Kerveros: Efficient and Scalable Cloud Admission Control

Big cloud server shortage could slow generative AI's breakneck pace

Article by Jacob Bourne | Apr 13, 2023

Microsoft Azure Reportedly Experiencing Capacity Shortages Amid Global Supply Chain Issues

Blog / Azure / Post
Microsoft Azure

- 65+ Azure regions
- 200+ datacenters worldwide
- 1K+ supported VM types
- 3M+ machines
- 14M+ VM Requests per hour
Cloud is Finite

- Availability Zone
- Running VM
- Capacity Reservation
- Failure and Maintenance
Cloud is Finite

- Availability Zone
- Running VM
- Capacity Reservation
- Failure and Maintenance
- Failed VM
Cloud is Finite

Admission Control: Should a new request be accepted?

- Availability Zone
- Running VM
- Capacity Reservation
- Failure and Maintenance
- Failed VM

VM Request
Admission Control in Azure

Admission Control: Should a new request be accepted?

Available Resources = Total Resources − Allocated Resources

Why is it hard?

- Network and Machine Failures
- Scheduled Maintenance
- Unscheduled Maintenance
- VM Requests
- Capacity Reservations
- Customer Scale-Outs

- Variability affecting supply and demand
Admission Control in Azure

**Admission Control:** Should a new request be accepted?

**Why is it hard?**

- Variability affecting supply and demand
- Hardware and VM type heterogeneity
Admission Control in Azure

Admission Control: Should a new request be accepted?

Why is it hard?

• Variability affecting supply and demand
• Hardware and VM type heterogeneity
  → fragmentation
Admission Control in Azure

Admission Control: Should a new request be accepted?

Why is it hard?

• Variability affecting supply and demand
• Hardware and VM type heterogeneity
  → fragmentation
• Placement constraints
Admission Control in Azure

Admission Control: Should a new request be accepted?

Solution: Kerveros: Cloud admission control at scale

Why is it hard?

- Variability affecting supply and demand
- Hardware and VM type heterogeneity → fragmentation
- Placement constraints

Goals

- Fast and Scalable
  - Throughput = 120,000+ requests/minute [1]
  - Avg. Latency = 5 – 10 ms
- Resource Efficient
  - 1% efficiency gain → $100+ M/year savings [1]

[1] Protean, OSDI ’20
Main Idea:
Late Binding of Reserved Capacity for Admission Control

Why Late Binding?

• High packing efficiency
• Accurate accounting
  • Tracks across different VM types
• Flexible packing with low overhead
• Fast admission decision
• Unclaimed reserved resources reused as preemptable VMs (e.g., spot VMs)
  → maximize ROI
Challenges with Late Binding

“Available Capacity” ≥ New Request

200 - 120 = \[80 \geq 50\] 🟢

50

New Request

Two machines

100

System Capacity:

2 x 100 = 200

120

Unclaimed

Reserved Capacity: [120]
Challenges with Late Binding

“Available Capacity” ≥ New Request

\[ 200 - 120 = 80 \geq 50 \]  

Accept Request?

System Capacity:
\[ 2 \times 100 = 200 \]

Two machines

Small VMs
[120]
Challenges with Late Binding

“Available Capacity” ≥ New Request

200 - 120 = \color{blue}{80} \geq \color{green}{50}

Claiming Small VM Reservations

System Capacity:
2 \times 100 = 200

Accept Request?

Two machines

50
Challenges with Late Binding

“Available Capacity” ≥ New Request
200 - 120 = 80 ≥ 50

Accept Request?

System Capacity:
2 x 100 = 200

Two machines
Challenges with Late Binding

"Available Capacity" ≥ New Request

200 - 120 = $80 \geq 50$

System Capacity:

$2 \times 100 = 200$
Challenges with Late Binding

“Available Capacity” ≥ New Request

\[200 - 120 = 80 ≥ 50\]  

Accept Request?

System Capacity:
\[2 \times 100 = 200\]
Challenges with Late Binding

“Available Capacity” ≥ New Request

\[200 - 120 = 80 \geq 50\]

Accept Request?

- **Claiming Large VM Reservations**
- **SLA Violation**

Solution: Allocable VM (AV)
Allocable VM (AV)

• Novel bookkeeping of available capacity
  • For every VM type, count of additional VMs that can fit

<table>
<thead>
<tr>
<th>VM Type</th>
<th>AV count</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>27408</td>
</tr>
<tr>
<td>M</td>
<td>6724</td>
</tr>
<tr>
<td>L</td>
<td>1588</td>
</tr>
</tbody>
</table>
Allocable VM (AV)

- Novel bookkeeping of available capacity
  - For every VM type, count of additional VMs that can fit
- Converts multi-dimensional demand to a single-dimension
- Develop two algorithms to adjust AV count for reserved capacity
  - Conversion Ratio Algorithm (CRA)
  - Linear Adjustment Algorithm (LAA)

<table>
<thead>
<tr>
<th>VM Type</th>
<th>Multi-dimensional Resource demand</th>
<th>AV count</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>{ CPU: 1, RAM: 2 GB, Disk: 64 GB, ... }</td>
<td>27408</td>
</tr>
<tr>
<td>M</td>
<td>{ CPU: 4, RAM: 8 GB, Disk: 256 GB, ... }</td>
<td>6724</td>
</tr>
<tr>
<td>L</td>
<td>{ CPU: 16, RAM: 32 GB, Disk: 1024 GB, ... }</td>
<td>1588</td>
</tr>
</tbody>
</table>
Kerveros In Action

