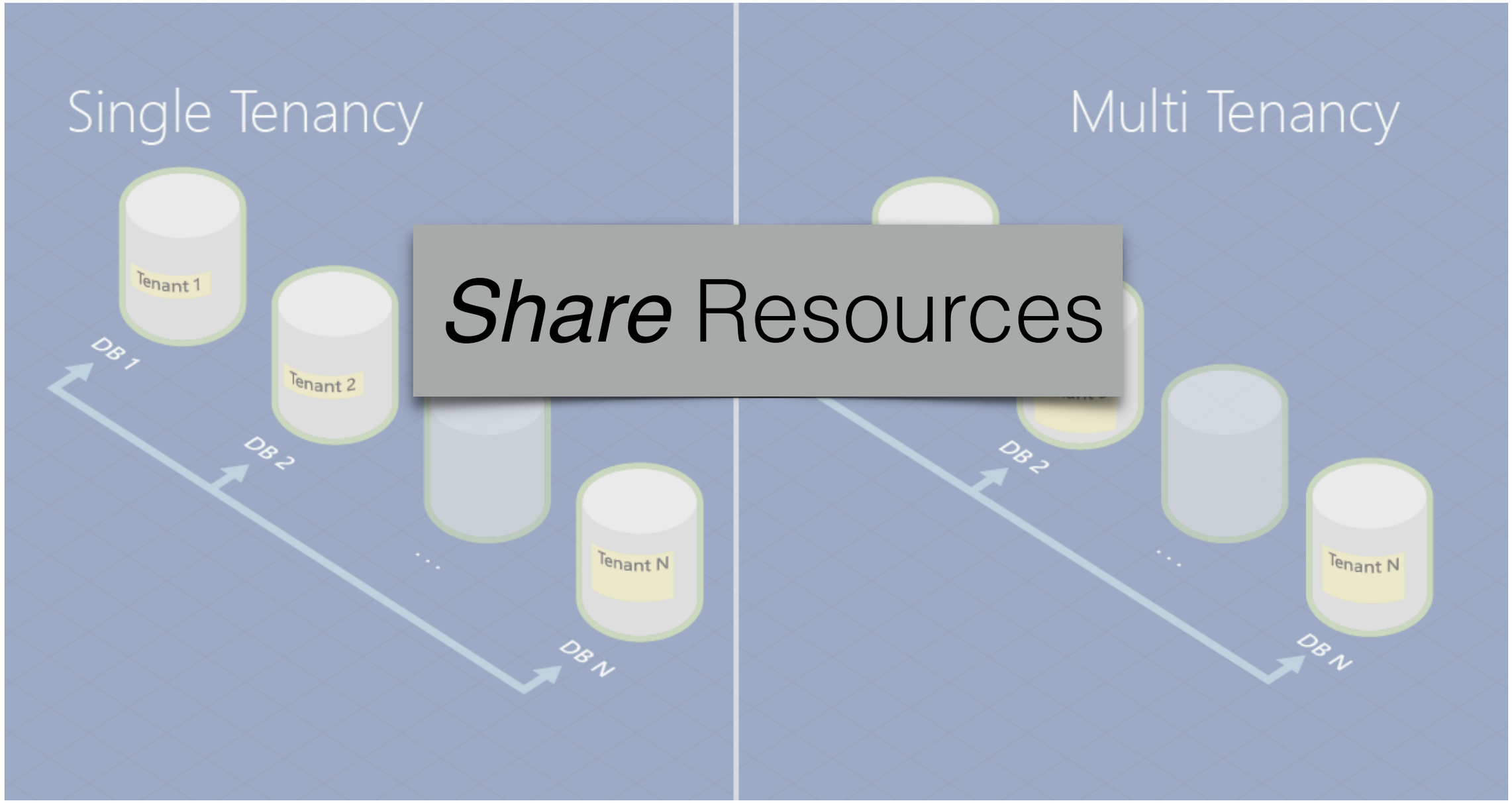
Millions of users, **thousands** of servers

Multi Tenancy

Multi Tenancy



Multi Tenancy



Share Resources

$$$

$$$

- Multi tenancy - economically viable
- One and a half million customer databases
- Resource utilization → cost
- Penalty for service outages → cost

$$$

- Multi tenancy - economically viable
- On

Minimize operation expenses + SLA violation penalties

- Resource utilization → cost
- Penalty for service outages → cost

What *is* a violation?

What *is* a violation?

Any tenant that is present on a machine with **high utilization** of any resources

What *is* a violation?

Any tenant that is present on a machine with **high utilization** of any resources

CPU Usage, Read/Write Pages, Memory

Microsoft Azure SQL Dataset Analysis

Microsoft Azure SQL Dataset Analysis

- Multiple tenants are housed in one physical SQL Server database
- 1 DC - 1 cluster trace, 100 -150 hosts
- Basic, Standard, Premium
- 3 way replication: 1 primary, 2 secondary
- Heterogeneous tenants - SLA objectives, resource usage

Time series readings capture resource utilization of customer database + replicas

Attribute
timestamp
interval_seconds
dbuid
replica_type
machine_id
max_worker_count
cpu_time
logical_reads_pages
logical_writes_pages
log_operations

Time series readings capture resource utilization of customer database + replicas

Attribute
timestamp
interval_seconds
dbuid
replica_type
machine_id
max_worker_count
cpu_time
logical_reads_pages
logical_writes_pages
log_operations

Summed utilization for a time window

Time series readings capture resource utilization of customer database + replicas

Attribute
timestamp
interval_seconds
dbuid
replica_type
machine_id
max_worker_count
cpu_time
logical_reads_pages
logical_writes_pages
log_operations

Used to identify service agreement level

Summed utilization for a time window

Modelling violations

Modelling violations

Any tenant that is present on a machine with **high utilization** of any resources

$$
\begin{aligned}
in\_violation(m,t) = & \left(\sum_{d \in D_{m,t}} \mathrm{READ}_{d,t} \right) > P * \mathrm{MAX\_READ} \vee \\
& \left(\sum_{d \in D_{m,t}} \mathrm{WRITE}_{d,t} \right) > P * \mathrm{MAX\_WRITE} \vee \\
& \left(\sum_{d \in D_{m,t}} \mathrm{CPU}_{d,t} \right) > P * \mathrm{MAX\_CPU} \vee \\
& \left(\sum_{d \in D_{m,t}} \mathrm{LOG}_{d,t} \right) > P * \mathrm{MAX\_LOG}
\end{aligned} \tag{1}
$$

Modelling violations

Any tenant that is present on a machine with **high utilization** of any resources

Threshold Set

$$
\begin{aligned}
in\_violation(m,t) = \Bigg(\sum_{d \in D_{m,t}} \mathrm{READ}_{d,t} \Bigg) &> P * \mathrm{MAX\_READ} \vee \\
\Bigg(\sum_{d \in D_{m,t}} \mathrm{WRITE}_{d,t} \Bigg) &> P * \mathrm{MAX\_WRITE} \vee \\
\Bigg(\sum_{d \in D_{m,t}} \mathrm{CPU}_{d,t} \Bigg) &> P * \mathrm{MAX\_CPU} \vee \\
\Bigg(\sum_{d \in D_{m,t}} \mathrm{LOG}_{d,t} \Bigg) &> P * \mathrm{MAX\_LOG}
\end{aligned}
\tag{1}
$$

SLA Penalties

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times when outage > 1%

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times when outage > 1%

y: machines
G: cost of running

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times v
  outage > 1%

y: machines
G: cost of running

SM: Scaling Factor
Q: No. of Months
v(d,q): percentage of violations

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times when outage > 1%

y: machines
G: cost of running

SM: Scaling Factor
Q: No. of Months
v(d,q): percentage of violations

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

violations*active_machines

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times when outage > 1%

y: machines
G: cost of running

SM: Scaling Factor
Q: No. of Months
v(d,q): percentage of violations

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

violations*active_machines

PMAX

- Weigh penalties by customers expected level of service
- Violations for high paying customers is costly

$$cost = y \times G + \frac{S_P}{Q} \sum_{t=1}^{N} \sum_{m=1}^{y} (in\_violation(m,t) * \sum_{d \in D_{m,t}} \frac{w_d}{10})$$

SLA Penalties

MSFT

- Penalty for customers experiencing availability outage > 0.1%
- Penalty 2.5 times when outage > 1%

y: machines
G: cost of running

SM: Scaling Factor
Q: No. of Months
v(d,q): percentage of violations

$$cost = y \times G + \frac{S_M}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})$$

$$v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (in\_violation(m_{d,t}, t) * is\_active(d,t))$$

violations*active_machines

PMAX

- Weigh penalties by customers expected level of service
- Violations for high paying customers is costly

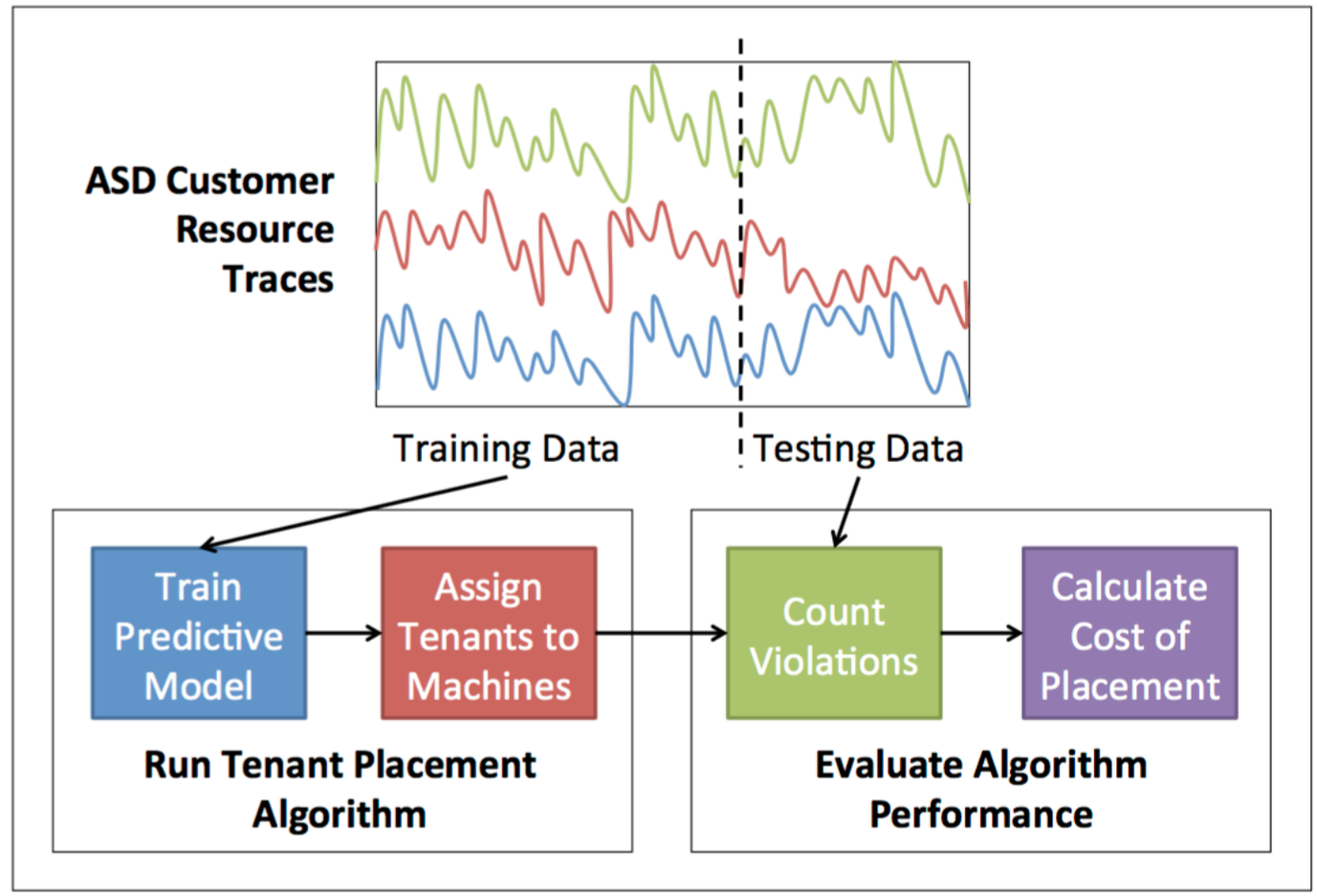
$$cost = y \times G + \frac{S_P}{Q} \sum_{t=1}^{N} \sum_{m=1}^{y} (in\_violation(m,t) * \sum_{d \in D_{m,t}} \frac{w_d}{10})$$

violations*service_level

Proposal

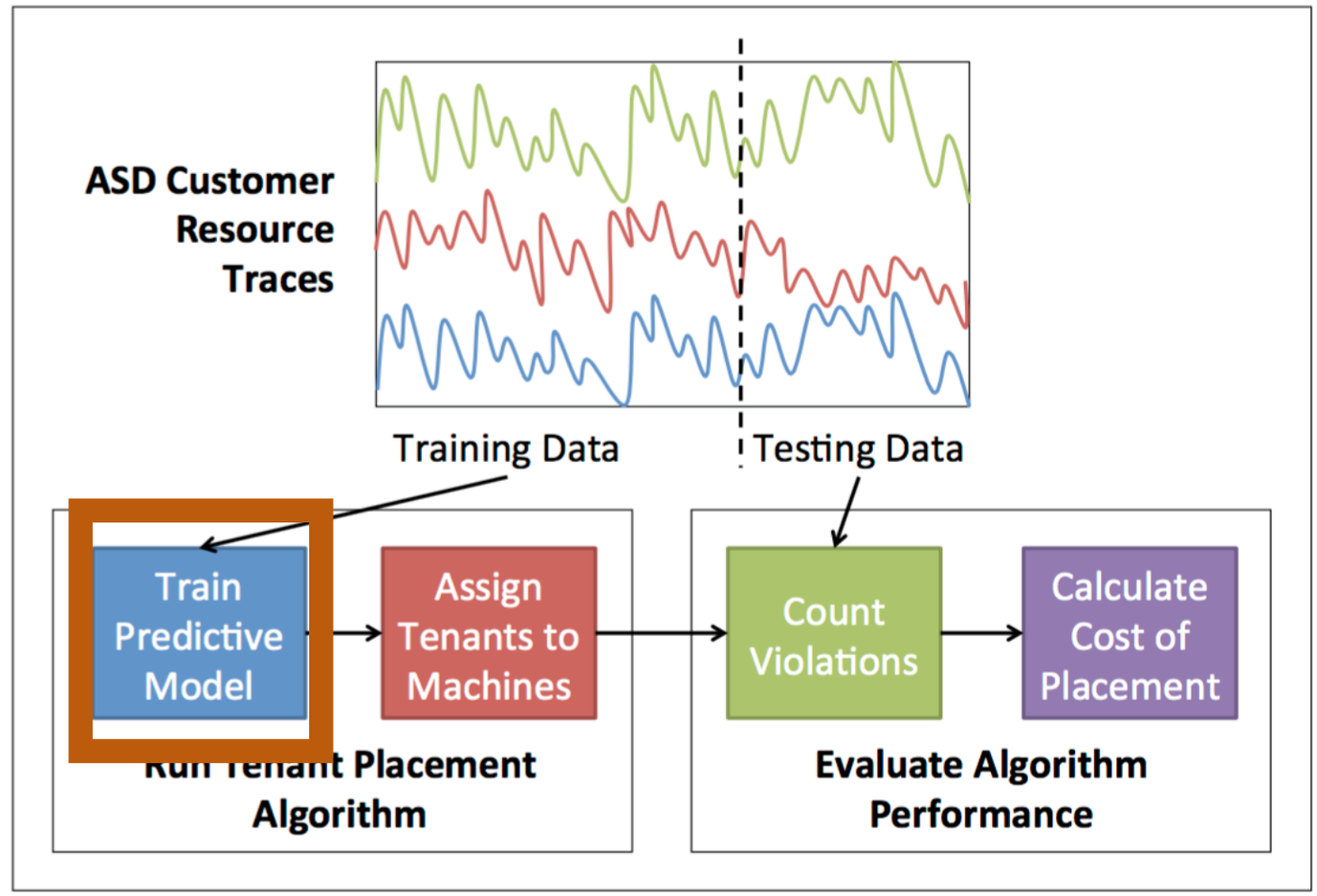
Proposal

- **STeP** (Scalable Tenant Placement)
- Predictive models
- Efficient and robust to load spikes



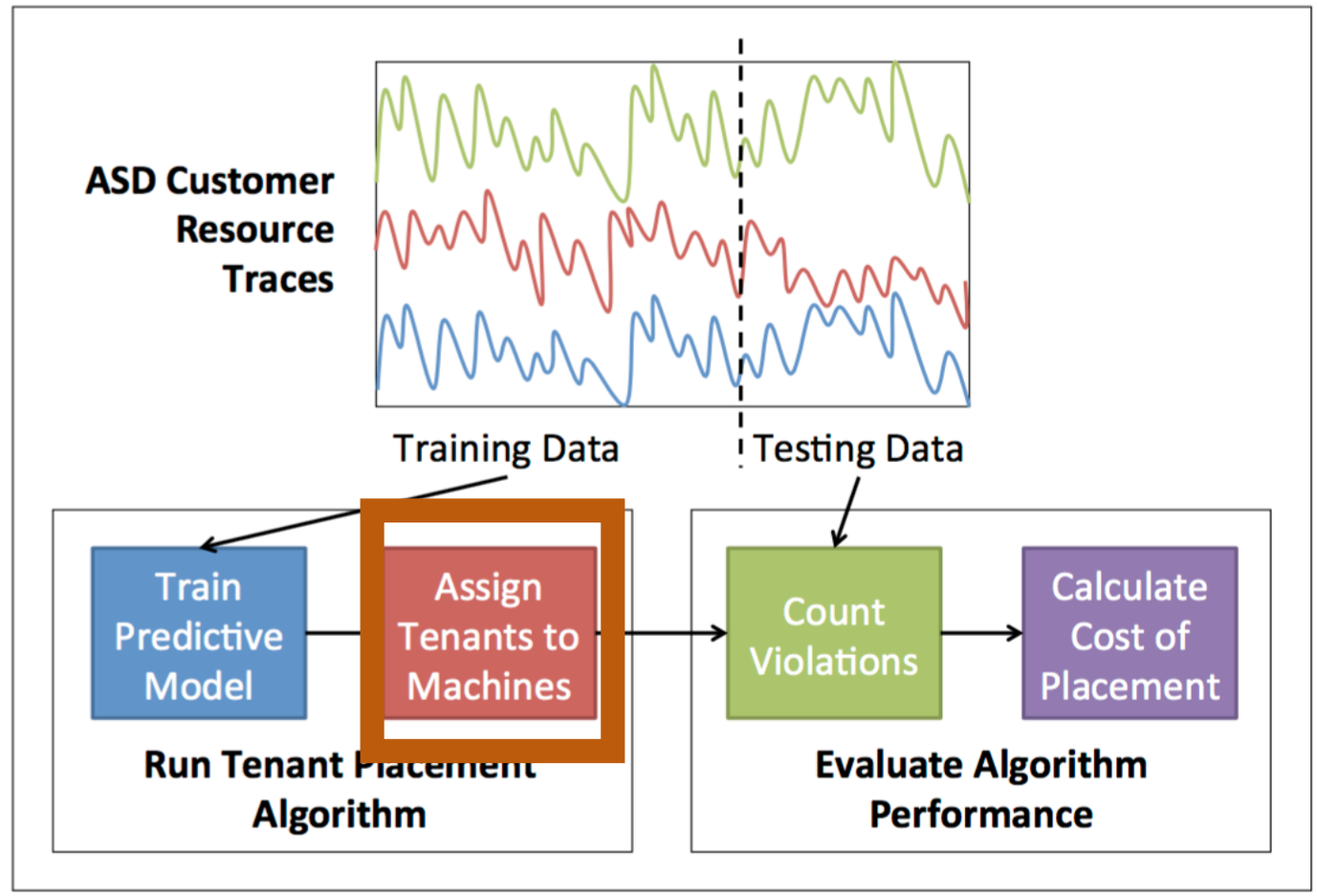
Proposal

- **STeP** (Scalable Tenant Placement)
- Predictive models
- Efficient and robust to load spikes



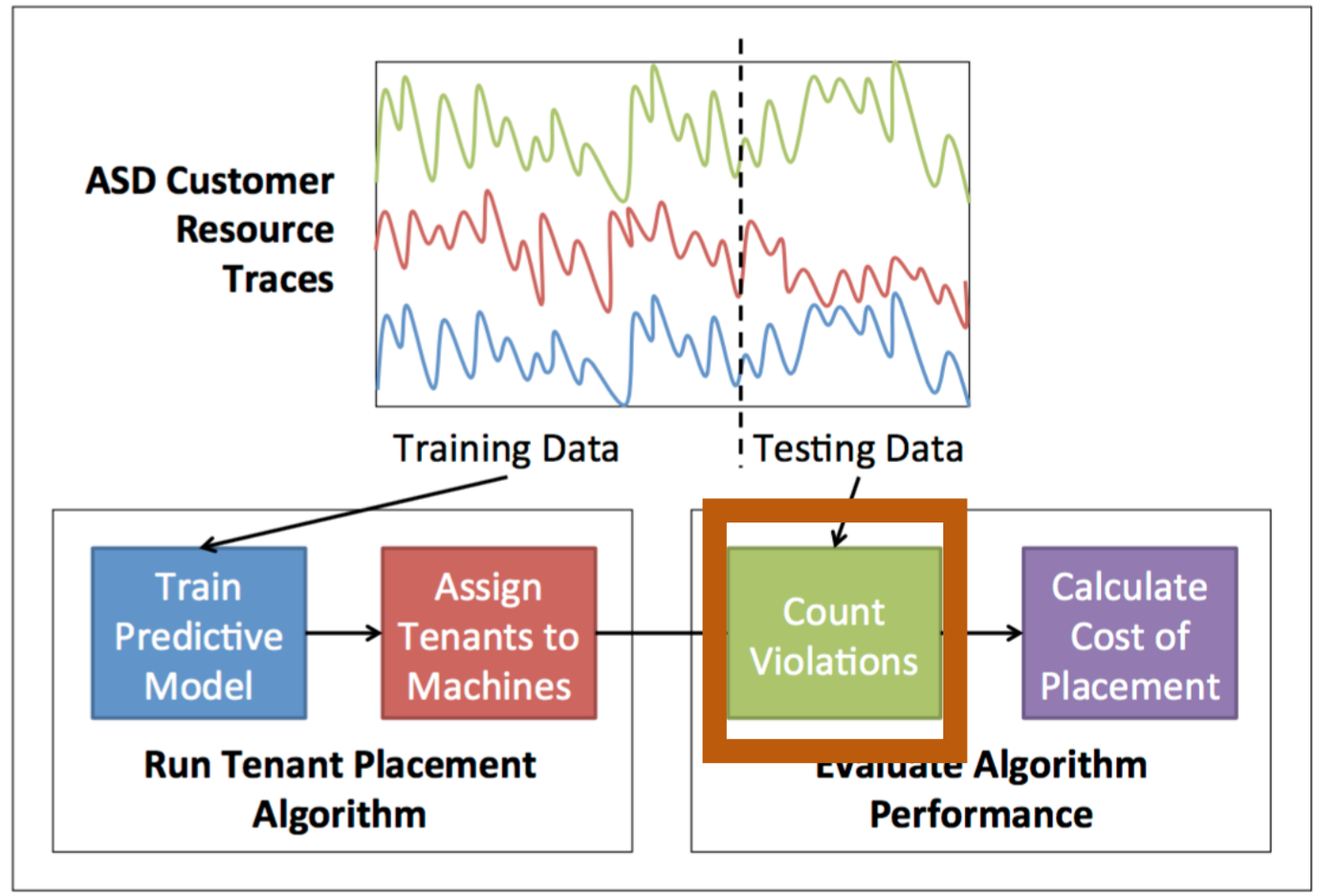
Proposal

- **STeP** (Scalable Tenant Placement)
- Predictive models
- Efficient and robust to load spikes



