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

Single Tenancy

Multi Tenancy

DB 1

Tenant 1

Tenant 2

... 

Tenant N

DB 2

DB N

DB 1

Tenant 3

Tenant 4

... 

Tenant N

DB 2

DB N
Multi Tenancy

Single Tenancy

Multi Tenancy

Share Resources
$\$$
- Multi tenancy - economically viable
- One and a half million customer databases
- Resource utilization $\rightarrow$ cost
- Penalty for service outages $\rightarrow$ cost
• Multi tenancy - economically viable
• One and a half million customer databases
• Resource utilization → cost
• Penalty for service outages → cost

Minimize operation expenses + SLA violation penalties
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
Time series readings capture resource utilization of customer database + replicas

Summed utilization for a time window
Time series readings capture resource utilization of customer database + replicas

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

\[
\text{in\_violation}(m, t) = \left( \sum_{d \in D_{m,t}} \text{READ}_{d,t} \right) > P \times \text{MAX\_READ} \lor \\
\left( \sum_{d \in D_{m,t}} \text{WRITE}_{d,t} \right) > P \times \text{MAX\_WRITE} \lor \\
\left( \sum_{d \in D_{m,t}} \text{CPU}_{d,t} \right) > P \times \text{MAX\_CPU} \lor \\
\left( \sum_{d \in D_{m,t}} \text{LOG}_{d,t} \right) > P \times \text{MAX\_LOG}
\] (1)
Modelling violations

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

\[
in\_violation(m, t) = \left( \sum_{d \in D_{m,t}} \text{READ}_{d,t} \right) \cdot P \cdot \text{MAX\_READ} \lor \\
\left( \sum_{d \in D_{m,t}} \text{WRITE}_{d,t} \right) > P \cdot \text{MAX\_WRITE} \lor \\
\left( \sum_{d \in D_{m,t}} \text{CPU}_{d,t} \right) > P \cdot \text{MAX\_CPU} \lor \\
\left( \sum_{d \in D_{m,t}} \text{LOG}_{d,t} \right) > P \cdot \text{MAX\_LOG}
\]
SLA Penalties
SLA Penalties

MSFT

• Penalty for customers experiencing availability outage > 0.1%

• Penalty 2.5 times when outage > 1%

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

\[
v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (\text{in\_violation}(m_{d,t}, t) \times \text{is\_active}(d, t))
\]
SLA Penalties

MSFT

• Penalty for customers experiencing availability outage > 0.1%

• Penalty 2.5 times when outage > 1%

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

\[
v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (\text{in\_violation}(m_{d,t}, t) \times \text{is\_active}(d, t))
\]
SLA Penalties

**MSFT**

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

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

\[
v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (\text{in\_violation}(m_{d,t}, t) \times \text{is\_active}(d, t))
\]

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

MSFT

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

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

\[ v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (\text{in\_violation}(m_{d,t}, t) \times \text{is\_active}(d, t)) \]
SLA Penalties

MSFT

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

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

\[ v_{d,q} = \frac{1}{N_q} \sum_{t=1}^{N_q} (\text{in\textunderscore violation}(m_{d,t}, t) \times \text{is\textunderscore active}(d, t)) \]

PMAX

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

\[ \text{cost} = y \times G + \frac{S_P}{Q} \sum_{t=1}^{N} \sum_{m=1}^{y} (\text{in\textunderscore violation}(m, t) \times \sum_{d \in D_{m,t}} \frac{w_d}{10}) \]
SLA Penalties

**MSFT**

- Penalty for customers experiencing availability outage > 0.1%

- Penalty **2.5 times** when outage > 1%

\[
\text{cost} = y \times G + \frac{SM}{Q} \sum_{q=1}^{Q} \sum_{d \in D} p(v_{d,q})
\]