Client Services

Allocation Worker Instances

Load Balancer

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

- Zonal admission control
- Considers all reserved capacity in zone
- Handles both VM and reservation requests
Kerveros In Action

Client Services

Allocation Worker Instances

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

Placement Store

VM Placement & Reserved Capacity State

- Zonal admission control
- Considers all reserved capacity in zone
- Handles both VM and reservation requests

**Request Handler Process**
- Request arrives → check AV count
- If enough AV in system, **Accept**
  - Update VM placement & reserved capacity state
- Else **Reject**

<table>
<thead>
<tr>
<th>VM Type</th>
<th>AV Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AV_S</td>
</tr>
<tr>
<td>M</td>
<td>AV_M</td>
</tr>
<tr>
<td>L</td>
<td>AV_L</td>
</tr>
</tbody>
</table>
Kerveros In Action

AV Count Estimation
- Initialize AV count in zone
  - Uses in-memory state snapshot
  - Counted independently for each VM type
- Subtracts AV count for reserved capacity
  - Convert between VM types

Conversion Ratio Algorithm (CRA)
- Converts AV count between VM types
- Handles multi-dimensional conversion
- Frequent AV count estimation: 1 minute

How do we get it?

<table>
<thead>
<tr>
<th>VM Type</th>
<th>AV Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AV_S</td>
</tr>
<tr>
<td>M</td>
<td>AV_M</td>
</tr>
<tr>
<td>L</td>
<td>AV_L</td>
</tr>
</tbody>
</table>
Kerveros In Action

Client Services ➔ Load Balancer ➔ Allocation Worker Instances

- Request Handlers
- VM Placement & Reserved Capacity State Snapshot
- AV Count Estimator (CRA)

AV Count Estimation
- Initialize AV count in zone
  - Uses in-memory state snapshot
  - Counted independently for each VM type
- Subtracts AV count for reserved capacity
  - Convert between VM types

Conversion Ratio Algorithm (CRA)
- Converts AV count between VM types
- Handles multi-dimensional conversion
- Frequent AV count estimation: 1 minute

Fast and Scalable

Rounding Errors ➔ Fragmentation

Conservative Estimation
Kerveros In Action

Client Services

Load Balancer

Allocation Worker Instances

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

Placement Store

VM Placement & Reserved Capacity State

• Common components with allocator

Linear Adjustment Estimator

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot
Kerveros In Action

Client Services

Allocation Worker Instances

- Common components with allocator
- Synthetic request for emulation

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

Placement Store

VM Placement & Reserved Capacity State

Linear Adjustment Estimator

Request Generator

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot
Kerveros In Action

Client Services

Load Balancer

Allocation Worker Instances

- Common components with allocator
- Synthetic request for emulation
- **Update: 30 minutes**

Request Handlers

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

Placement Store

ML Platform

Linear Adjustment Estimator

Request Handlers

Request Generator

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot
Linear Adjustment Algorithm (LAA)

- Accurate → Resource efficient
- Slow, compute intensive

Common components with allocator

Synthetic request for emulation

Update: 30 minutes
Kerveros In Action

Client Services

Allocation Worker Instances

VM Placement & Reserved Capacity State

AV Count Estimator (CRA)

Request Handlers

VM Placement & Reserved Capacity State Snapshot

Pub/Sub

ML Platform

AV Count Estimator (CRA)

VM Placement & Reserved Capacity State Snapshot

Linear Adjustment Estimator

Request Generators

Request Handlers
Kerveros In Action

Client Services

Load Balancer

Allocation Worker Instances

AV Count Estimator (CRA)

VM Placement & Reserved Capacity

State Snapshot

Request Generator

Request Handlers

AV Count Estimator (CRA)

Linear Adjustment Estimator

VM Placement & Reserved Capacity

State Snapshot

Placement Store

ML Platform

Pub/Sub

Fast but Conservative

Slow but Accurate

Kerveros: Fast and Accurate
Alternate Solutions

• **Partition (PT)**[^SOSP ’21]
  • Approach: Reserve capacity by partitioning machines
  • Pro: Greater control over resources and isolation → Works on private cloud
  • Con: Fragmentation with high heterogeneity → Wastes resources in public cloud

• **Placeholder (PH)**
  • Approach: Allocate and reserve resources for reservations
  • Pro: Simple and Guarantees SLA
  • Con: Early binding to allocated resources → Low packing efficiency
How Resource Efficient is Kerveros?

Kerveros ensures high resource utilization.
How does Kerveros Deal with Failures?

Kerveros achieves consistent fours 9s of availability.
How Scalable is Kerveros?

Kerveros scales well with inventory size.
Conclusion

• **Kerveros**: Admission control system in Microsoft Azure
  • Variable supply and demand
  • Hardware and VM type heterogeneity

• Scalable and resource efficient in cloud scale

• Achieves high resource utilization while maintaining SLA
  • Late binding of reserved resources for admission control
  • Allocable VM (AV)