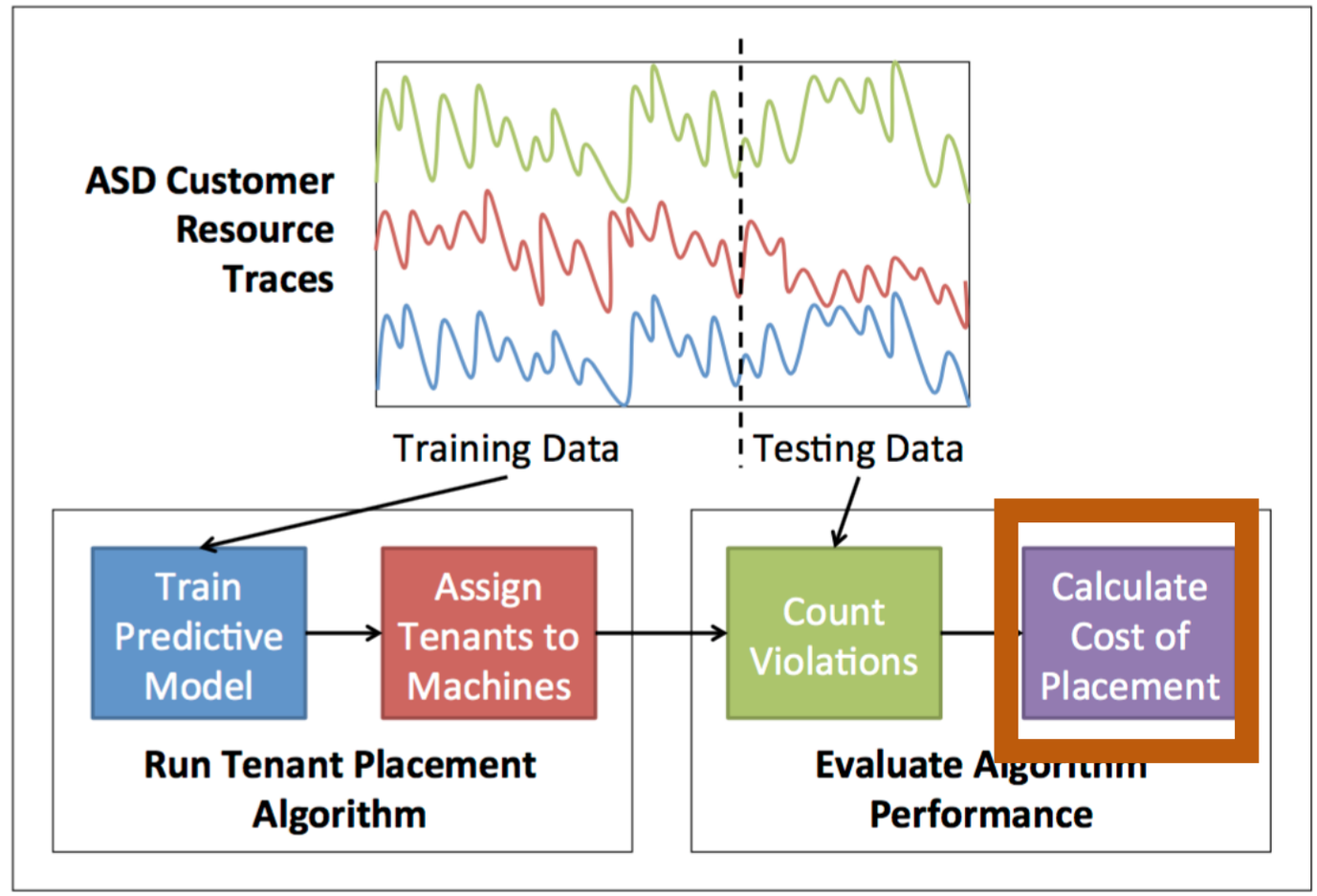
Proposal

- **STeP** (Scalable Tenant Placement)
- Predictive models
- Efficient and robust to load spikes



Proposal

- **STeP** (Scalable Tenant Placement)
- Predictive models
- Efficient and robust to load spikes



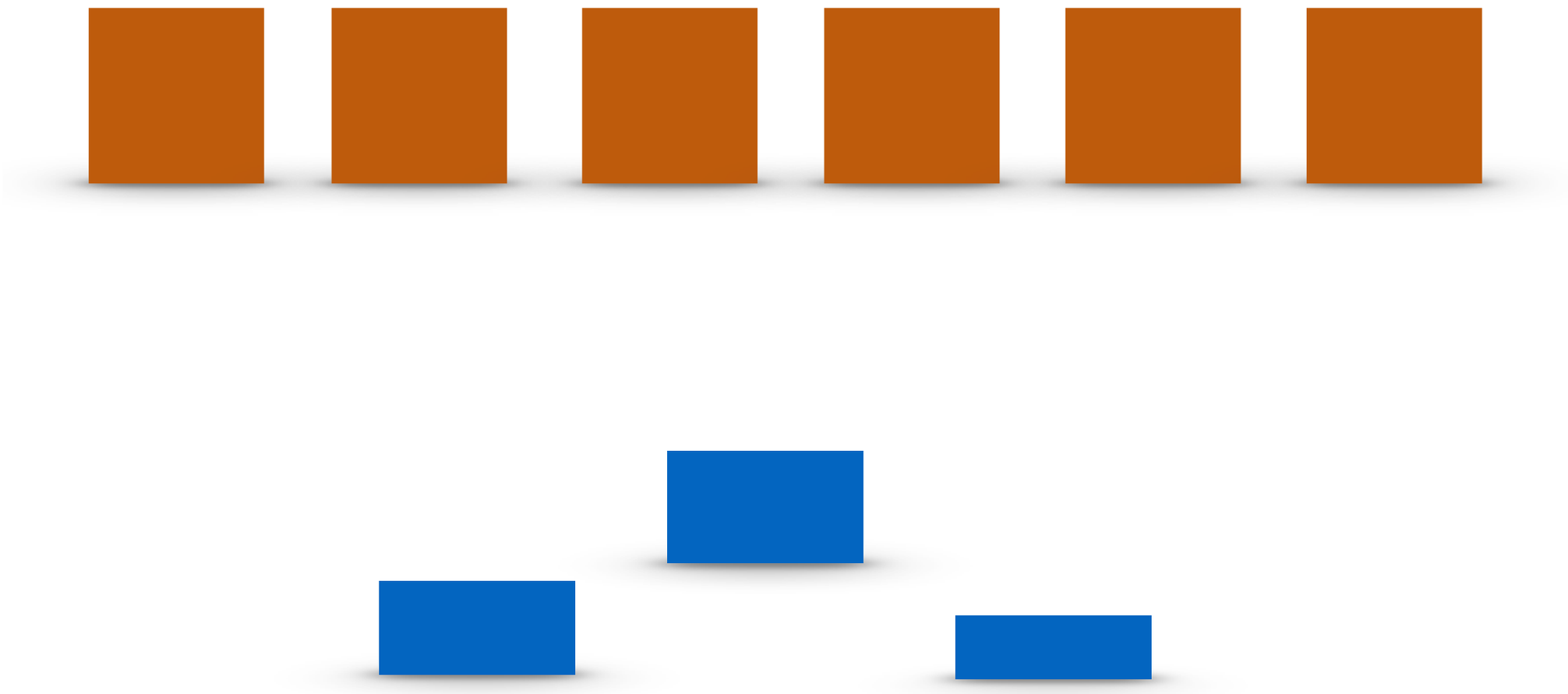
Tenant Placement Algorithms

(The fun part)

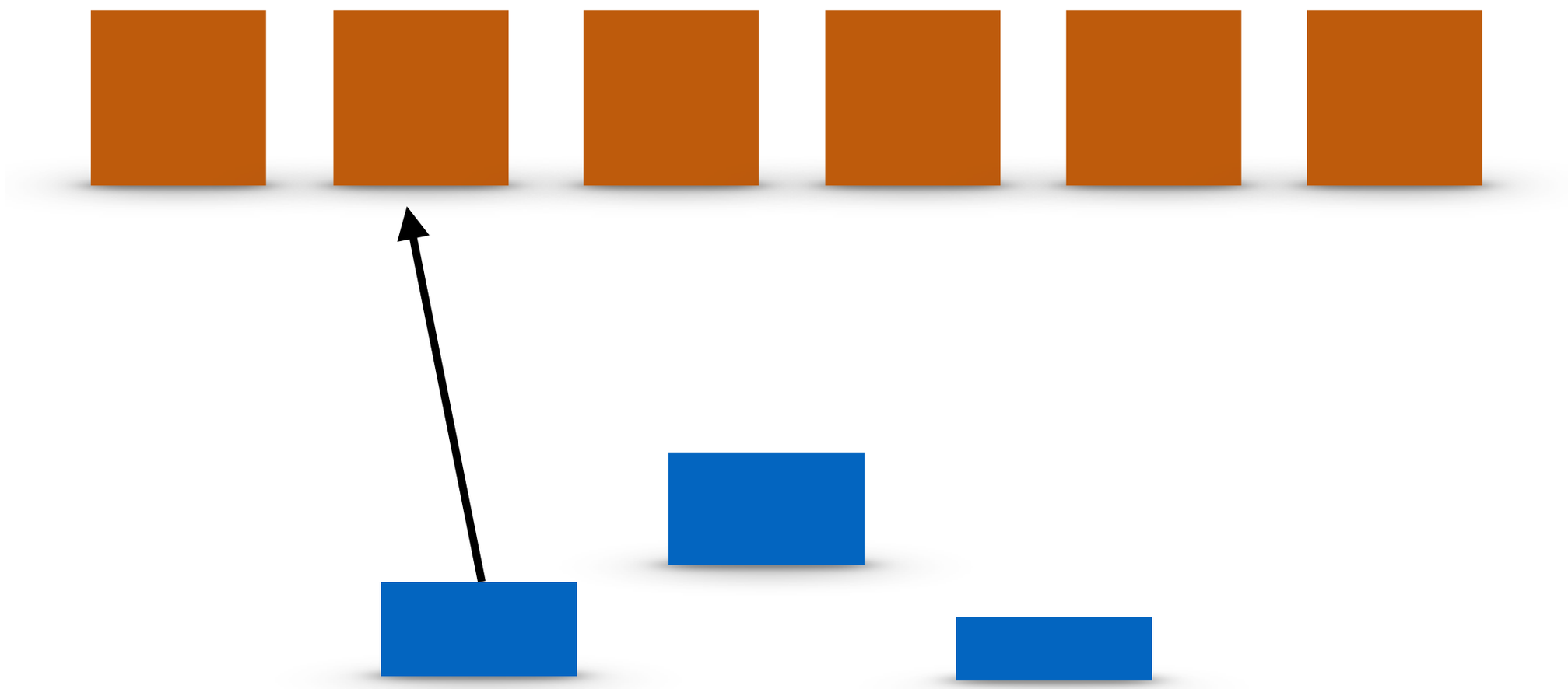
Random

Random

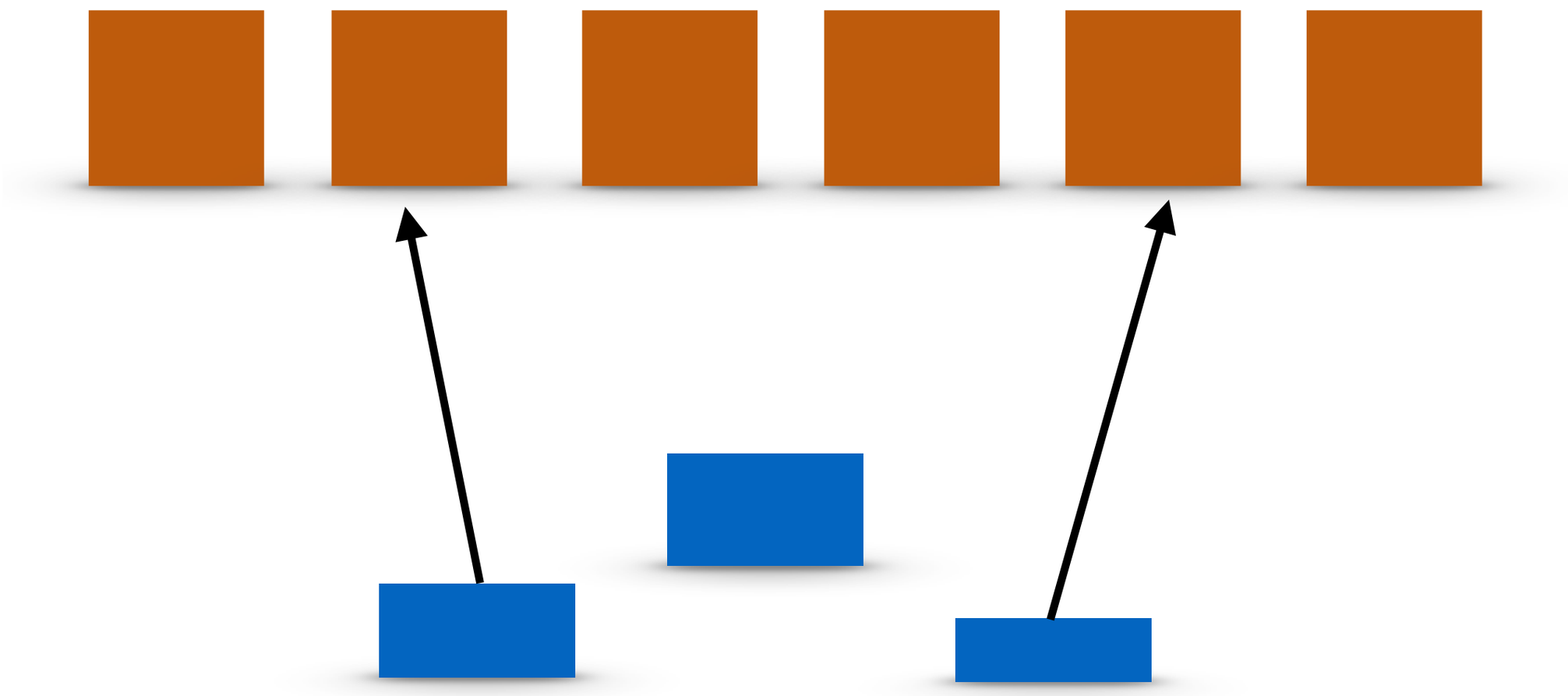
Random



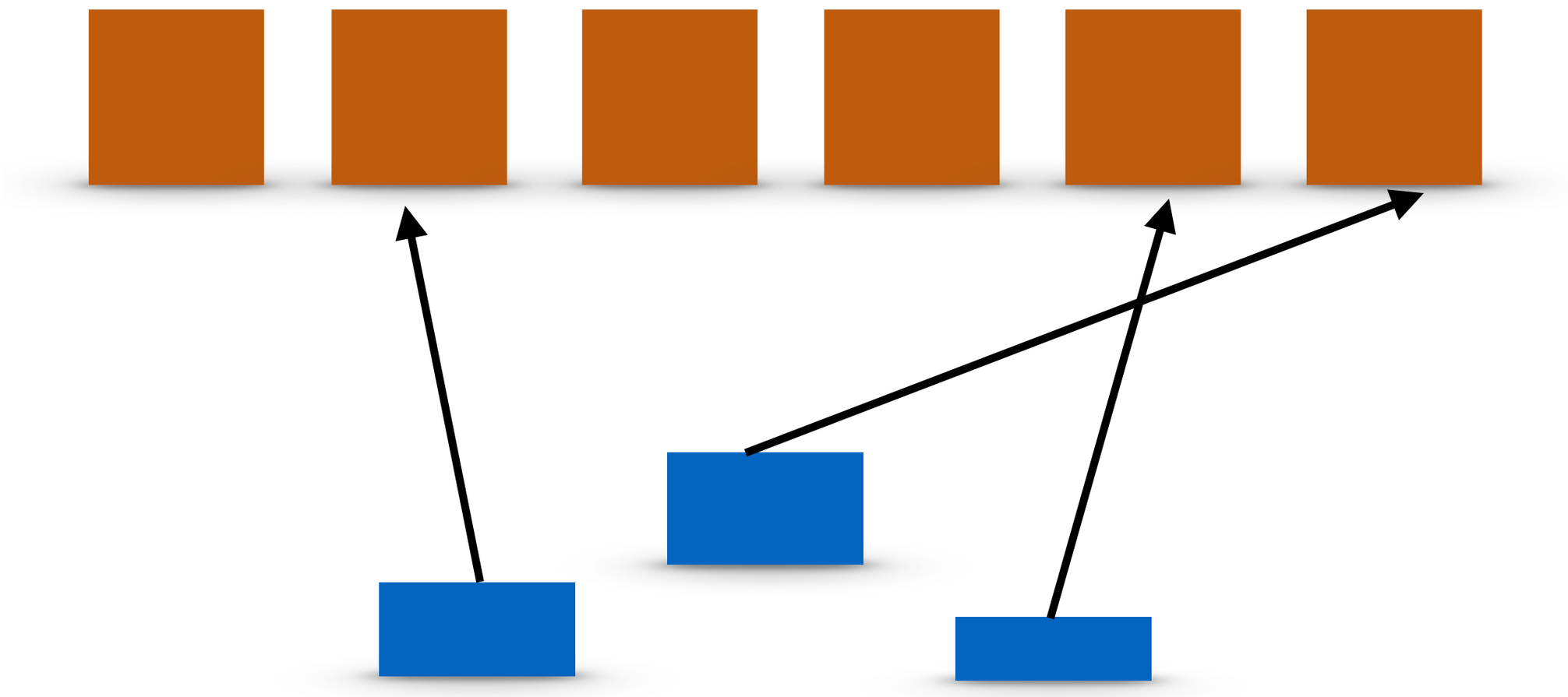
Random



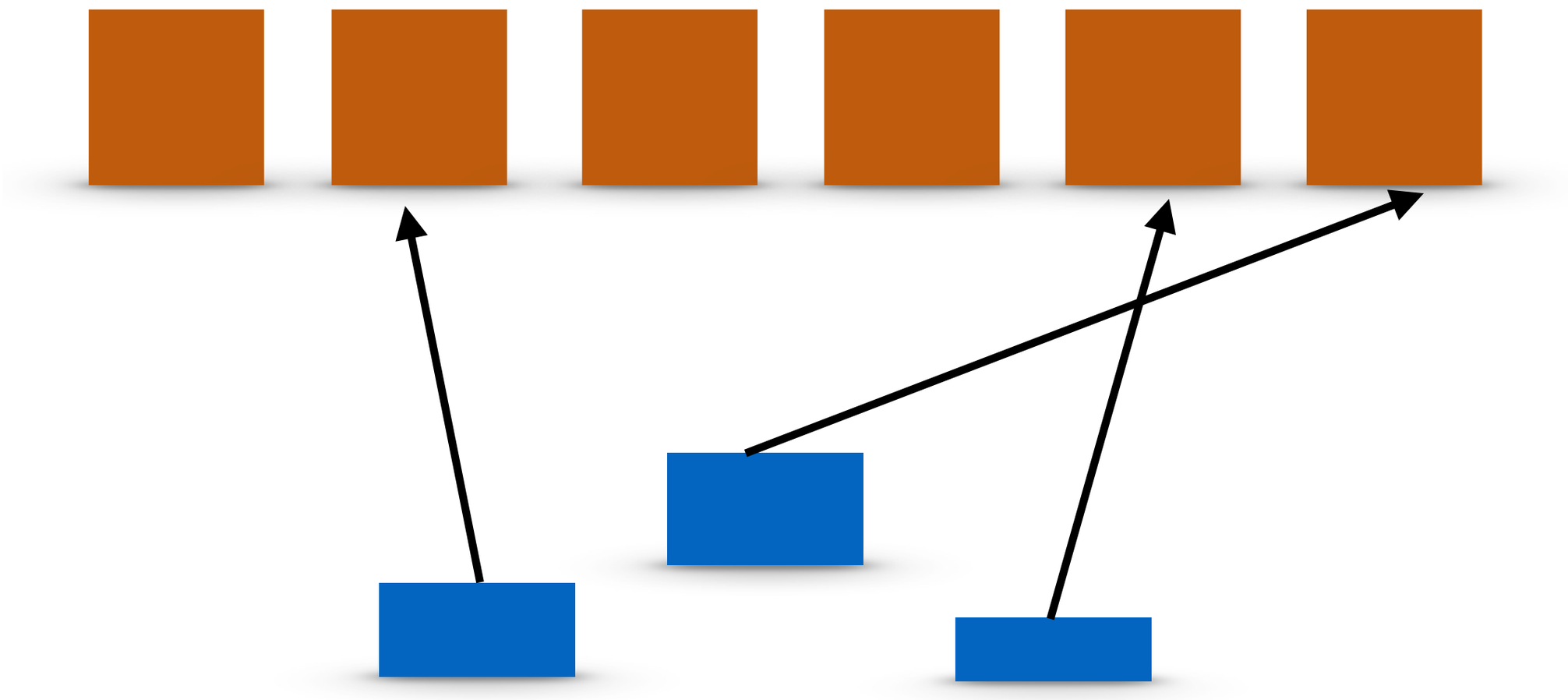
Random



Random



Random



Works quite well, actually

Greedy - Scalar Cost Model

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

CPU Usage of DB at **time t**

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

CPU Usage of DB at **time t**

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

CPU Usage of DB at **time t**

Number of **time slices**

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

CPU Usage of DB at **time t**

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

Number of **time slices**

$$a(w_d, z) = \begin{cases} \operatorname*{avg}_{s \in D_z, w_s = w_d} (base_{s,z}) & \text{if } \exists s \in D_z \text{ s.t. } w_s = w_d \\ \operatorname*{avg}_{s \in D_z} (base_{s,z}) & \text{otherwise} \end{cases}$$

Greedy - Scalar Cost Model

Cost model based on average CPU usage:

$$cost_{d,z} = base_{d,z} + \lambda * a(w_d, z)$$

CPU Usage of DB at **time t**

$$base_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})$$

Number of **time slices**

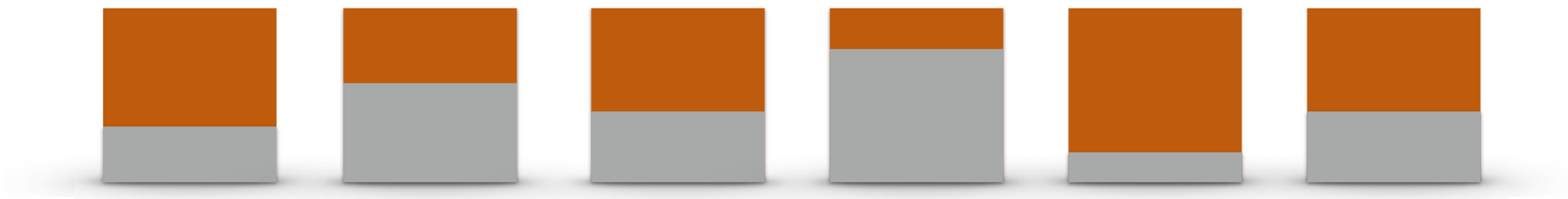
Additional cost: Avg of all DBs in same SLA level

$$a(w_d, z) = \begin{cases} \operatorname*{avg}_{s \in D_z, w_s = w_d} (base_{s,z}) & \text{if } \exists s \in D_z \text{ s.t. } w_s = w_d \\ \operatorname*{avg}_{s \in D_z} (base_{s,z}) & \text{otherwise} \end{cases}$$