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

**PMAX**

- Weigh penalties by customers expected level of service

- Violations for **high paying** customers is **costly**

\[
\text{cost} = y \times G + \frac{Sp}{Q} \sum_{t=1}^{N} \sum_{m=1}^{y} (in\_violation(m,t) \times \sum_{d \in D_{m,t}} \frac{wd}{10})
\]

violations*service_level

violations*active_machines

violations*active_machines
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

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:

\[ \text{cost}_{d,z} = \text{base}_{d,z} + \lambda * a(w_d, z) \]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[
\text{cost}_{d,z} = \text{base}_{d,z} + \lambda \ast a(w_d, z)
\]

\[
\text{base}_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (\text{CPU}_{d,t})
\]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[
\text{cost}_{d,z} = \text{base}_{d,z} + \lambda \ast a(w_d, z)
\]

\[
\text{base}_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (\text{CPU}_{d,t})
\]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[ \text{cost}_{d,z} = \text{base}_{d,z} + \lambda \ast a(w_d, z) \]

\[ \text{base}_{d,z} = \frac{1}{N_d,z} \sum_{t=1}^{N_d,z} \text{CPU}_{d,t} \]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[ \text{cost}_{d,z} = \text{base}_{d,z} + \lambda \times a(w_d, z) \]

\[ \text{base}_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (\text{CPU}_{d,t}) \]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[
\text{cost}_{d,z} = \text{base}_{d,z} + \lambda \times a(w_d, z)
\]

\[
\text{base}_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (\text{CPU}_{d,t})
\]

\[
a(w_d, z) = \begin{cases} 
\text{avg}_{s \in D_z, w_s = w_d} (\text{base}_{s,z}) & \text{if } \exists s \in D_z \text{ s.t. } w_s = w_d \\
\text{avg}_{s \in D_z} (\text{base}_{s,z}) & \text{otherwise}
\end{cases}
\]
Greedy - Scalar Cost Model

Cost model based on average CPU usage:

\[
\text{cost}_{d,z} = \text{base}_{d,z} + \lambda \times a(w_d, z)
\]

\[
\text{base}_{d,z} = \frac{1}{N_{d,z}} \sum_{t=1}^{N_{d,z}} (CPU_{d,t})
\]

Additional cost: Avg of all DBs in same SLA level

\[
a(w_d, z) = \begin{cases} 
\text{avg} & \text{if } \exists s \in D_z \text{ s.t. } w_s = w_d \\
\text{avg} & \text{otherwise}
\end{cases}
\]
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:

\[ L_{m,z} = \max \left( \sum_{d \in D_m} \left( \text{cost}_{d,z_i} \right) \right) \]

• 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:

\[
L_{m,z} = \max_1^k \left( \sum_{d \in D_m} (\text{cost}_{d,z_i}) \right)
\]

• Choose machine having minimum value of maximum summed time series
Tenant to place
Summed CPU Usage over time
Summed CPU Usage over time

Machine 1

Machine 2

Tenant to place

Machine 1
Summed CPU Usage over time

Machine 1

Time

Machine 2

Tenant to place

Time

Machine 1

Machine 2
Summed CPU Usage over time

Machine 1

Machine 2

Tenant to place

Max load

Machine 1

Machine 2
Summed CPU Usage over time

Machine 1

Time

Machine 2

Time

Tenant to place

Time

Max load

Max load

Machine 1

Machine 2
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

\[\text{FFT Coefficients}\]

\textbf{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: \( \text{numDB/numMachines} \)
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 $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{6}$th 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
95% fewer violations
PMAX Cost

MSFT Cost

Allocation Type

Scalar
Random
"Best Fit" algorithm

Best fit produces a lot more violations

95% fewer violations

Tenant Violations per Hour

Cluster Size (Nodes)
Allocation Type

Scalar | Summed Time Series | FFT

Tenant Violations per Hour

Cluster Size (Nodes)

PMAX Cost

MSFT Cost
No improvements
The graph shows the allocation type and cost comparison for different cluster sizes. The x-axis represents the cluster size (in nodes) and the y-axis represents the number of tenant violations per hour. The graphs indicate that there are no improvements with densely packed scenarios.

For the PMAX Cost graph, the y-axis represents the average cost per month (in millions), and the x-axis represents the cluster size (in nodes). The bars show that the cost remains relatively constant across different cluster sizes.

For the MSFT Cost graph, similar trends are observed, with the cost per month remaining stable across various cluster sizes.

The graphs suggest that the cost for both allocation types remains stable regardless of the cluster size, with no significant improvements observed in densely packed scenarios.
Every week, moved at most one
Every week, moved at most one
Every week, moved at most one

More frequent migrations?
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?