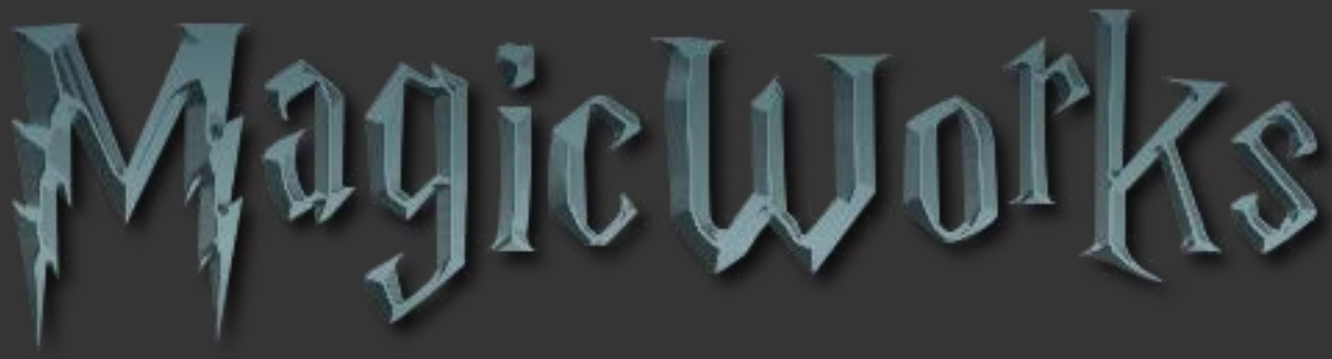
# What's the Problem?




MagicWorks™ are struggling

- Analytics queries too slow
- Data volumes are growing
- Their servers are at capacity
- Queries are already tuned by SQL experts