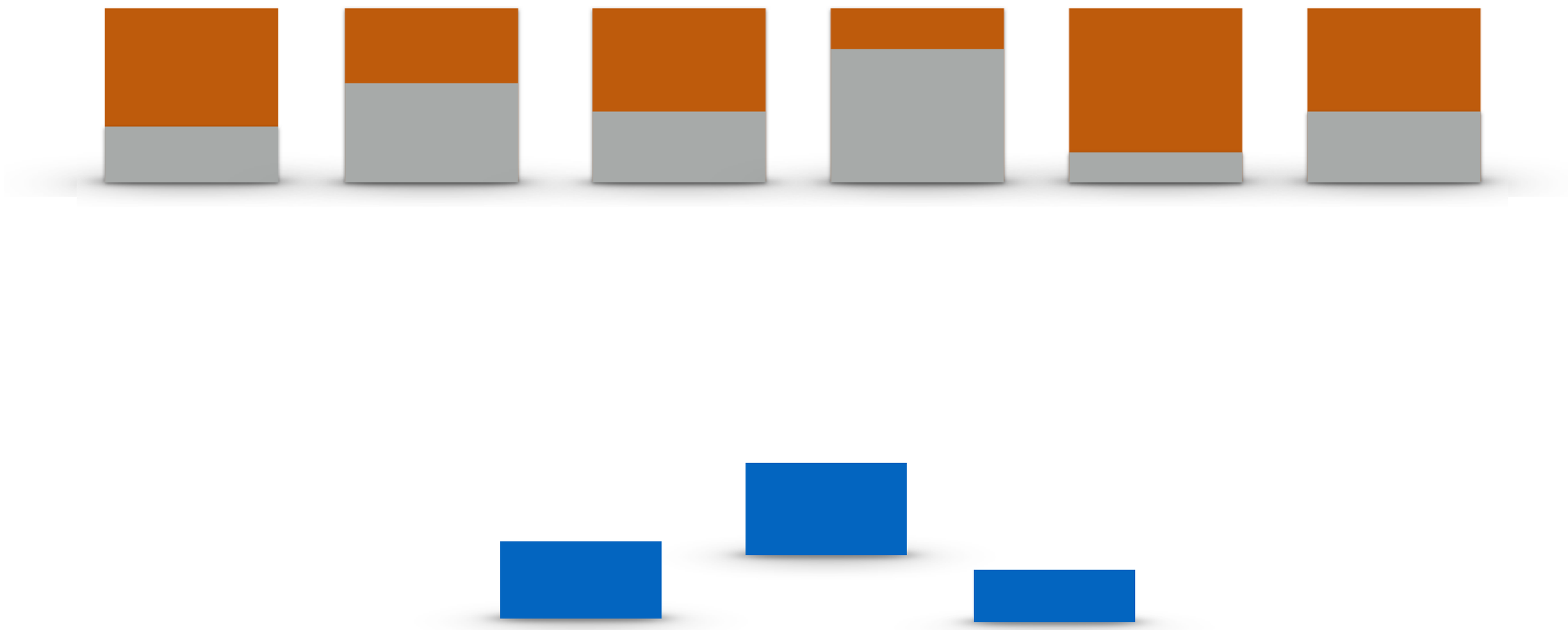
Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines

Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines

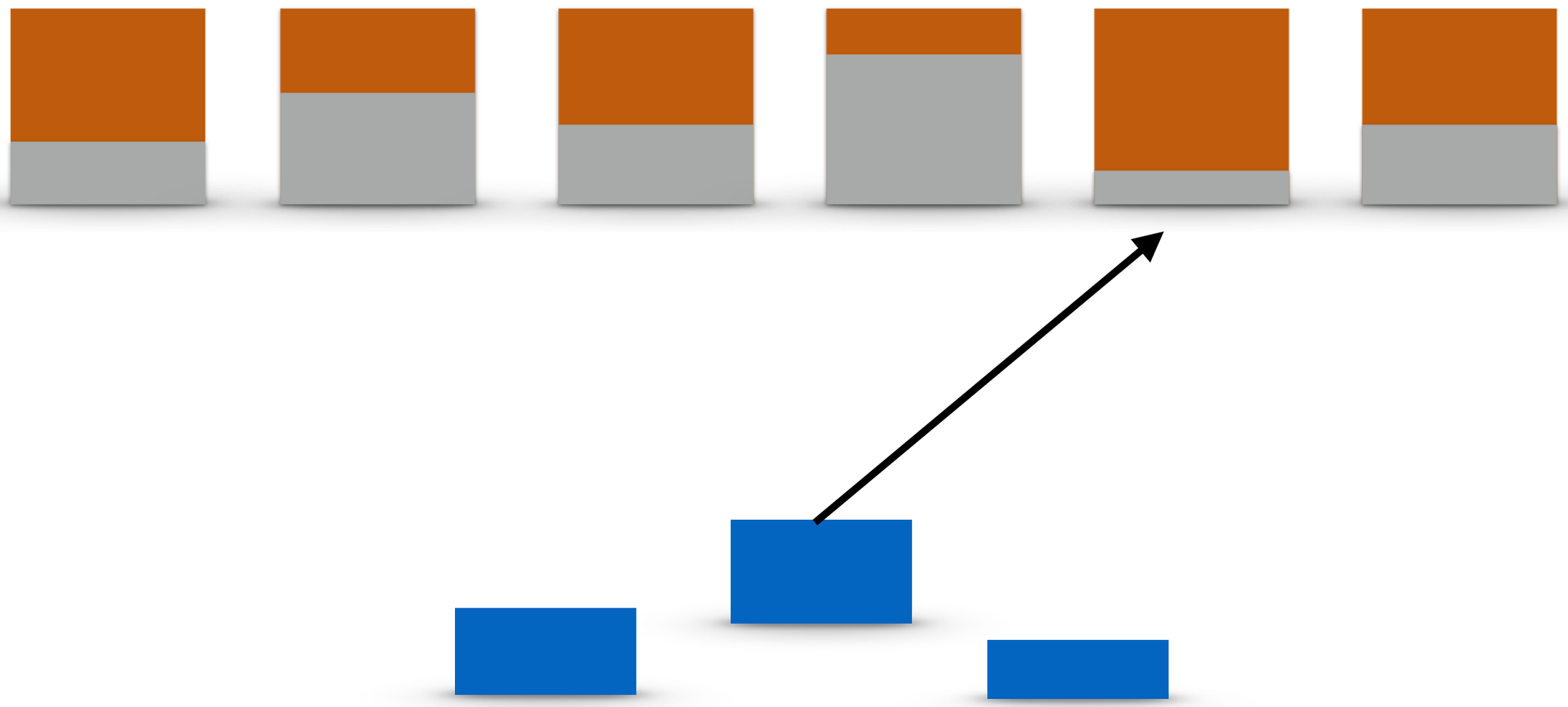
Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



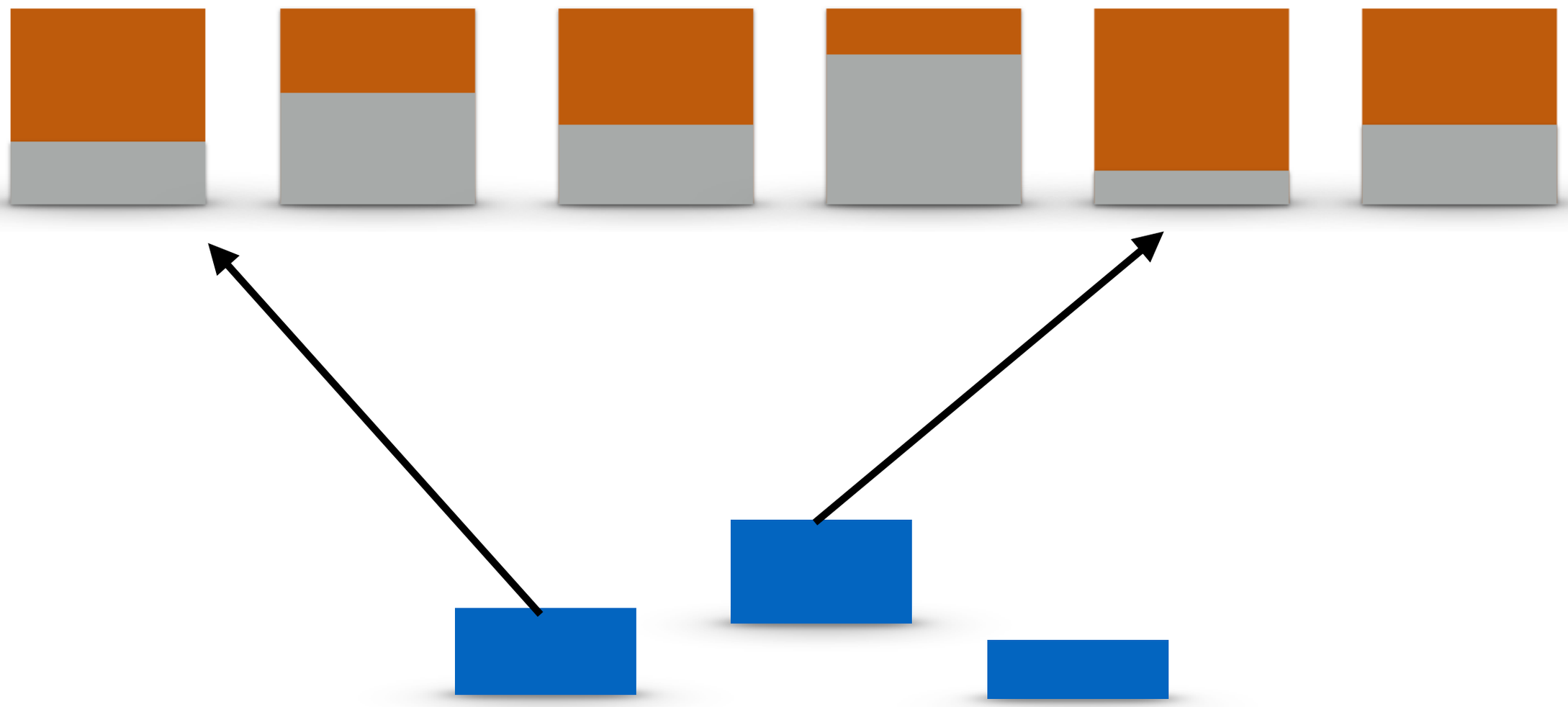
Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



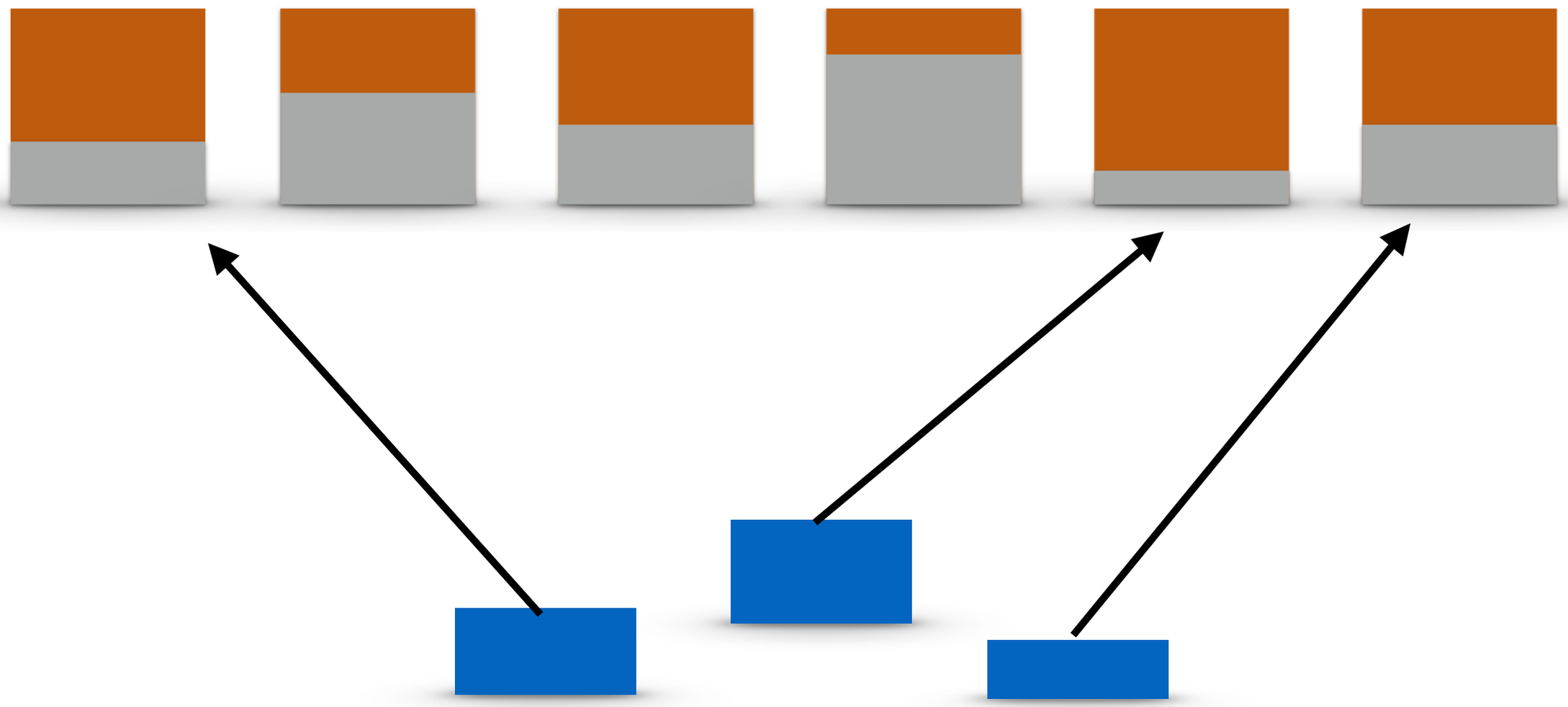
Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



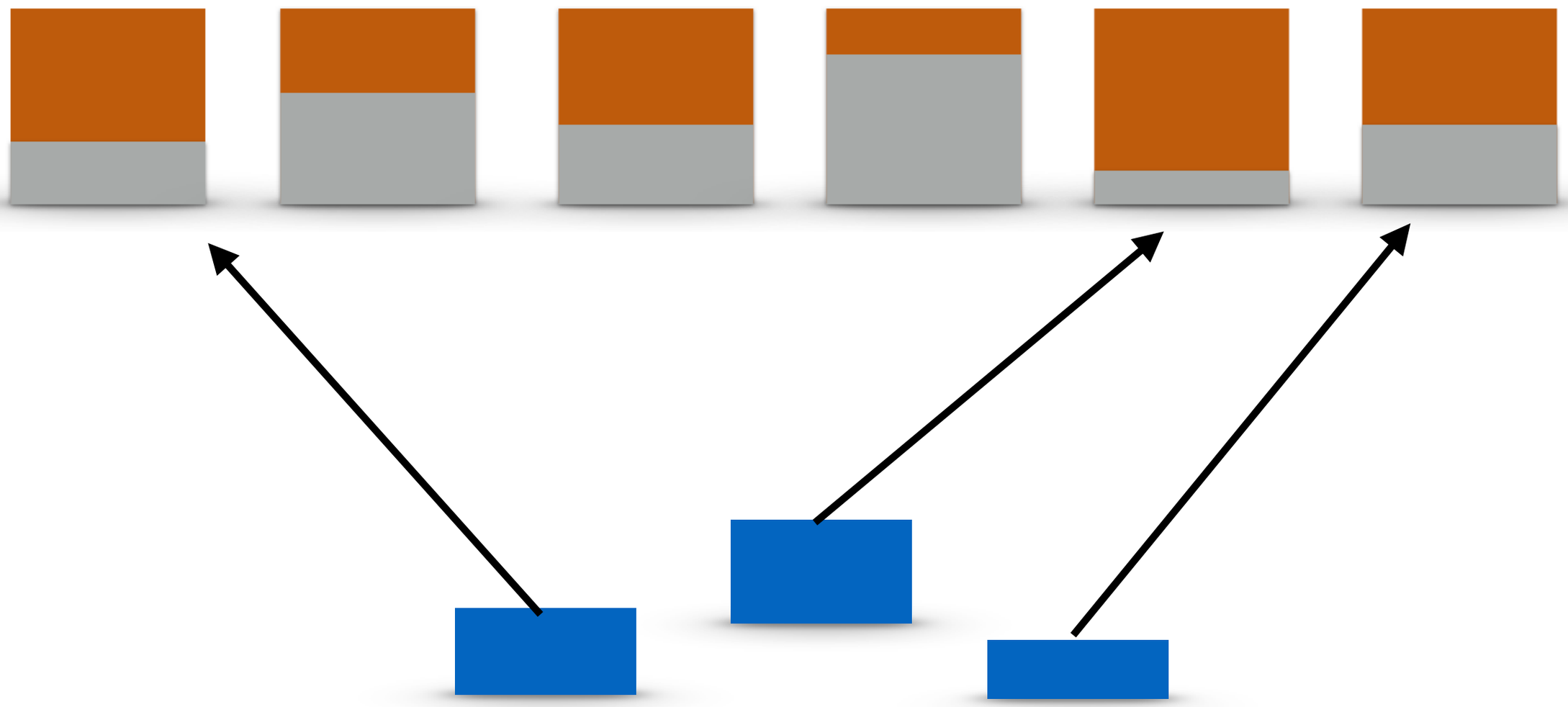
Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



Greedy bin packer algorithm: Evenly distribute the summed cost of the databases across the machines



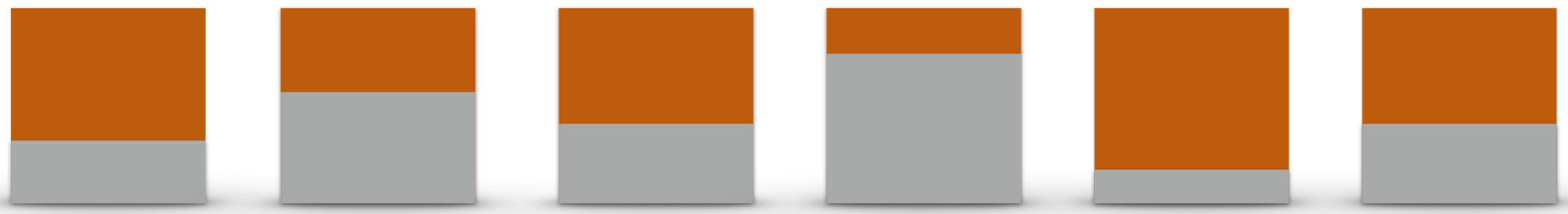
“Worst Fit”!

Best Fit Algorithm?

Sorted by increasing cost, place on most loaded

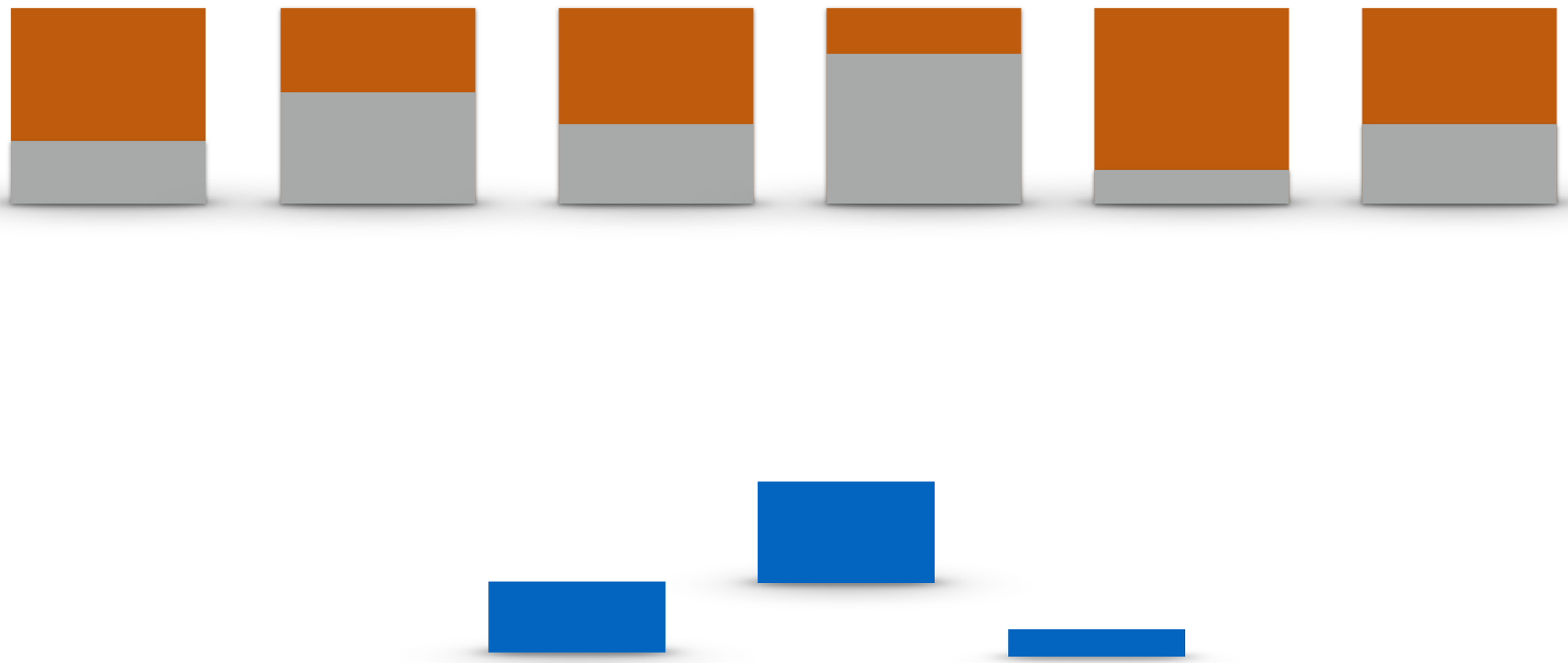
Best Fit Algorithm?

Sorted by increasing cost, place on most loaded



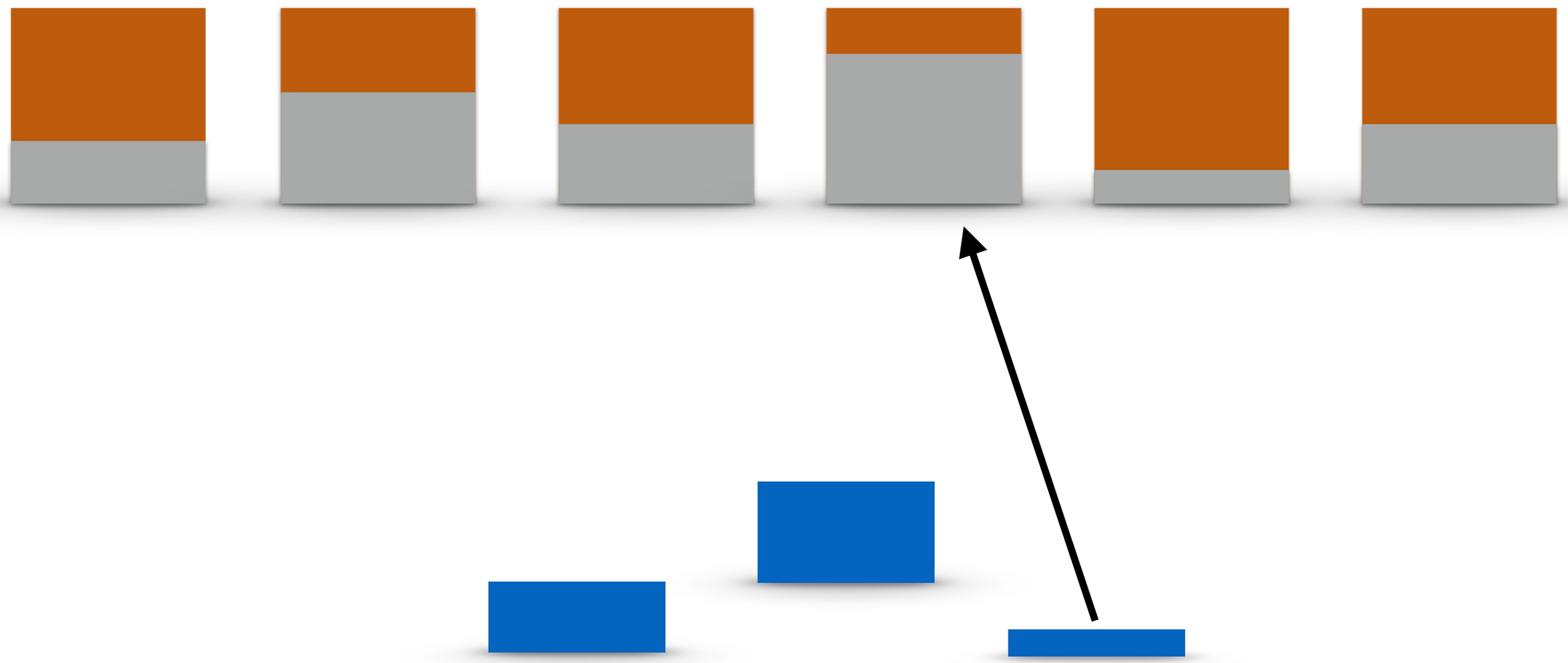
Best Fit Algorithm?

Sorted by increasing cost, place on most loaded



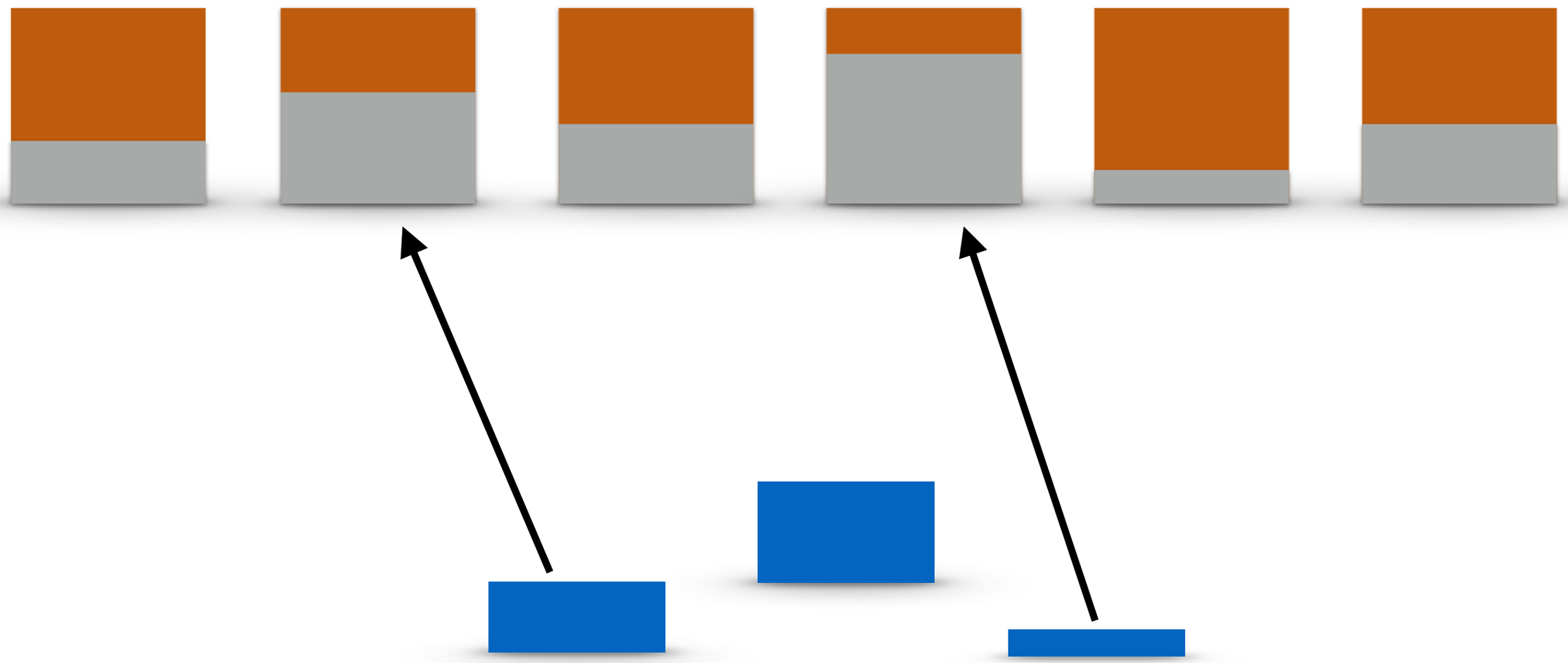
Best Fit Algorithm?

Sorted by increasing cost, place on most loaded



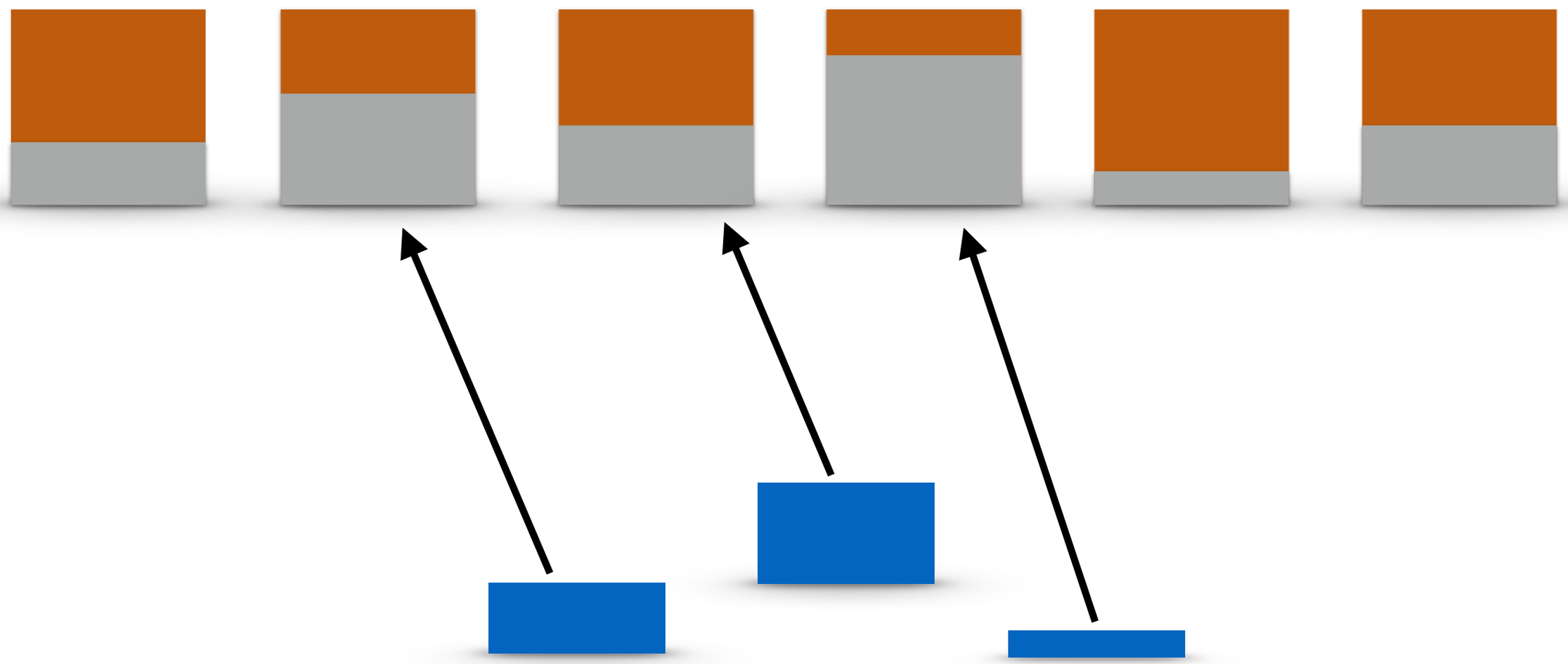
Best Fit Algorithm?

Sorted by increasing cost, place on most loaded



Best Fit Algorithm?

Sorted by increasing cost, place on most loaded



Can we do even better by taking advantage of the training data at a finer granularity?

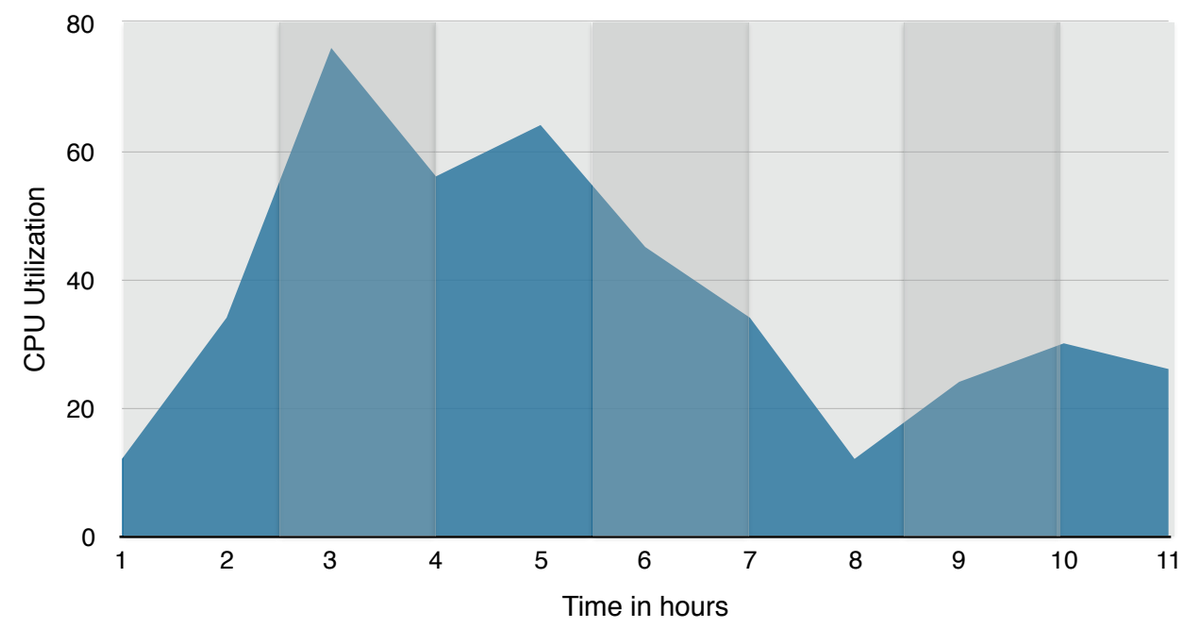
Summed Time Series Cost Model

Collocate databases if their resource usage time series are anti-correlated

Summed Time Series Cost Model

Collocate databases if their resource usage time series are anti-correlated

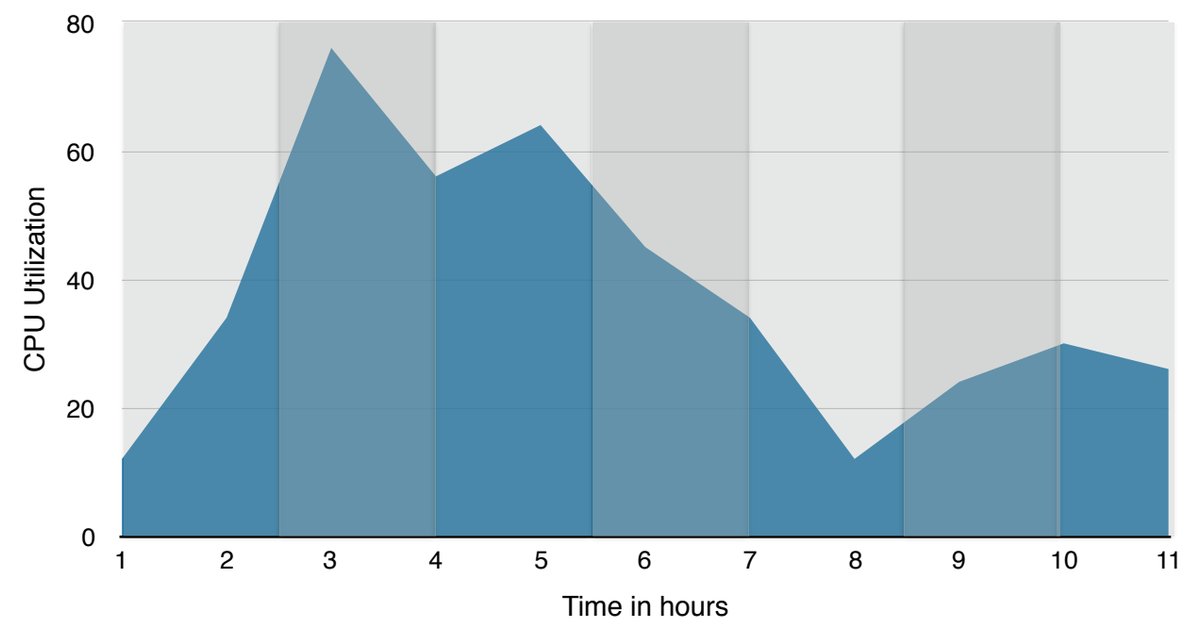
- Aggregated each DB time series into **k** segments



Summed Time Series Cost Model

Collocate databases if their resource usage time series are anti-correlated

- Aggregated each DB time series into **k** segments



- The load on machine **m** during training period **z** is:

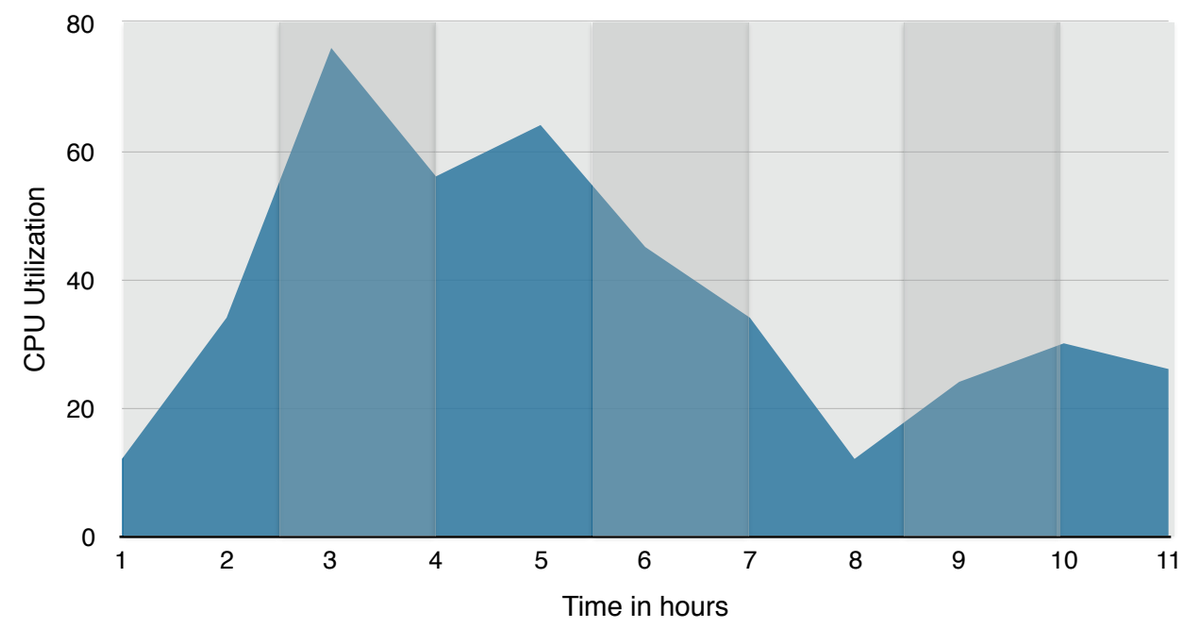
$$L_{m,z} = \max_{i=1}^{k}\left(\sum_{d \in D_m} (cost_{d,z_i})\right)$$

- Choose machine having minimum value of maximum summed time series

Summed Time Series Cost Model

Collocate databases if their resource usage time series are anti-correlated

- Aggregated each DB time series into **k** segments



- The load on machine **m** during training period **z** is:

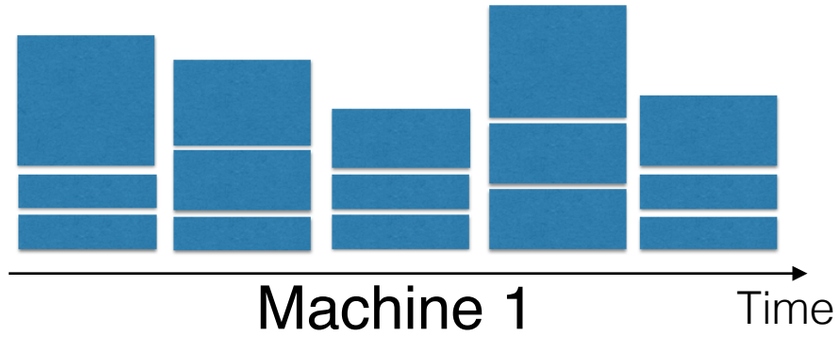
$$L_{m,z} = \max_{i=1}^{k}\left(\sum_{d \in D_m} (cost_{d,z_i})\right)$$