# Azure SQL Data Warehouse

# Agenda

Service architecture

Elastic performance and scalability

Table storage

Business Continuity

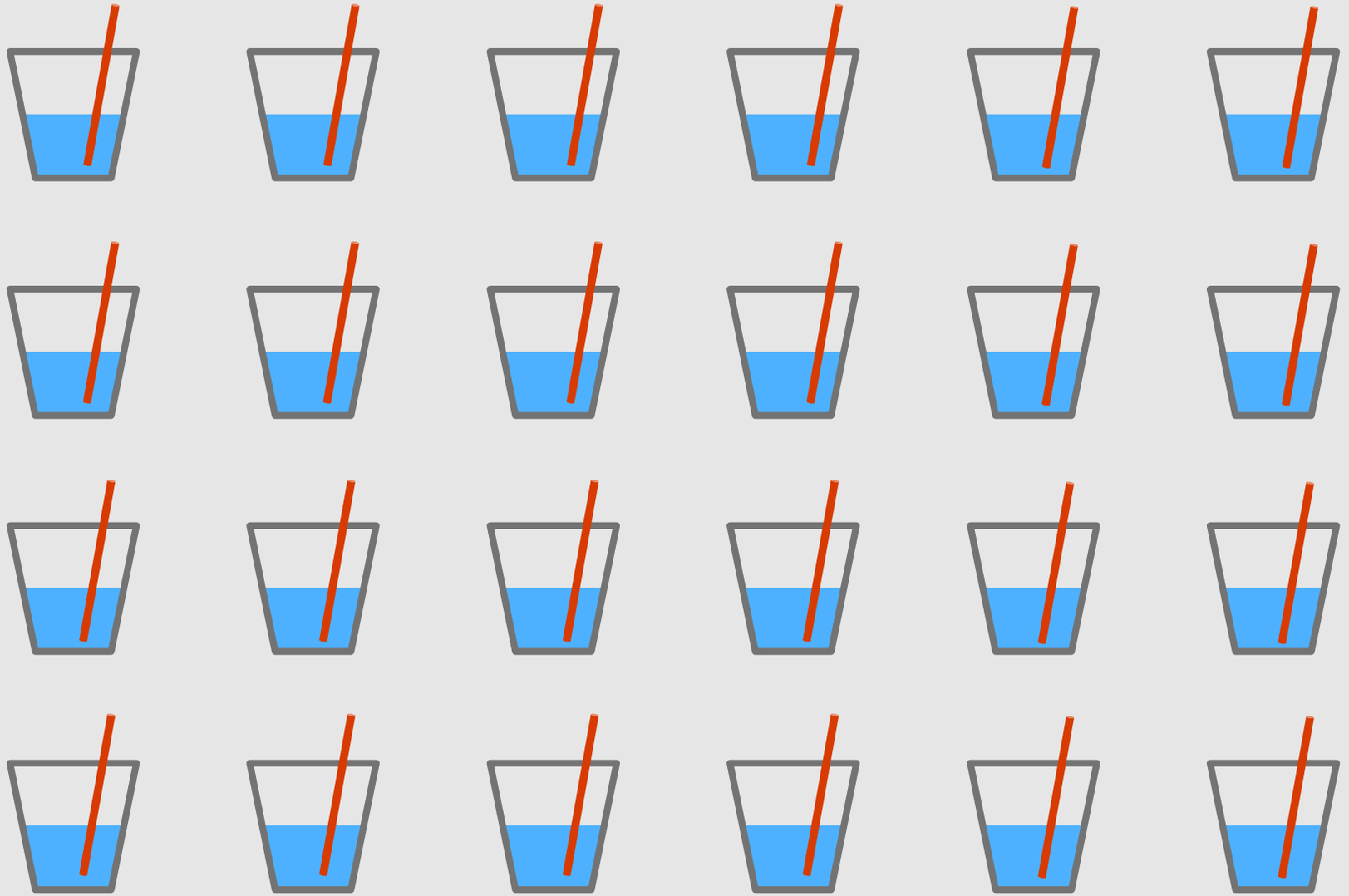
Workload Management

Service Integration

# But first…

Who likes Drinking Games?

# Scaling out: The ultimate team game…



# Target workload: Analytics

Store large volumes of data

Consolidate disparate data into a single location

Shape, model, transform and aggregate data

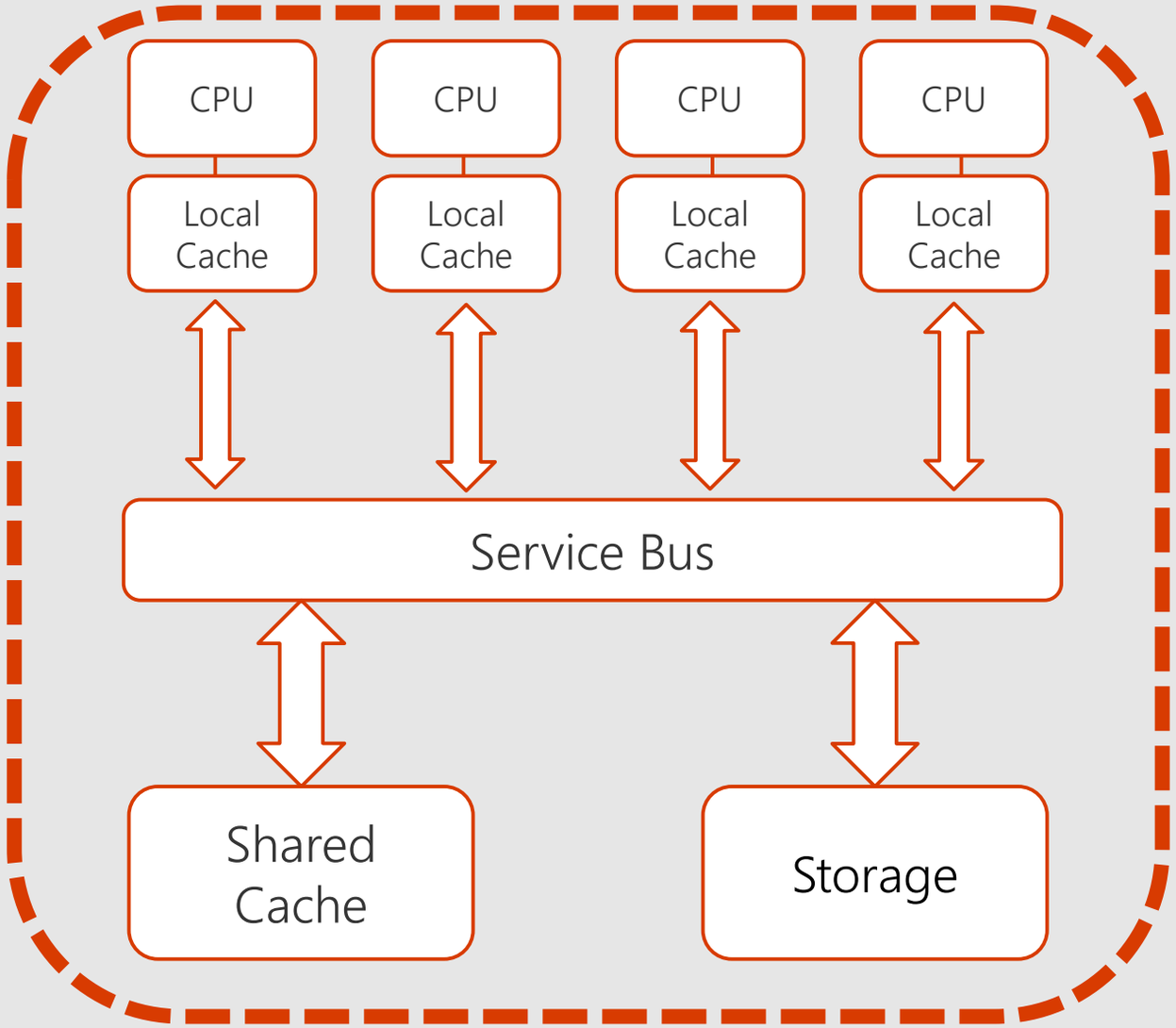
Perform query analysis across large datasets