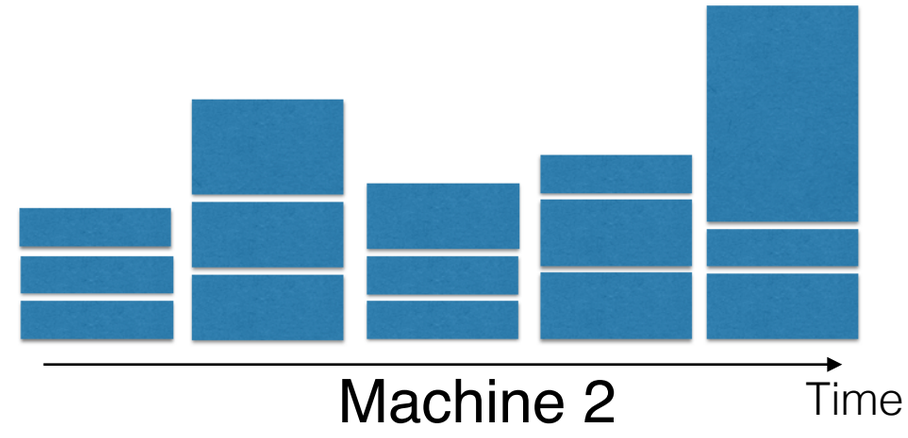
Maximum cost in the given time slices

- Choose machine having minimum value of maximum summed time series

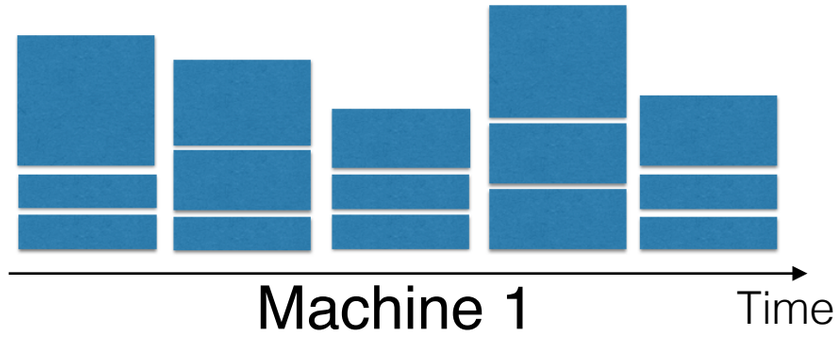

Time
Tenant to place

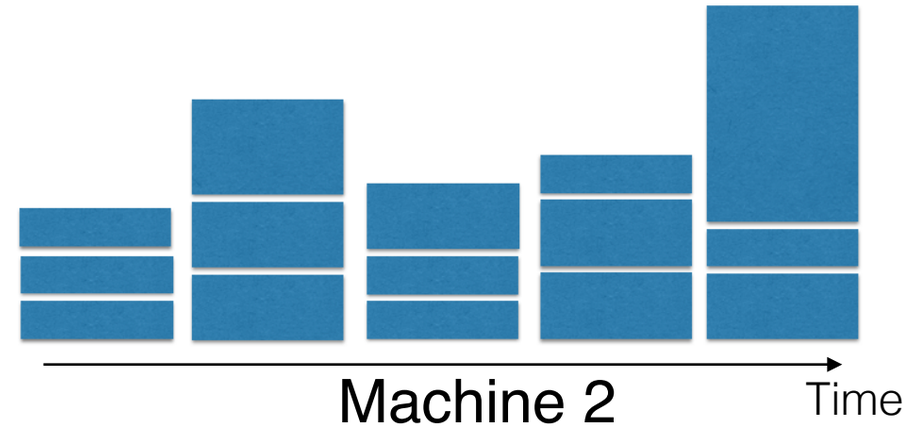
Summed CPU Usage over time

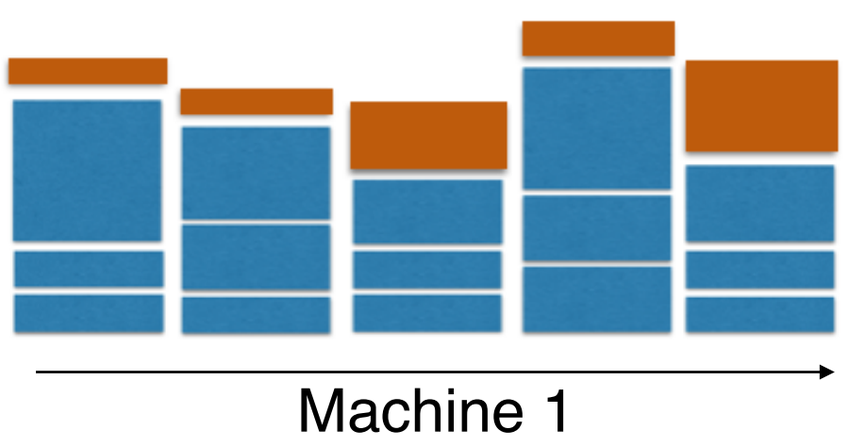




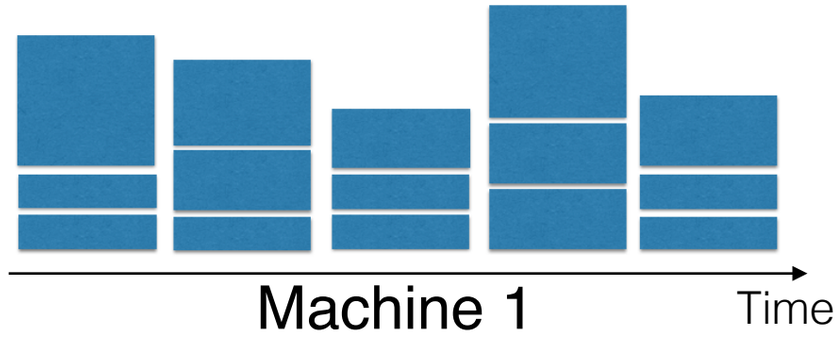
Summed CPU Usage over time

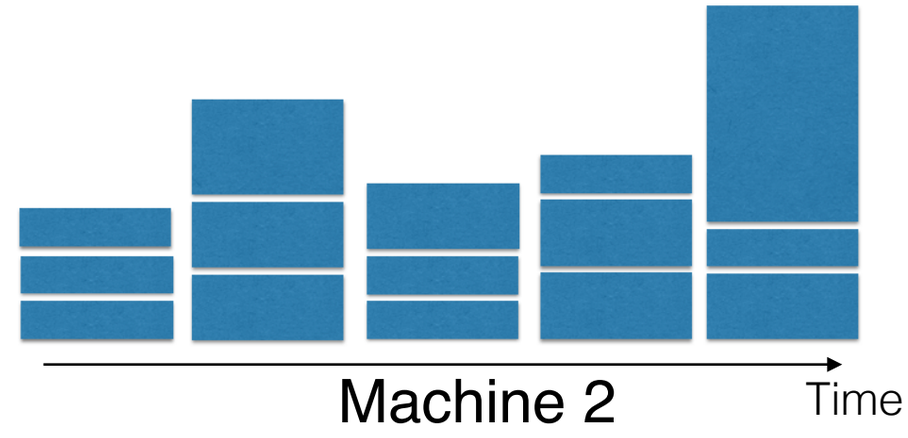


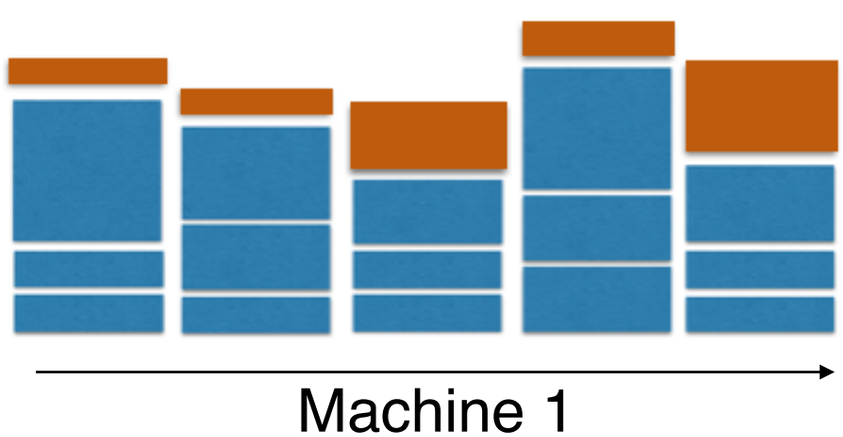


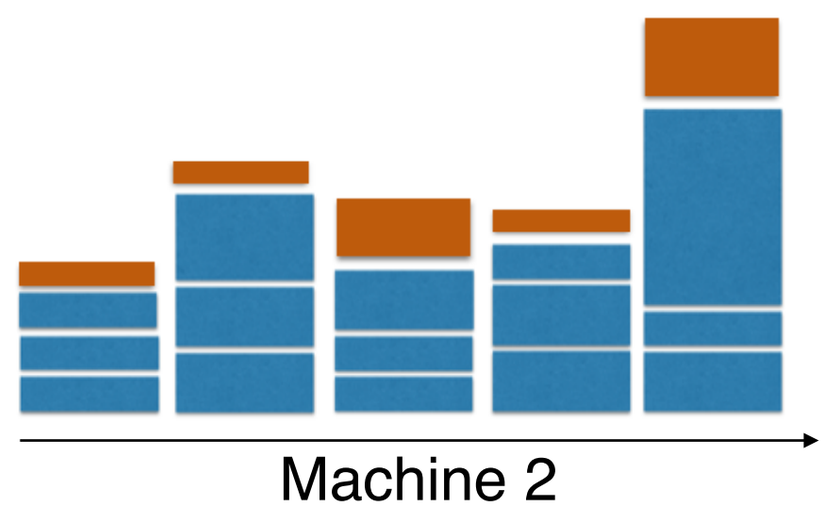


Summed CPU Usage over time

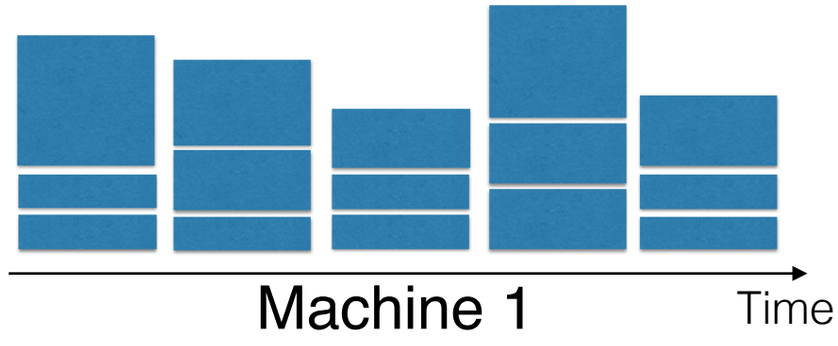


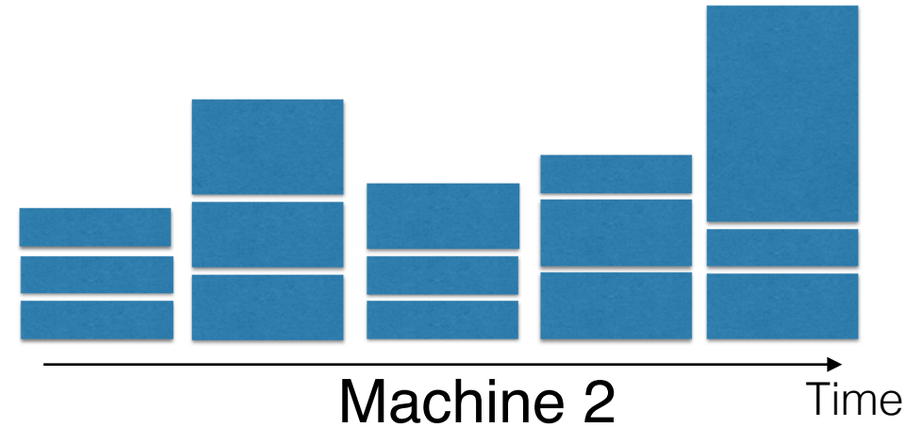


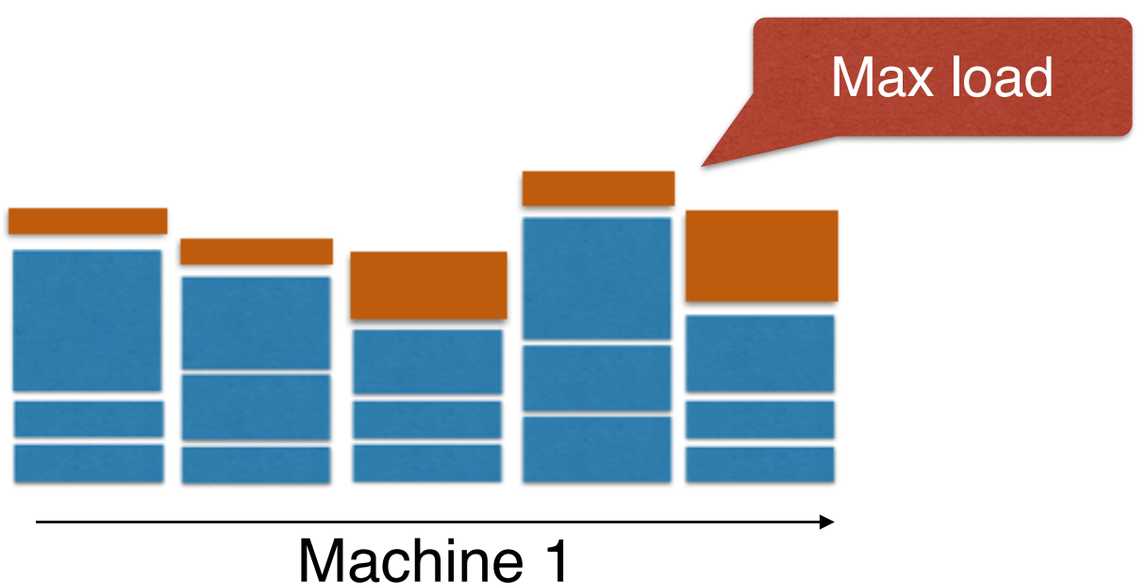


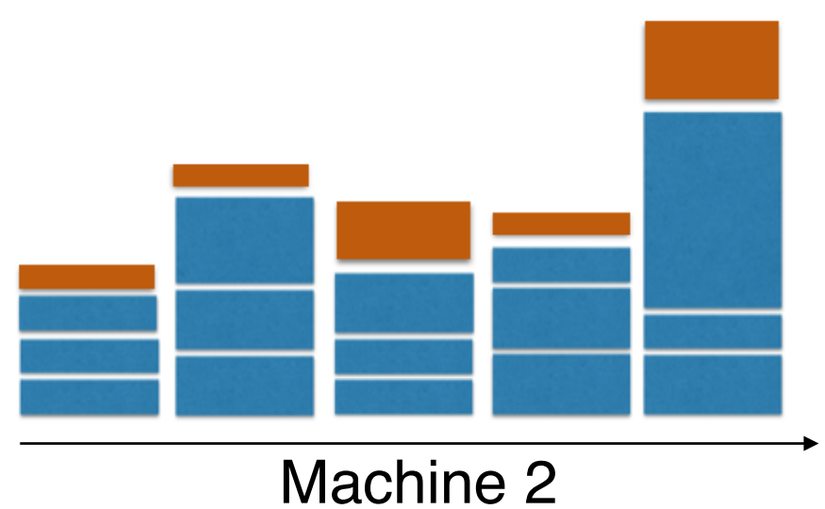


Summed CPU Usage over time

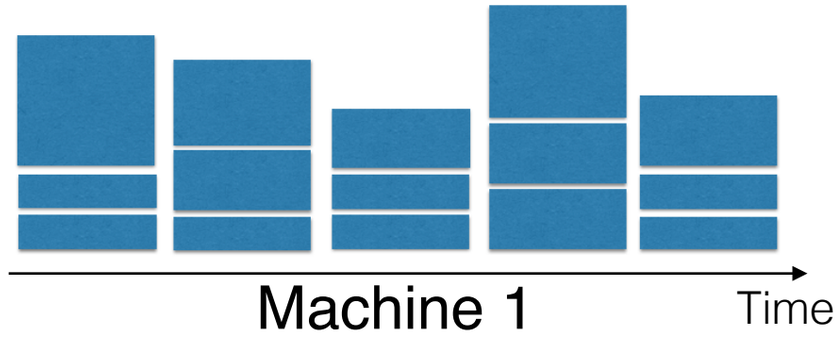


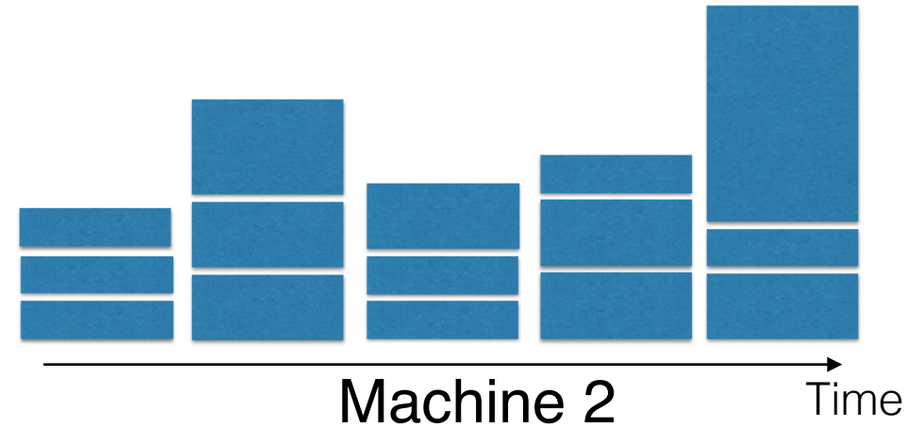


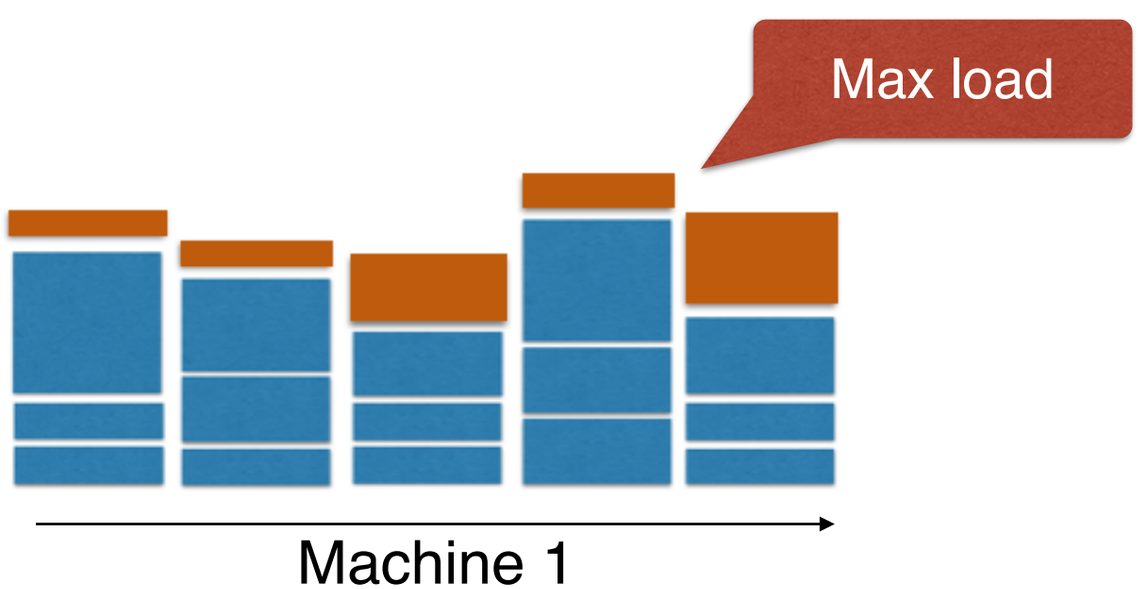


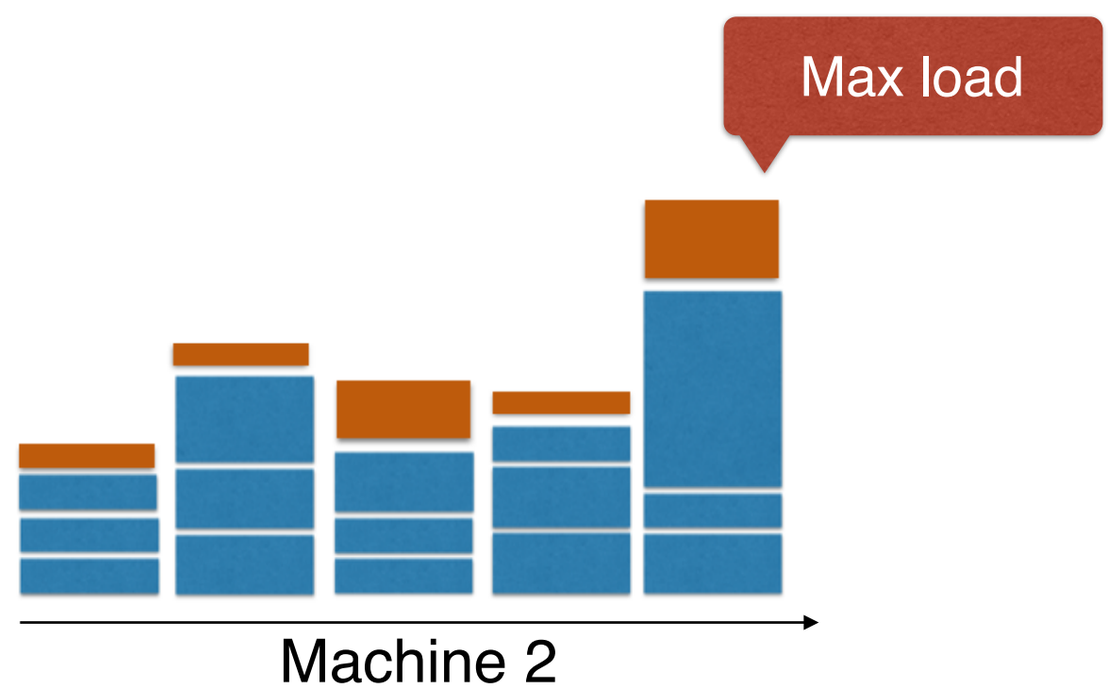


Summed CPU Usage over time

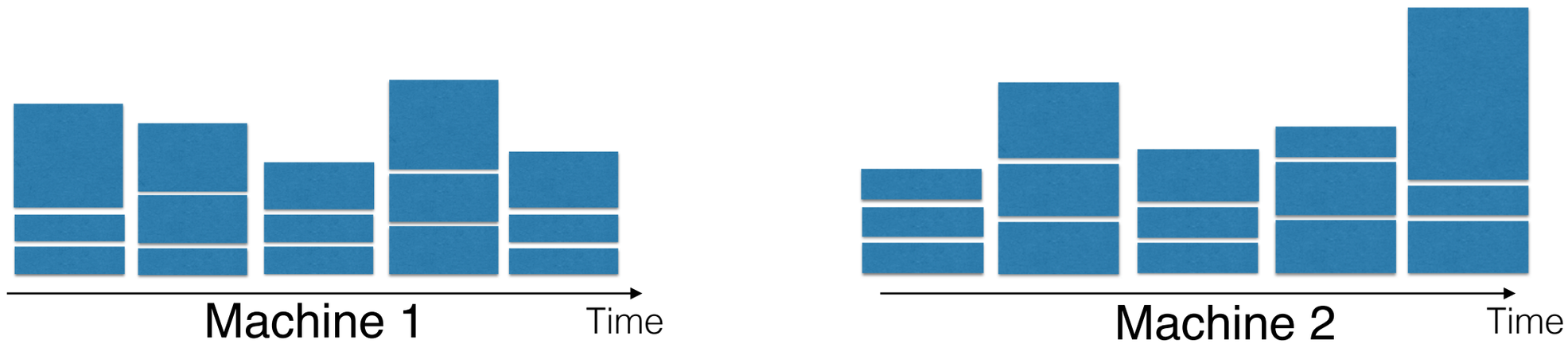






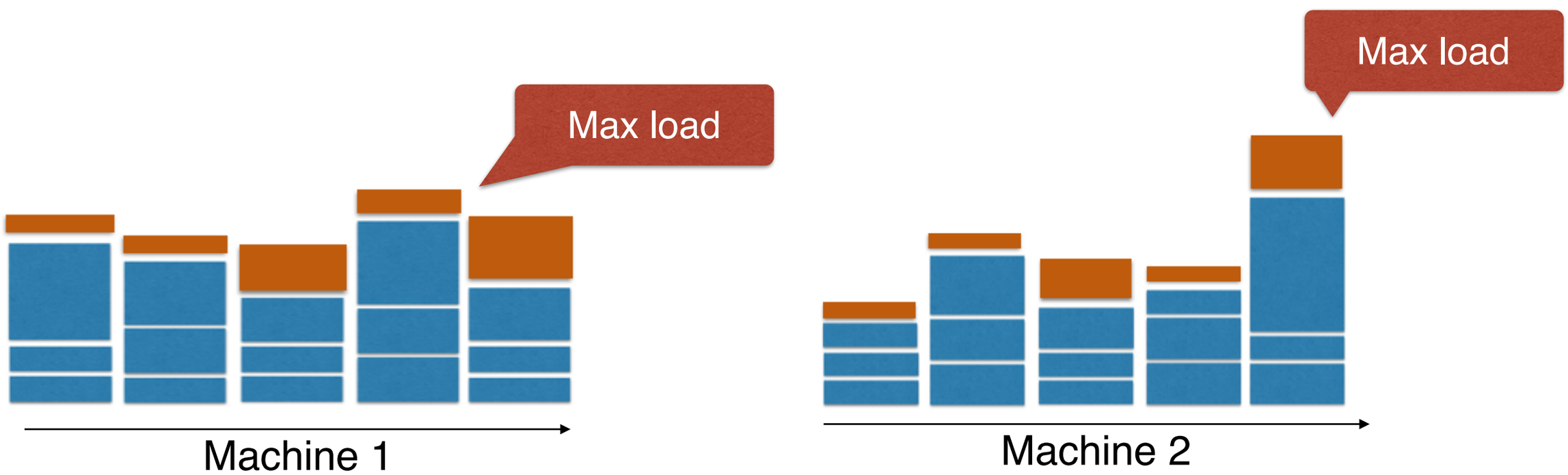


Summed CPU Usage over time



Minimize max load, pick machine 1

Tenant to place



CPU usage Time series

CPU usage Time series

↓

FFT Coefficients

CPU usage Time series

↓

FFT Coefficients

Why? Time series modeled with few coefficients

FFT Covariance Cost Model

FFT Covariance Cost Model

- Pairwise covariance of FFT coefficients - distance
  - Anti-correlated, less covariance
  - Hierarchy created, anti-correlated are closer

FFT Covariance Cost Model

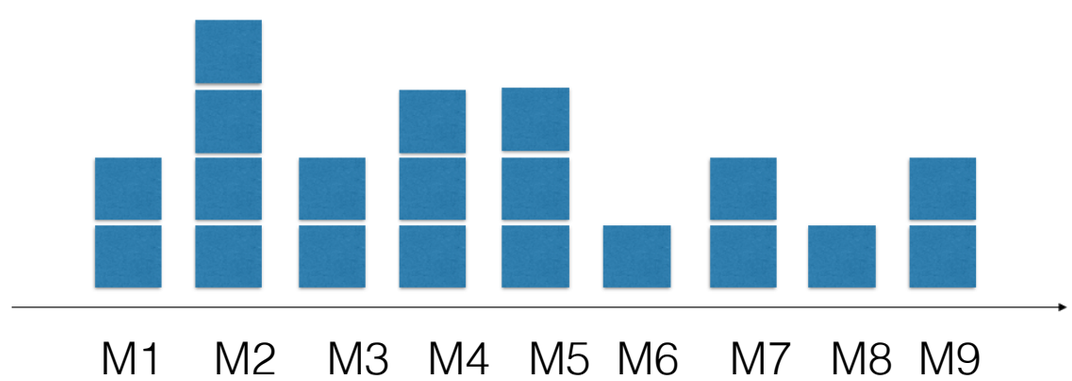
- Pairwise covariance of FFT coefficients - distance
  - Anti-correlated, less covariance
  - Hierarchy created, anti-correlated are closer
- Agglomerative hierarchical clustering with complete linkage
  - One cluster per machine
  - Cluster size: num**DB**/num**Machines**

Workloads change over time and databases are constantly being created and dropped due to customer churn

Dynamic Allocations

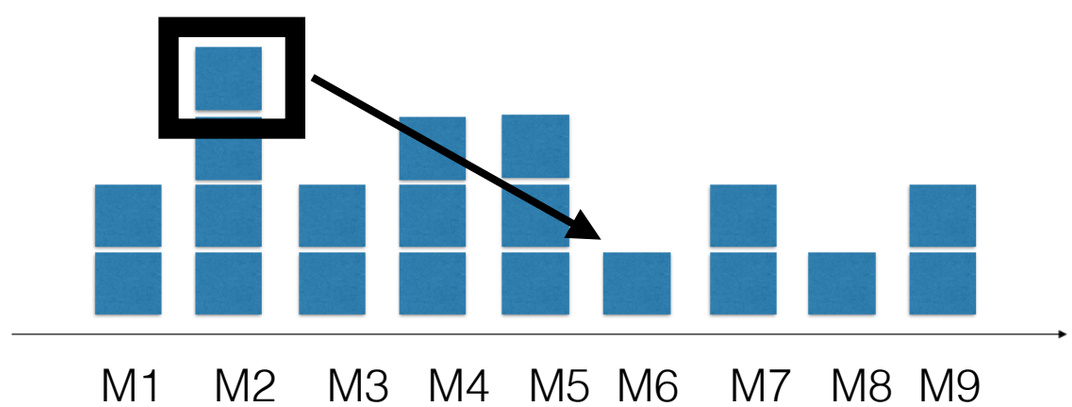
Dynamic Allocations

- Monitor machines for overload
- Balance load by swapping primary replicas, migrating to other machines - to least loaded machine
- Upper cap on the number of swapping



Dynamic Allocations

- Monitor machines for overload
- Balance load by swapping primary replicas, migrating to other machines - to least loaded machine
- Upper cap on the number of swapping

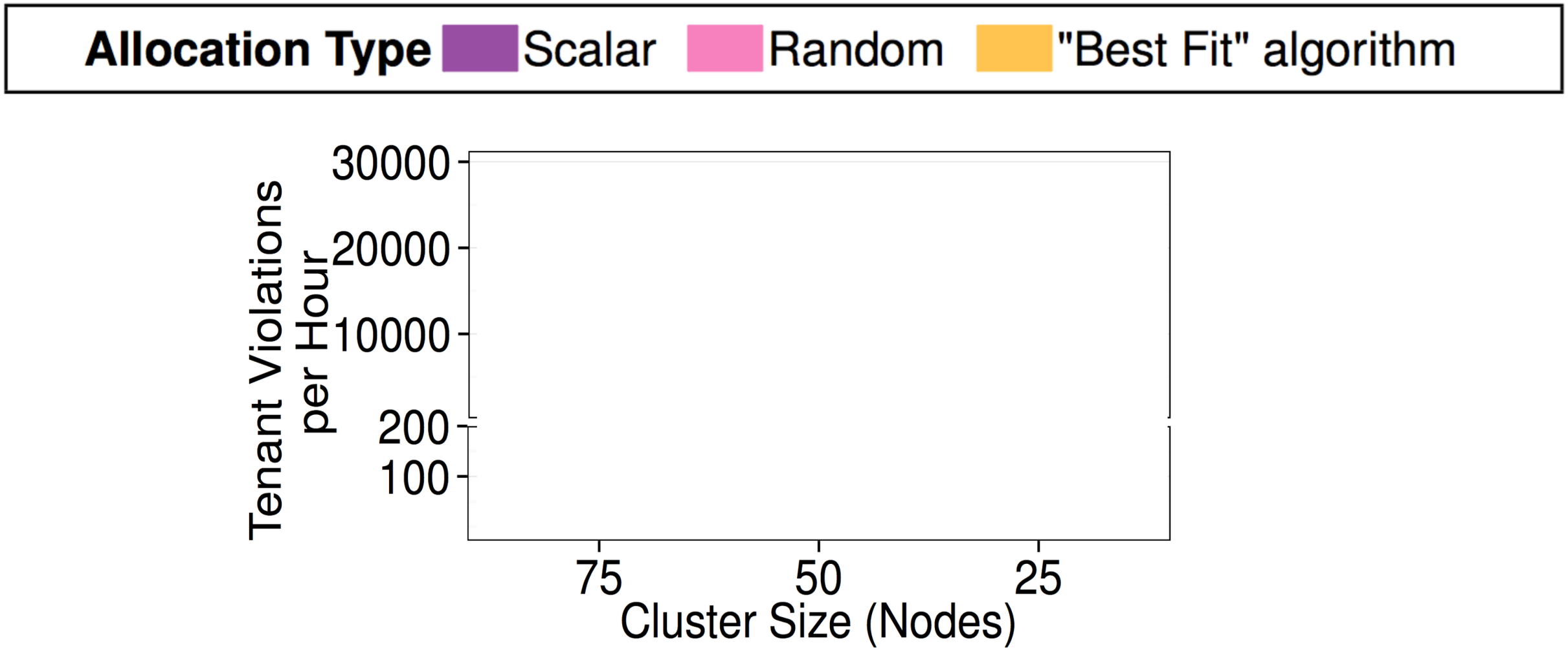


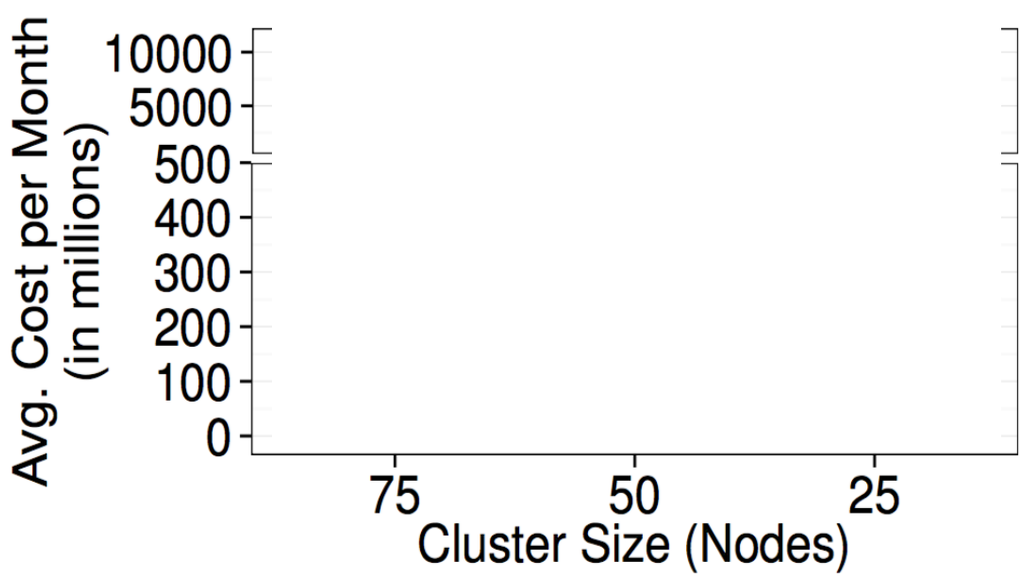
Evaluation

Experimental Set up

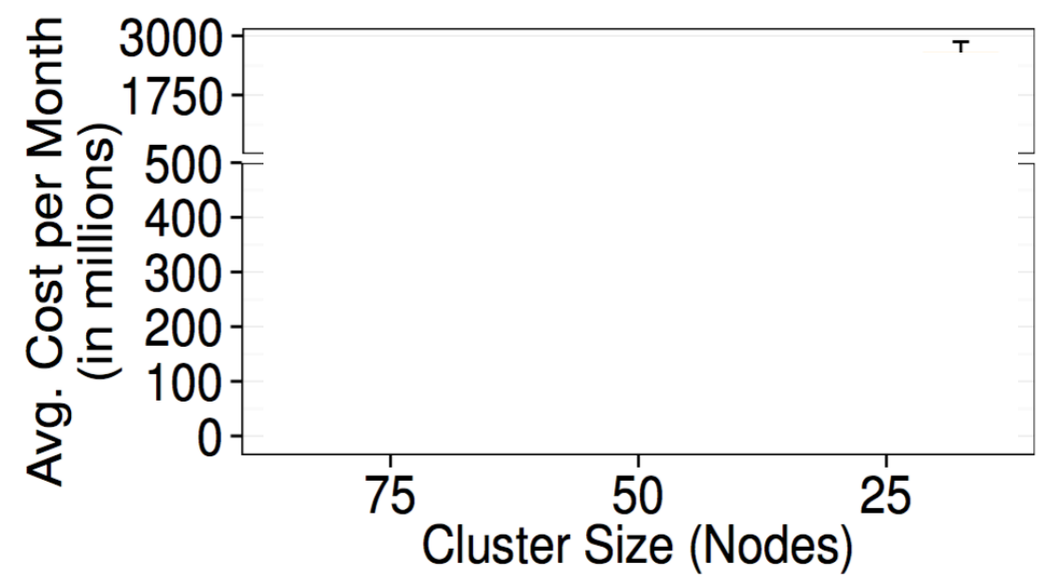
Experimental Set up

- North American cluster of 151 nodes, run on 1/2, 1/3 and 1/6th of the nodes
- Compare violations, PMAX cost and MSFT cost
- Scale factors S_p and S_m for the cost models is calculated as 30 and 46656 respectively
- Train model on 2 months, test on 1 month, cross validation

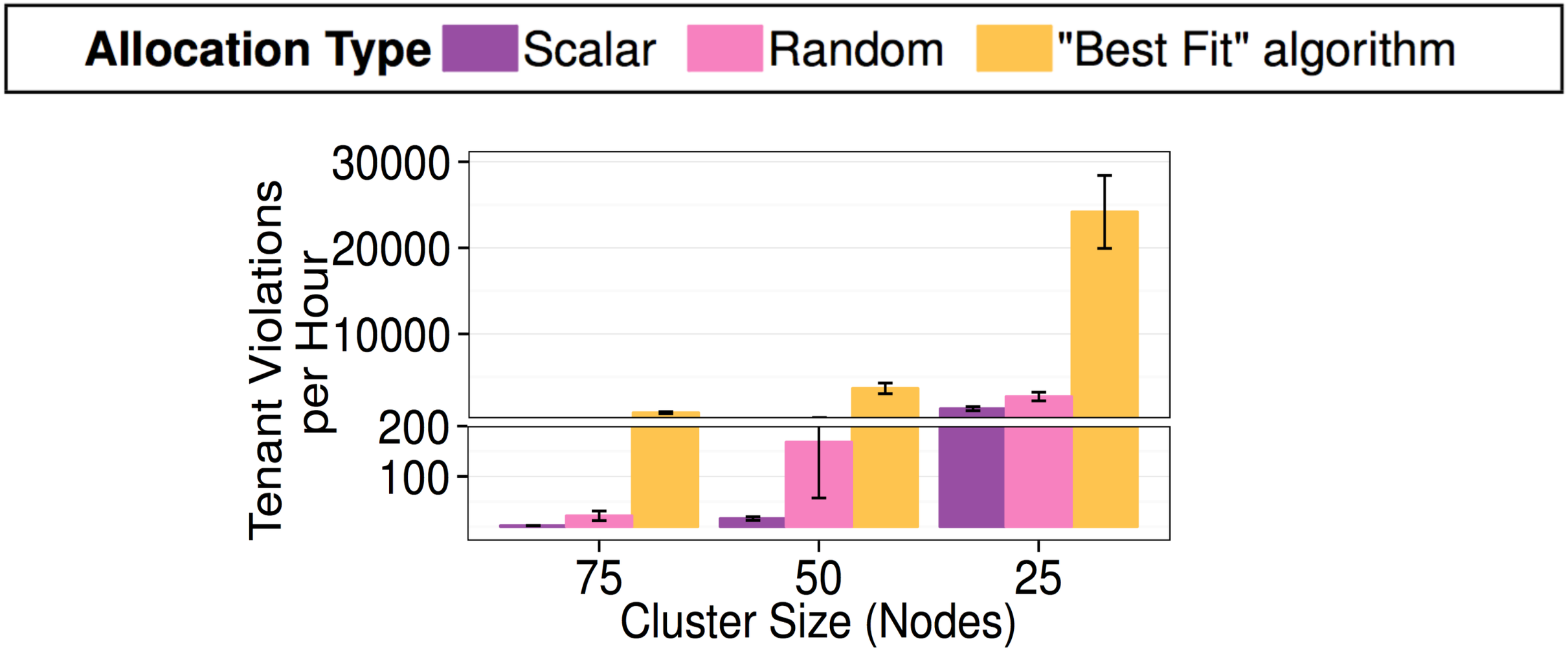


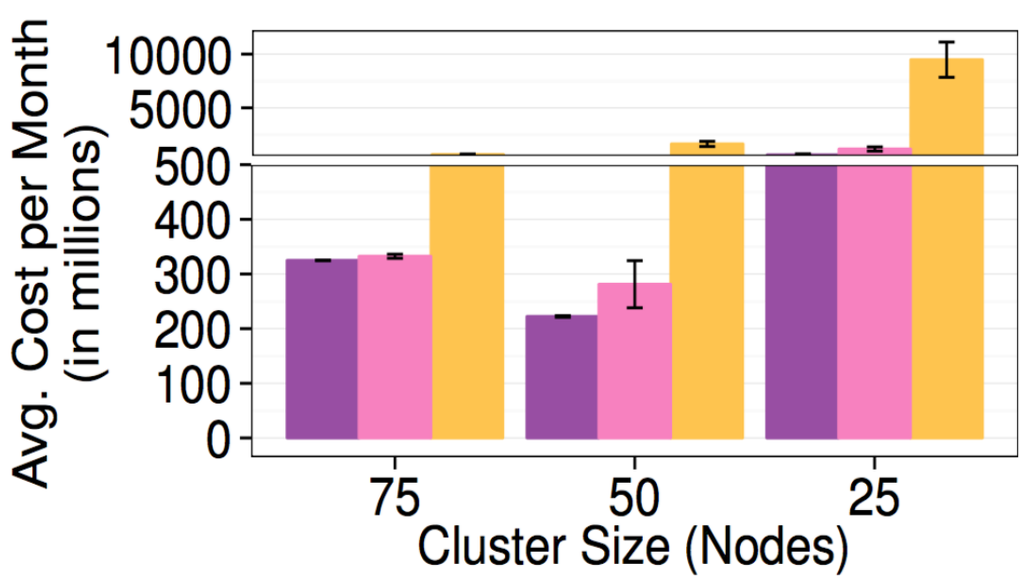


PMAX Cost

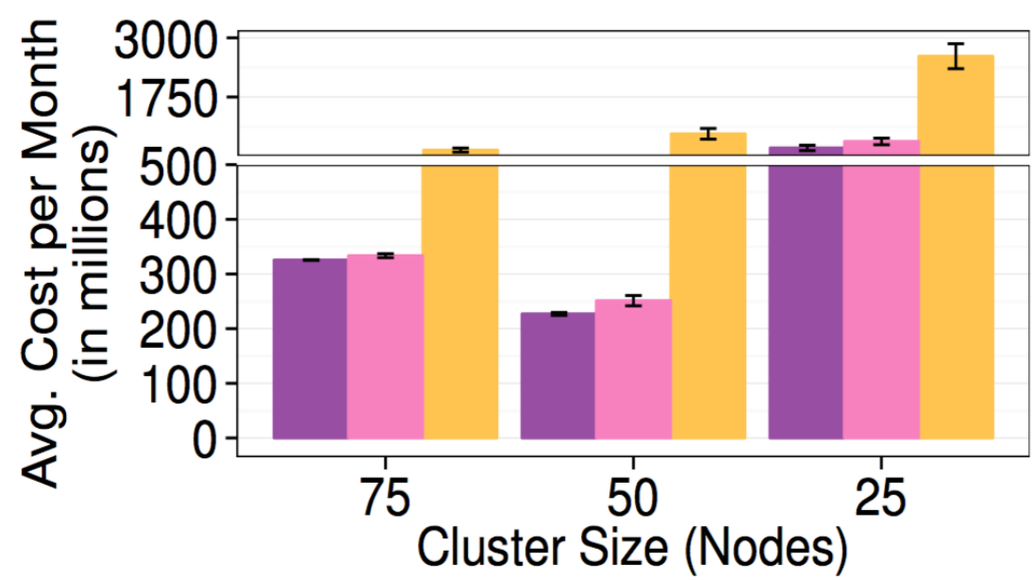


MSFT Cost

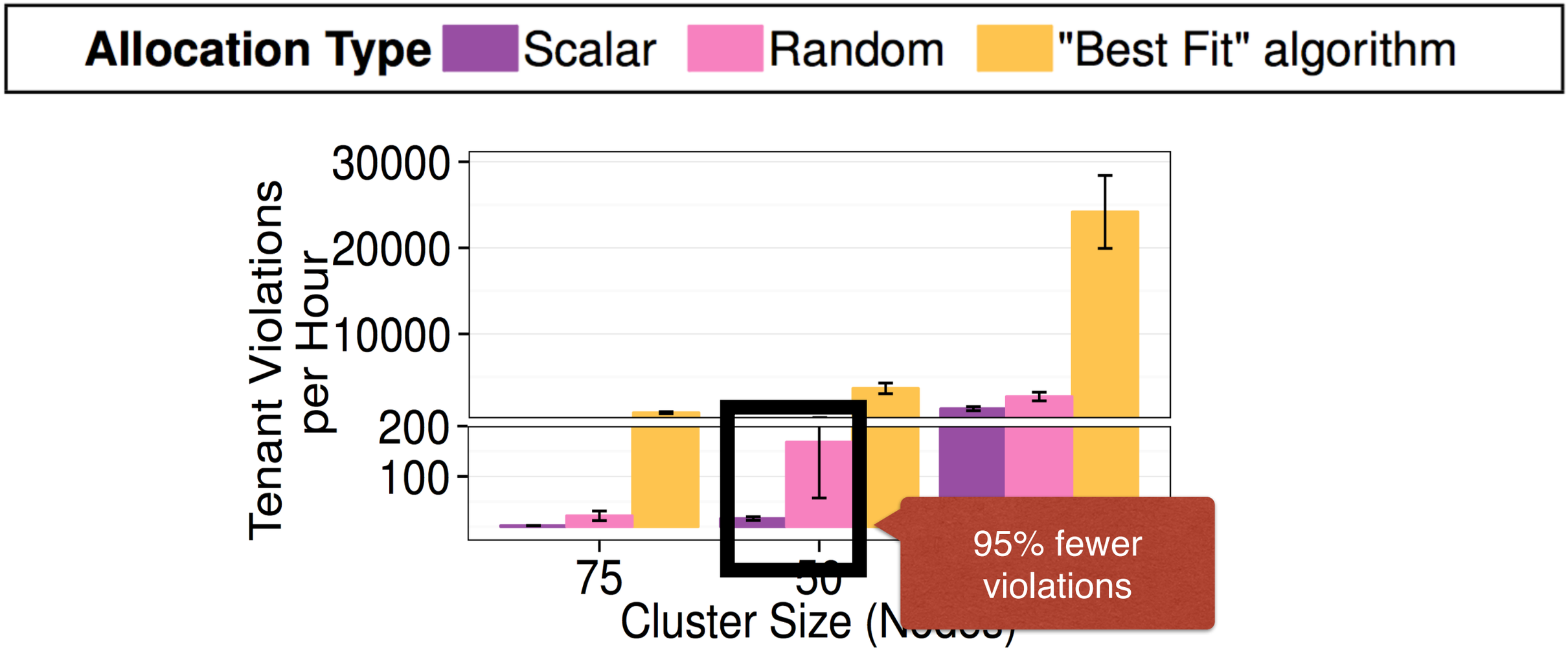


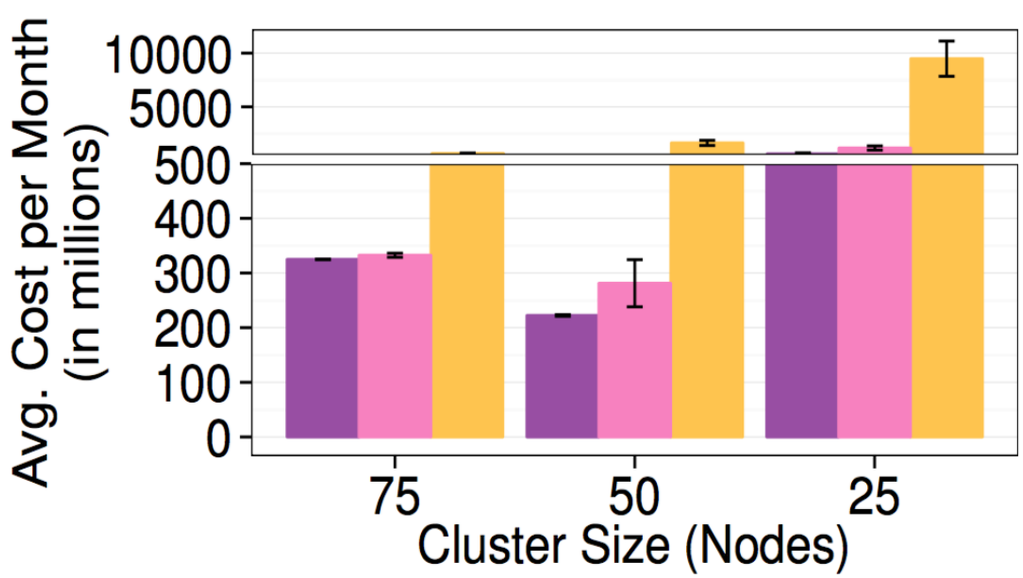


PMAX Cost

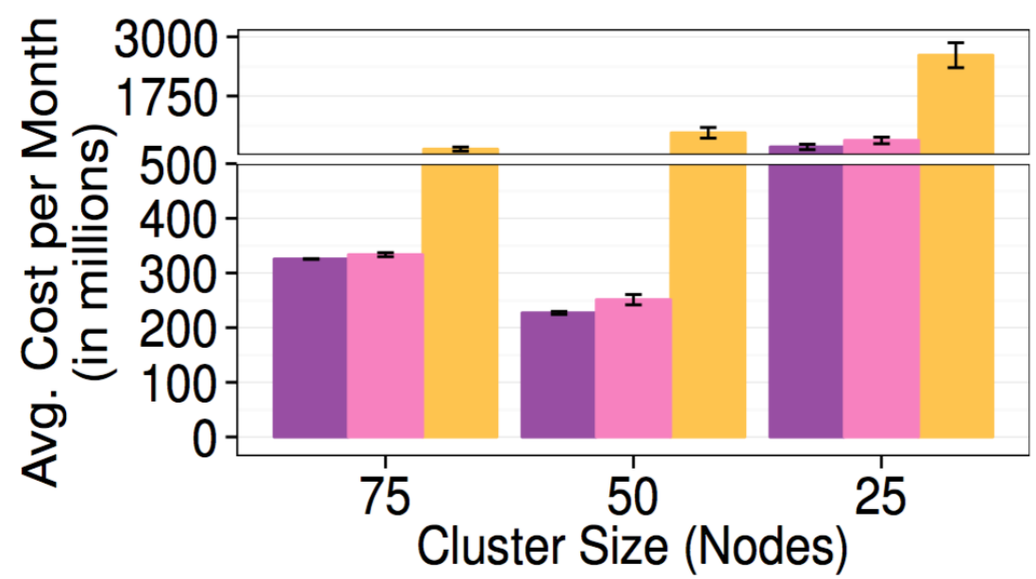


MSFT Cost

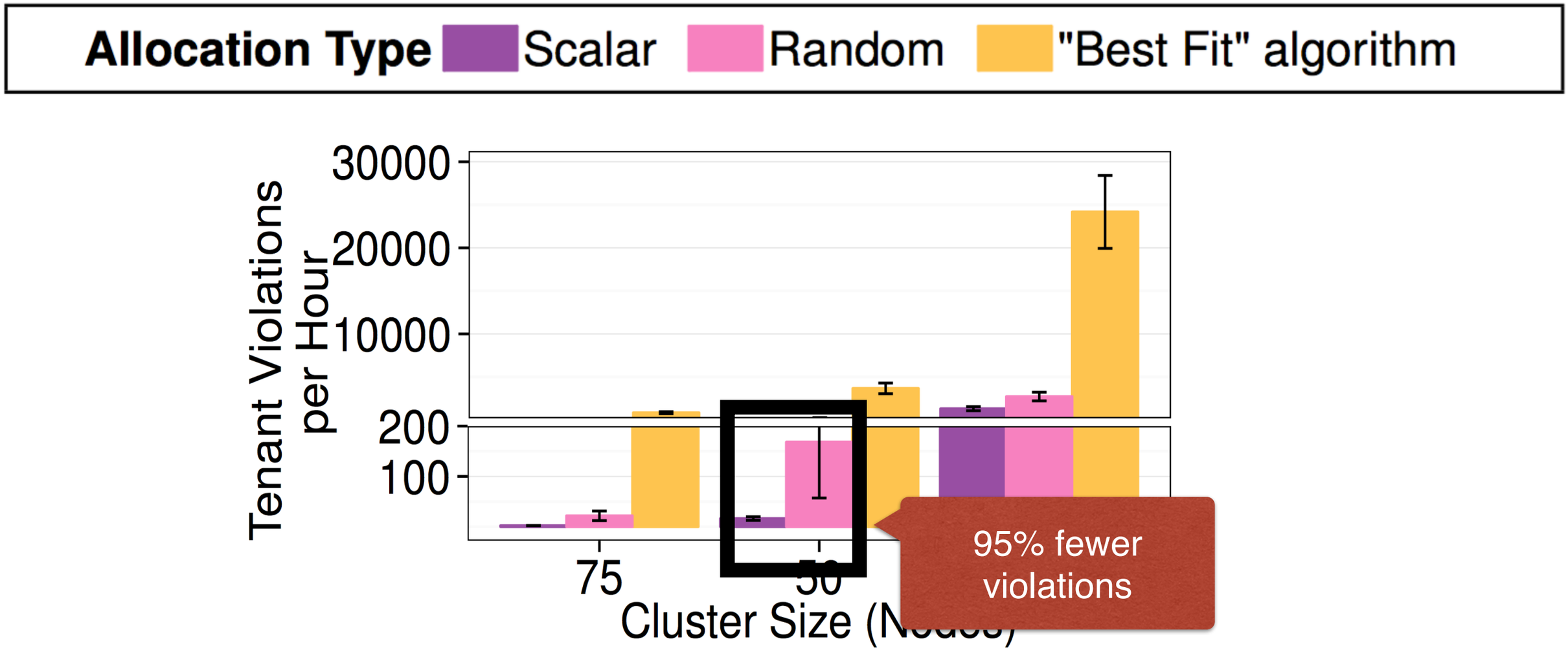


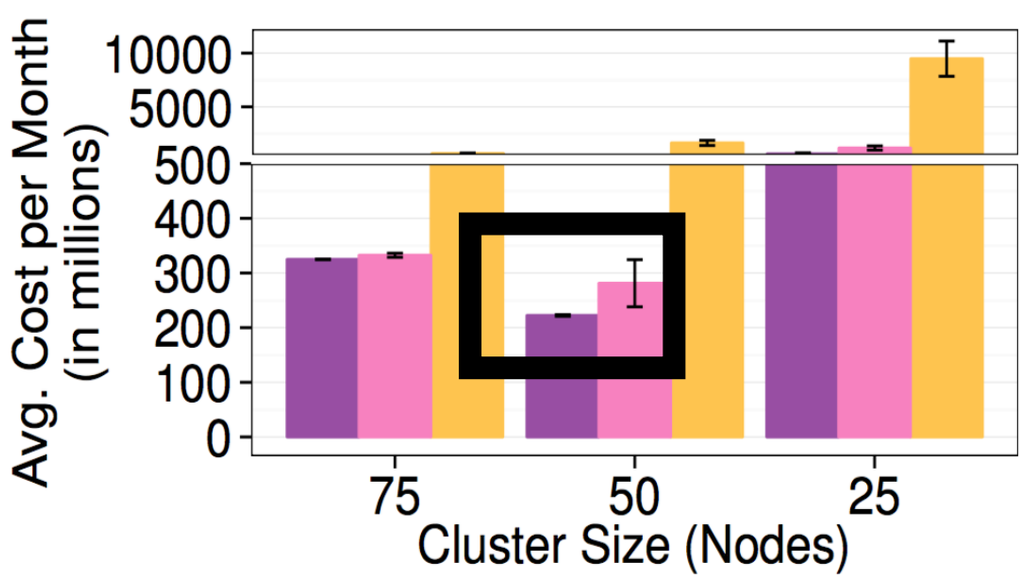


PMAX Cost

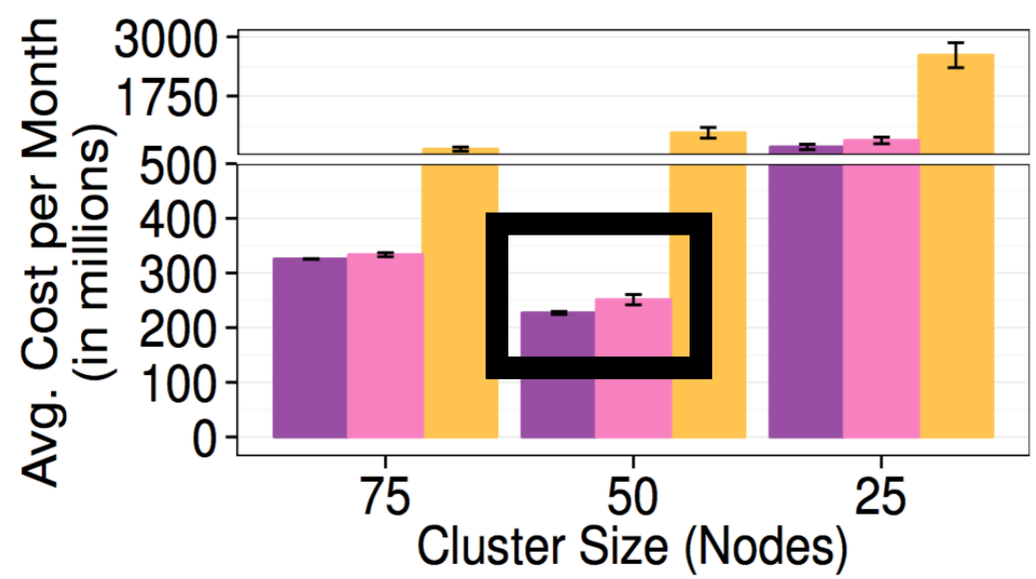


MSFT Cost

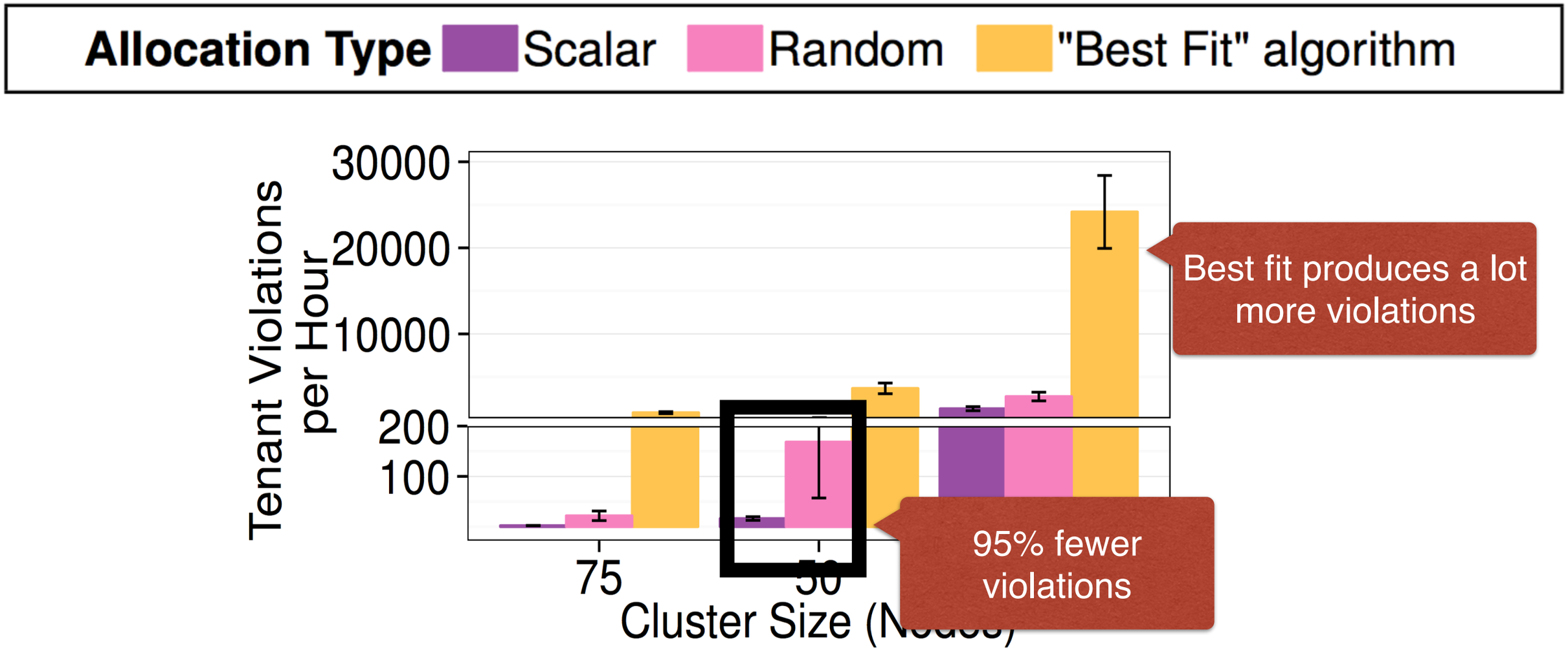


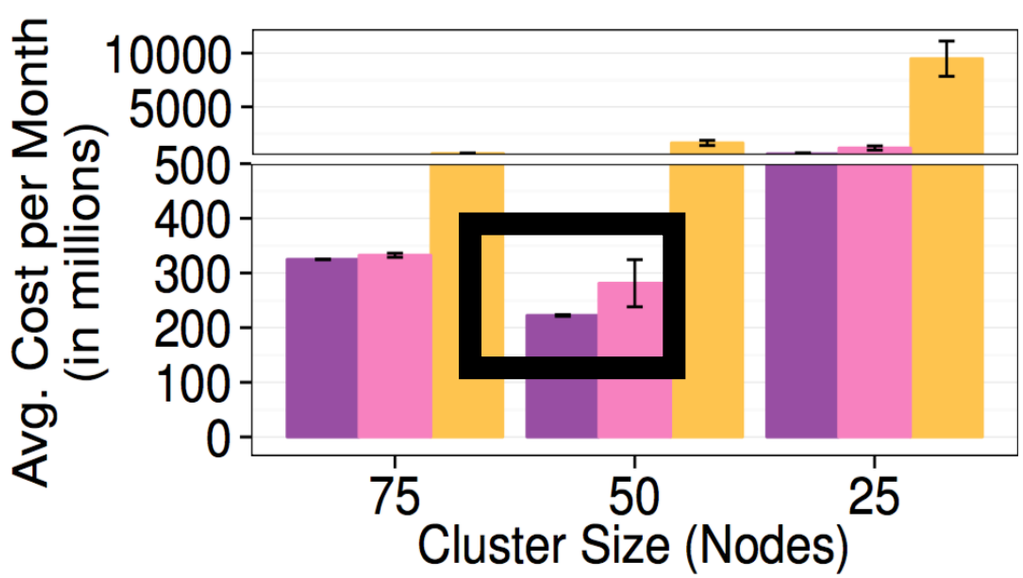


PMAX Cost

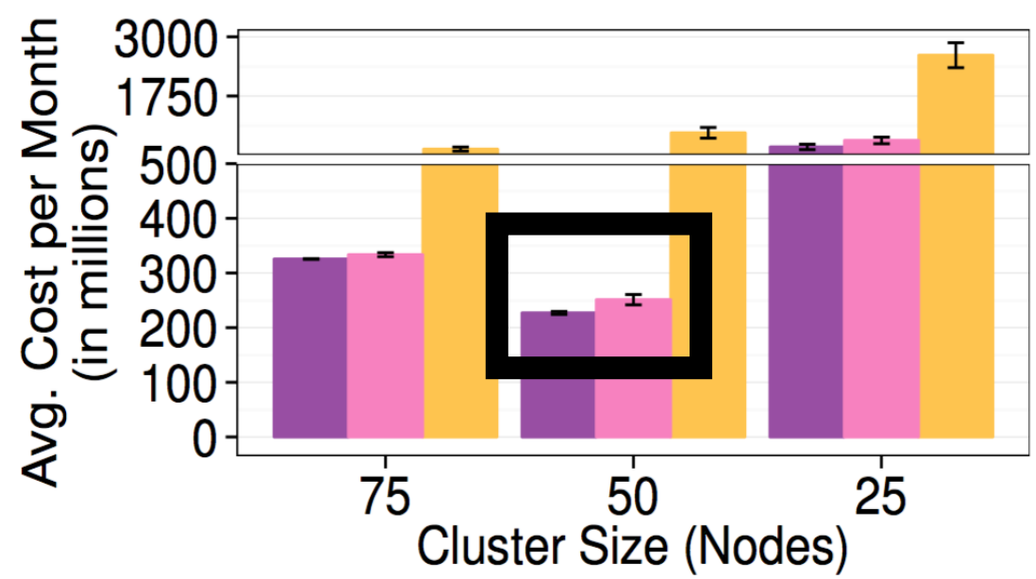


MSFT Cost

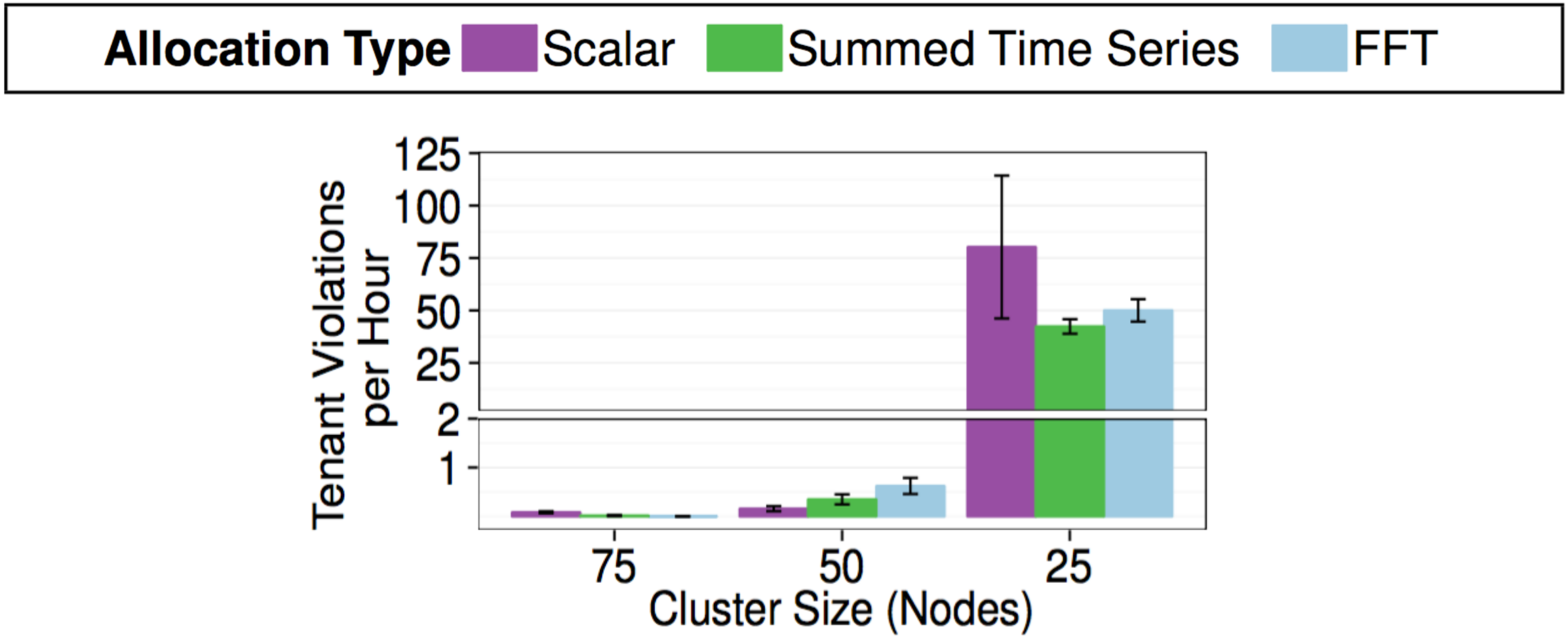


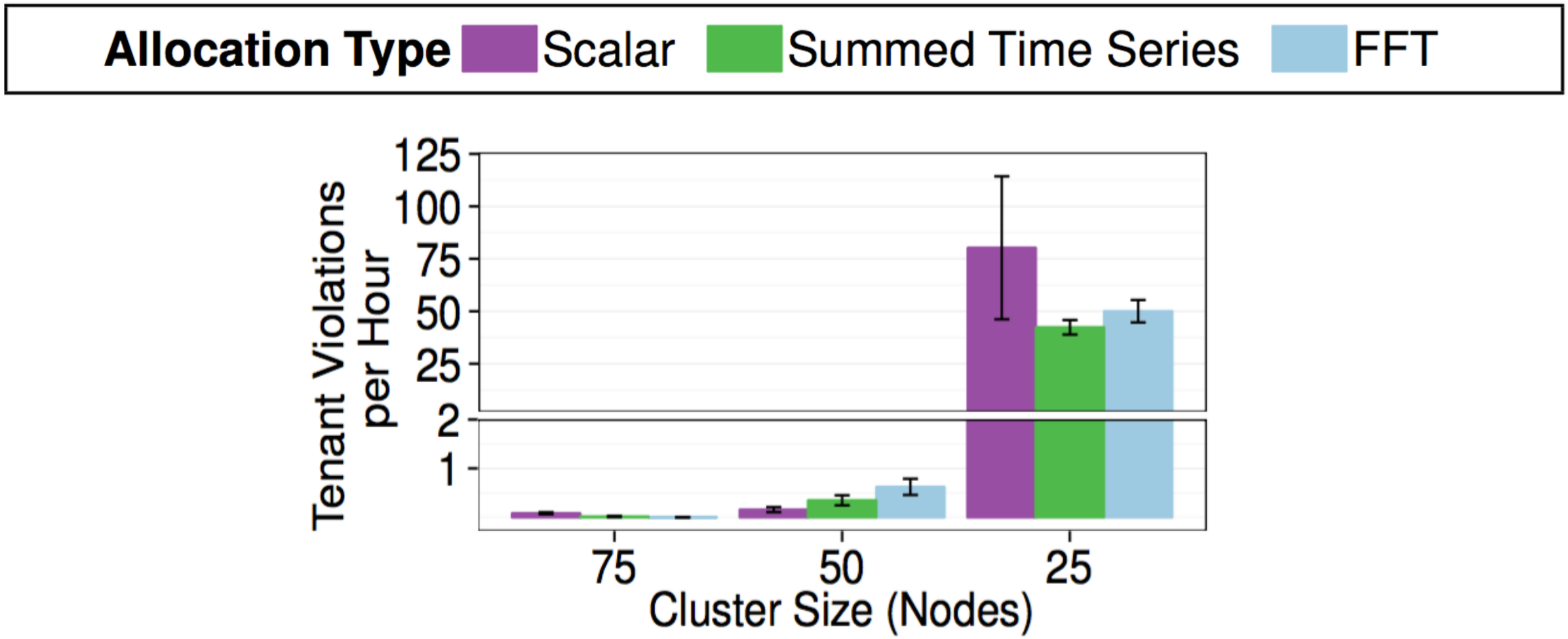


PMAX Cost

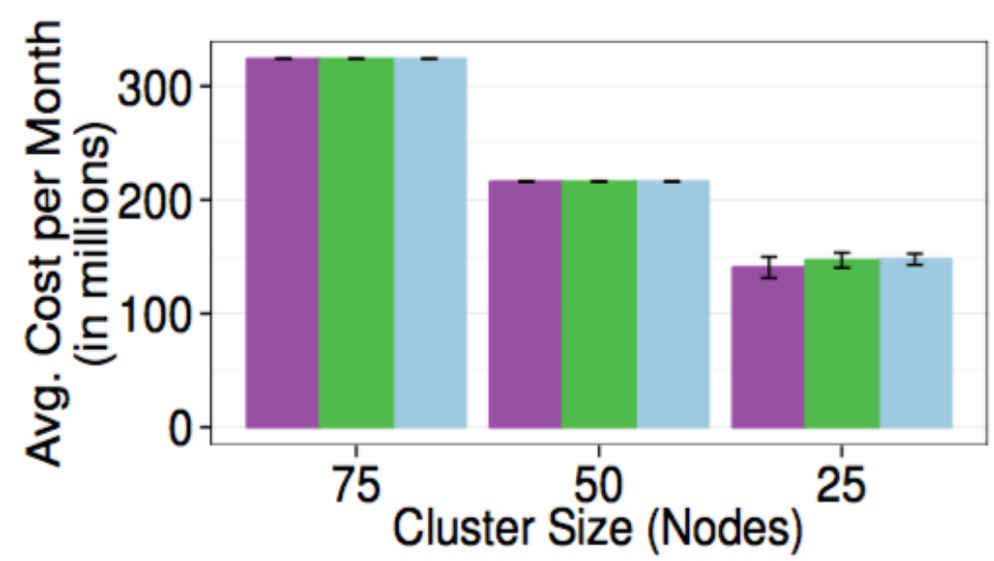


MSFT Cost

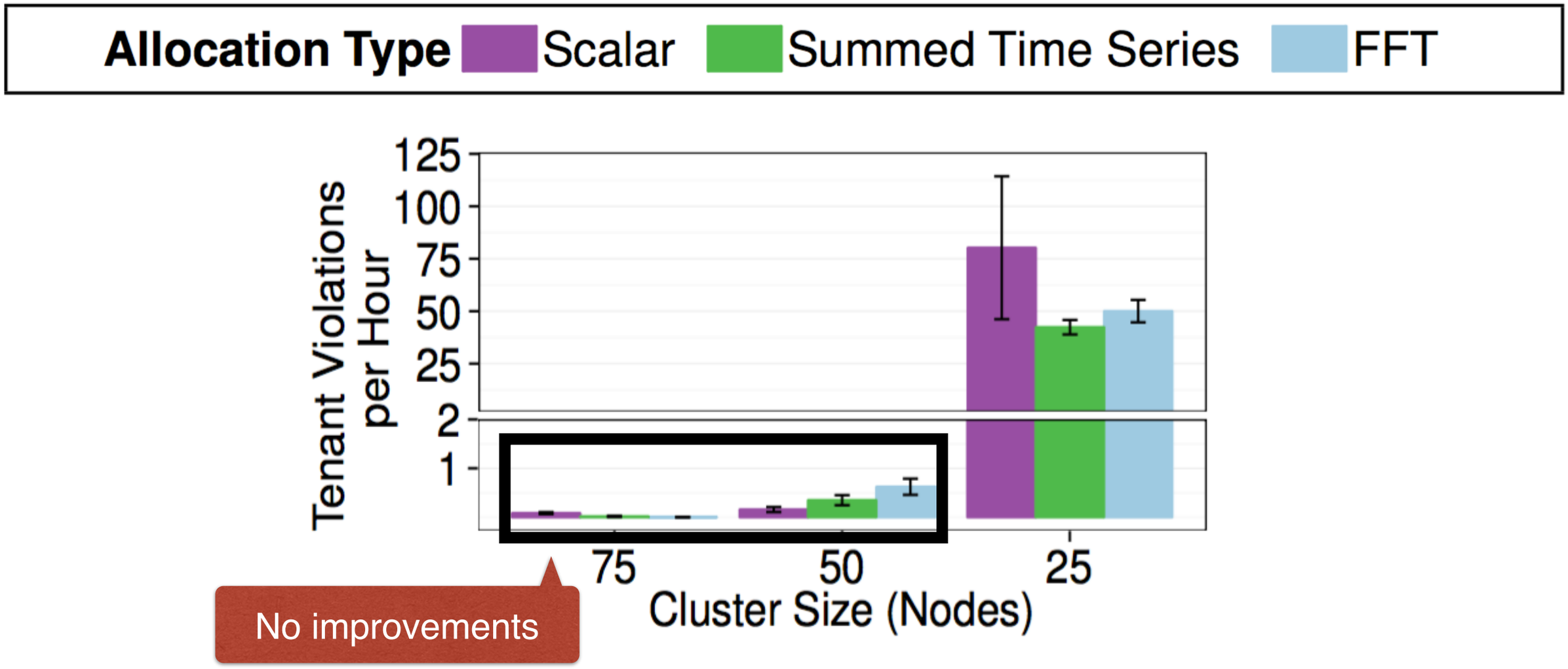

Allocation Type
Scalar
Summed Time Series
FFT
Tenant Violations per Hour
125
100
75
50
25
2
1
75
50
25
Cluster Size (Nodes)



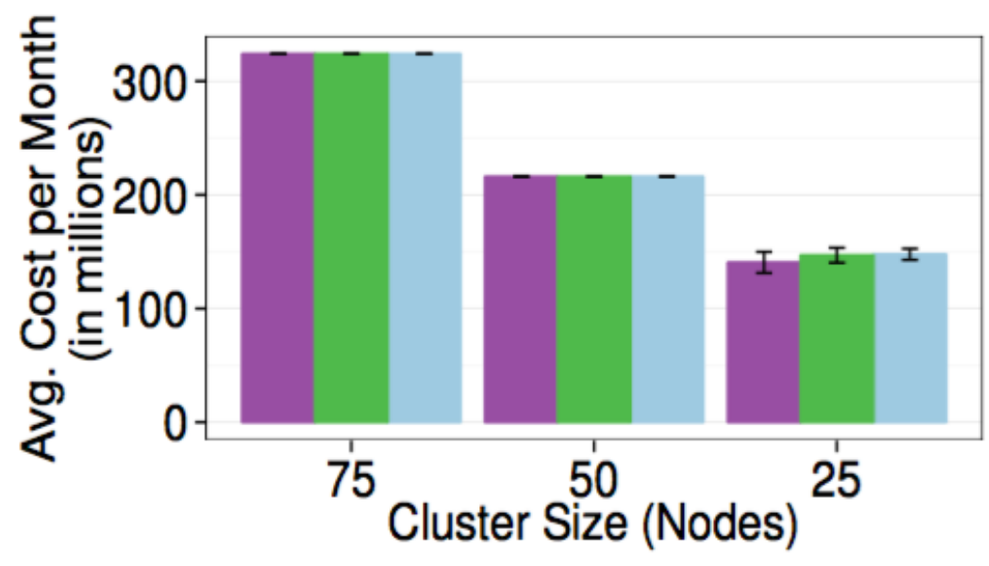
PMAX Cost



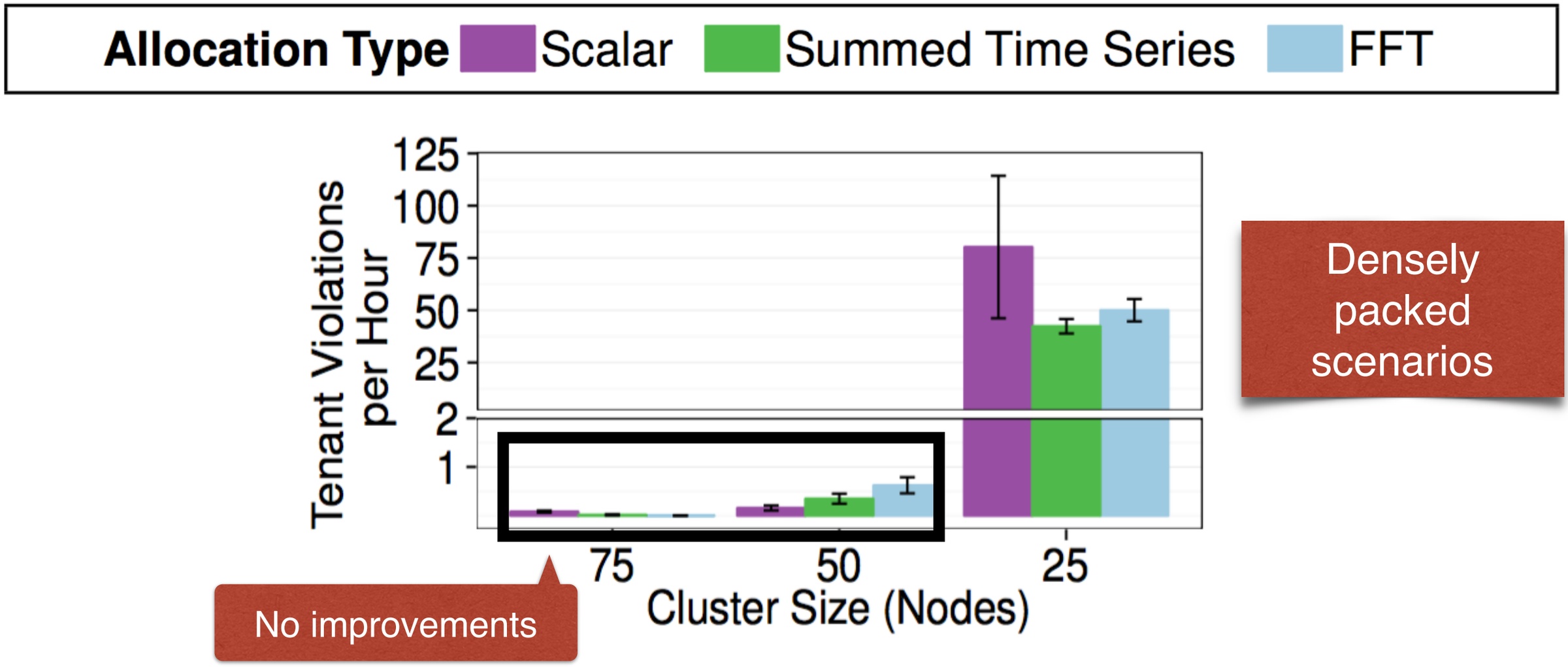
MSFT Cost



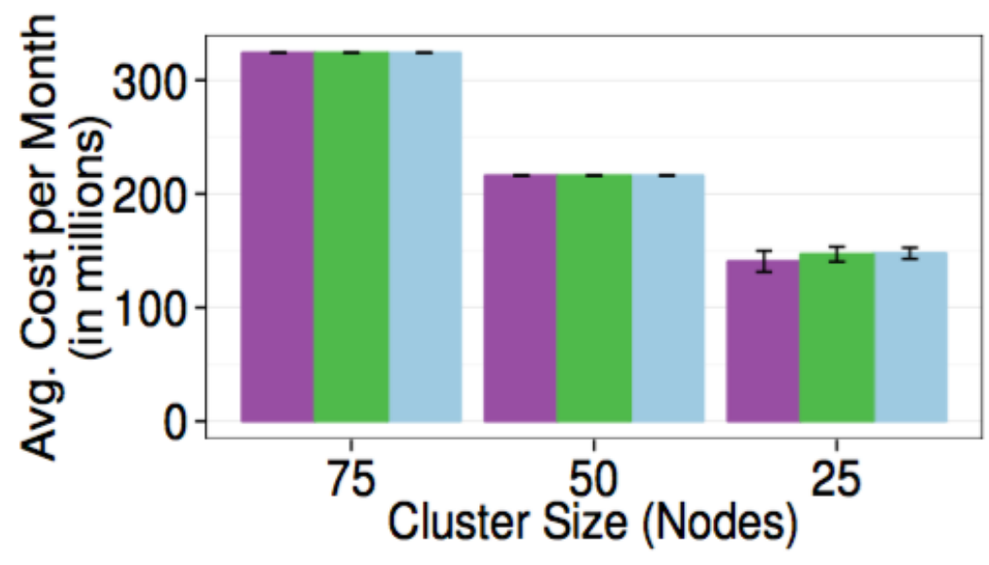
PMAX Cost



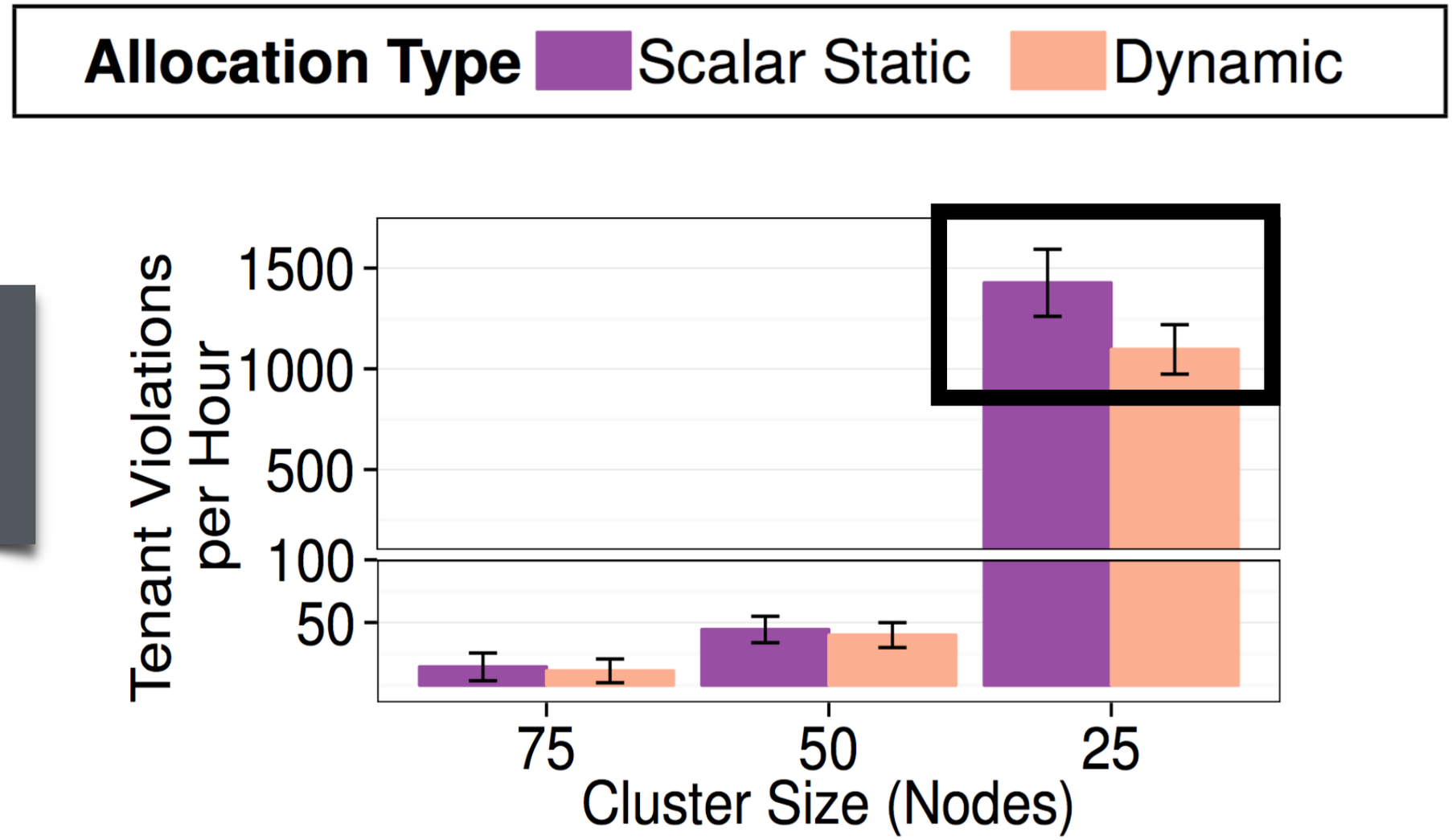
MSFT Cost



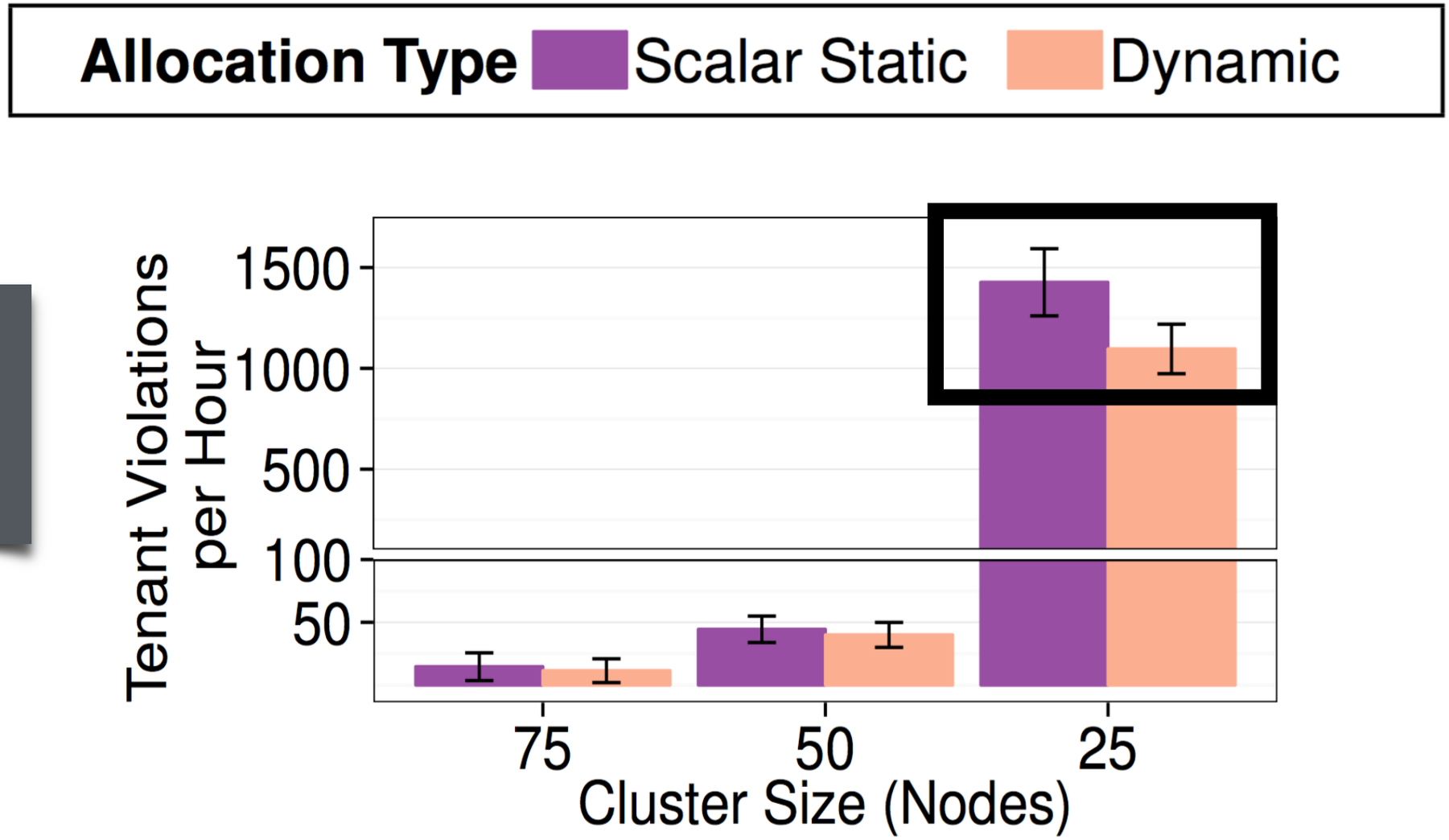
PMAX Cost



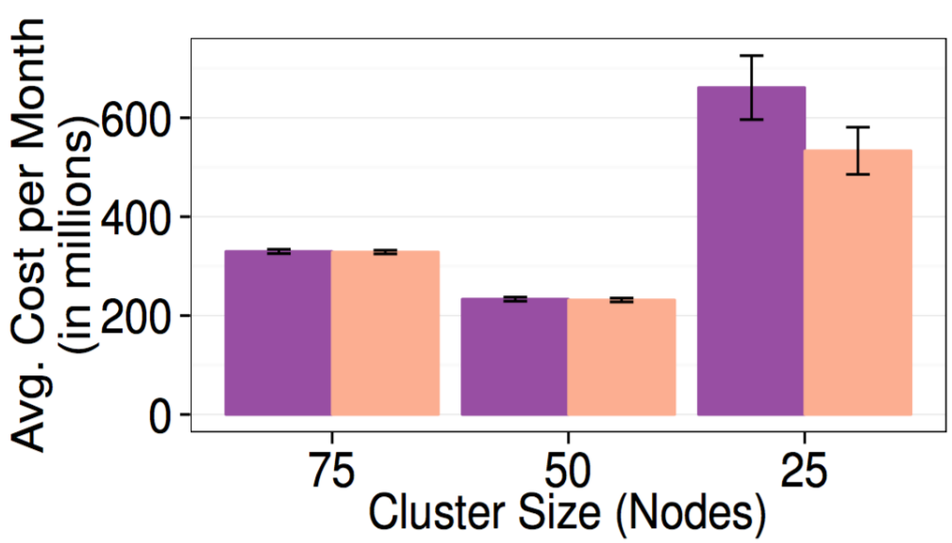
MSFT Cost



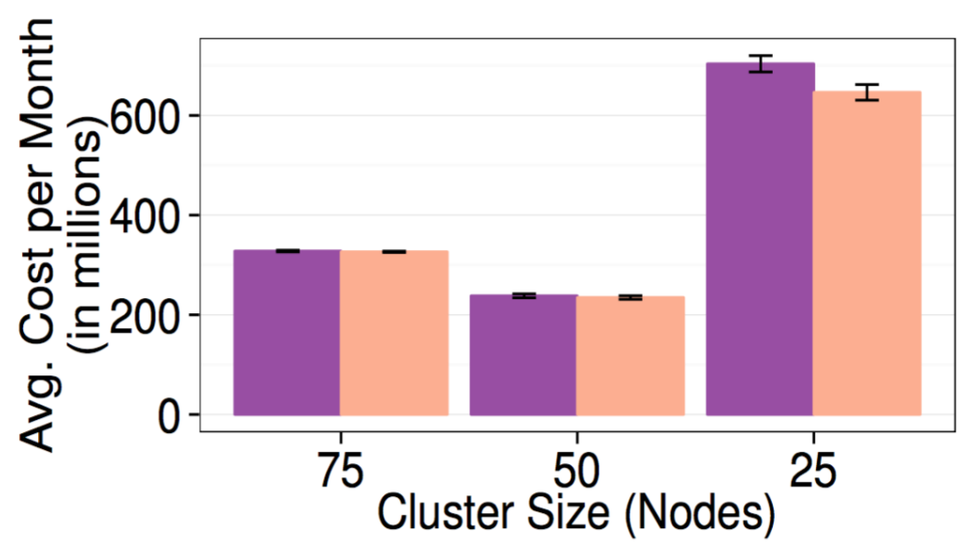
Every week, moved **at most** one



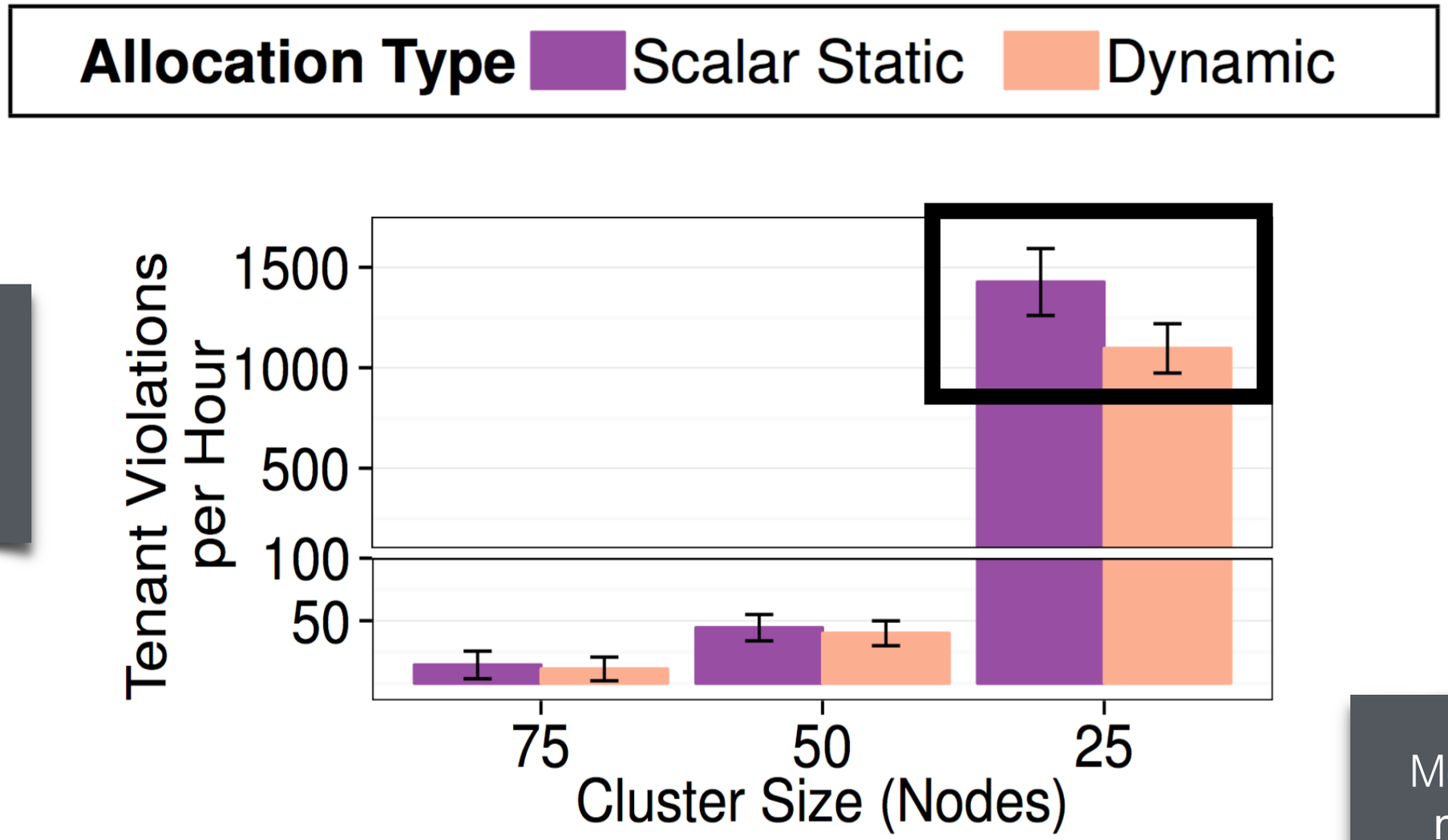
Every week, moved **at most** one



PMAX Cost

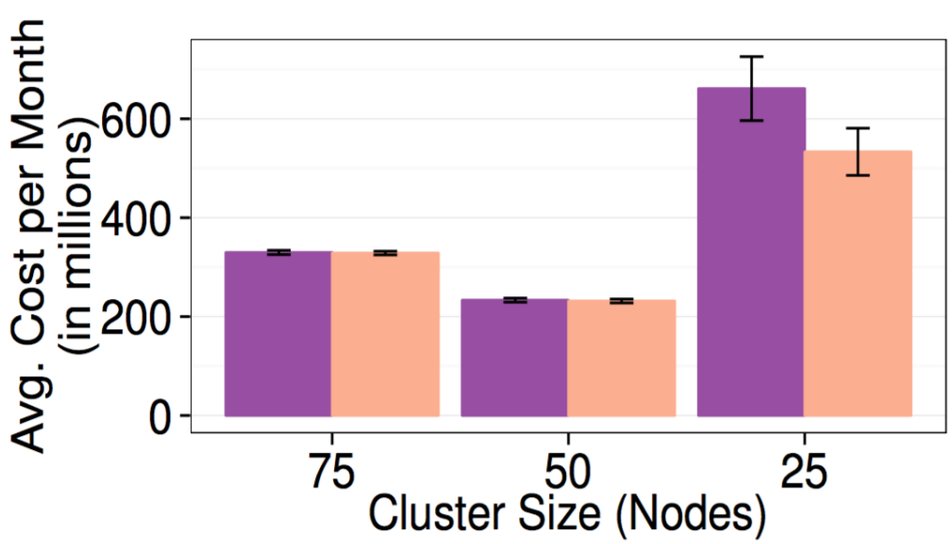


MSFT Cost

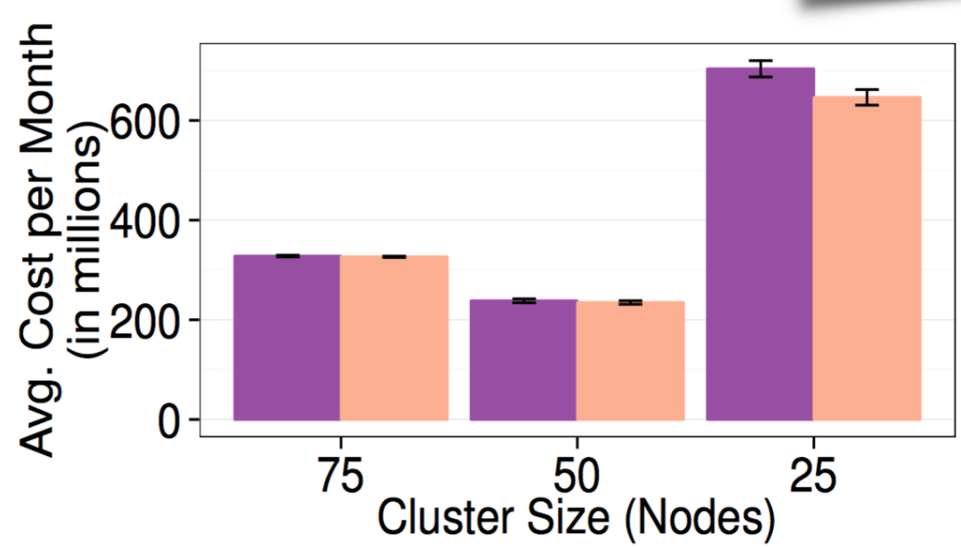


Every week, moved **at most** one

More **frequent** migrations?



PMAX Cost



MSFT Cost

Takeaways

- Publicly available dataset for effectiveness of placement algorithms
- Predictive greedy algorithms based on dataset
- Complex algorithms using time series of CPU usage
- Dynamic version enables tenant migration

Questions?