Ad-hoc reporting across large data volumes

All using simple SQL constructs

"SQL on SQL"

# SQLDW Service Architecture

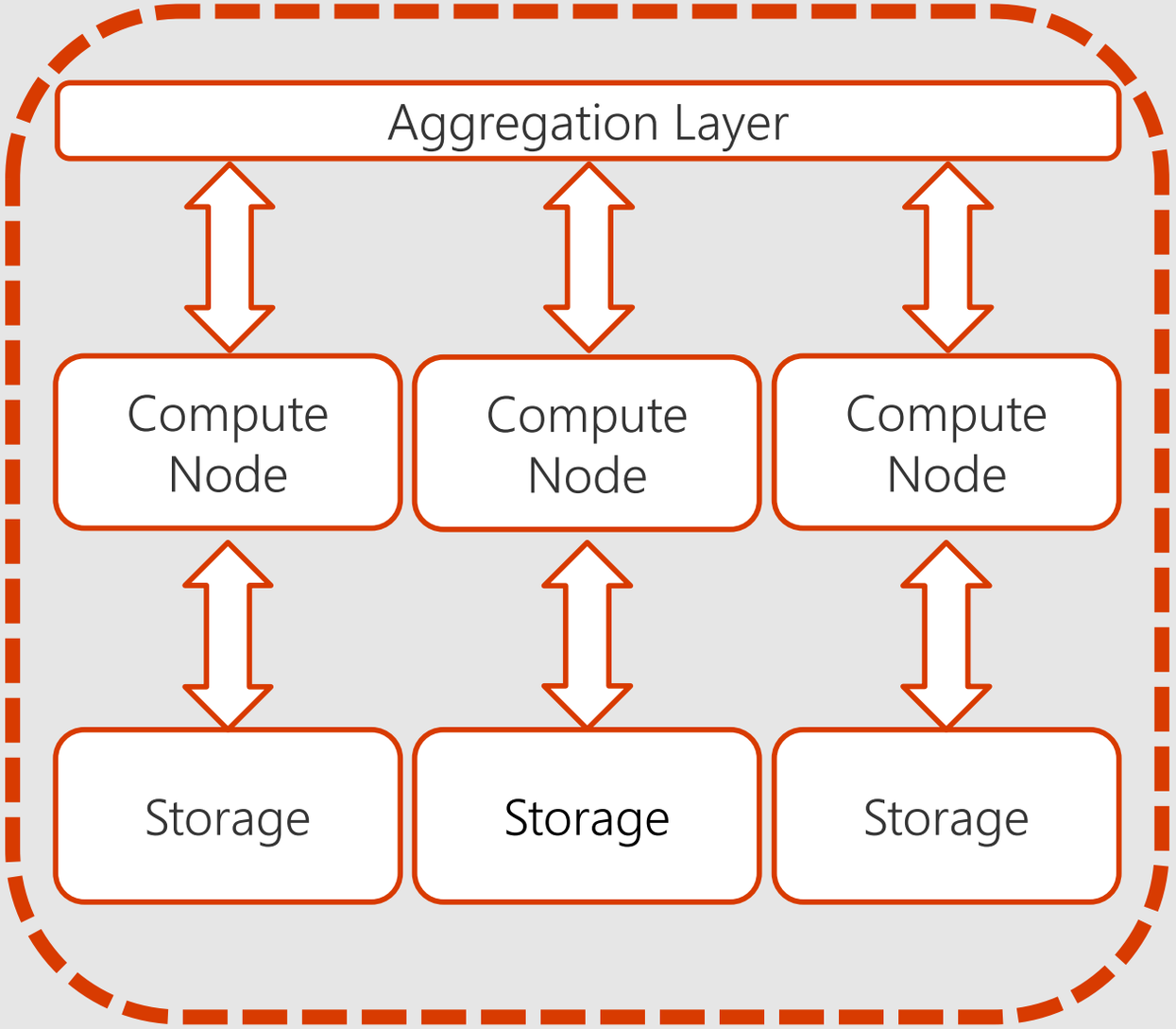


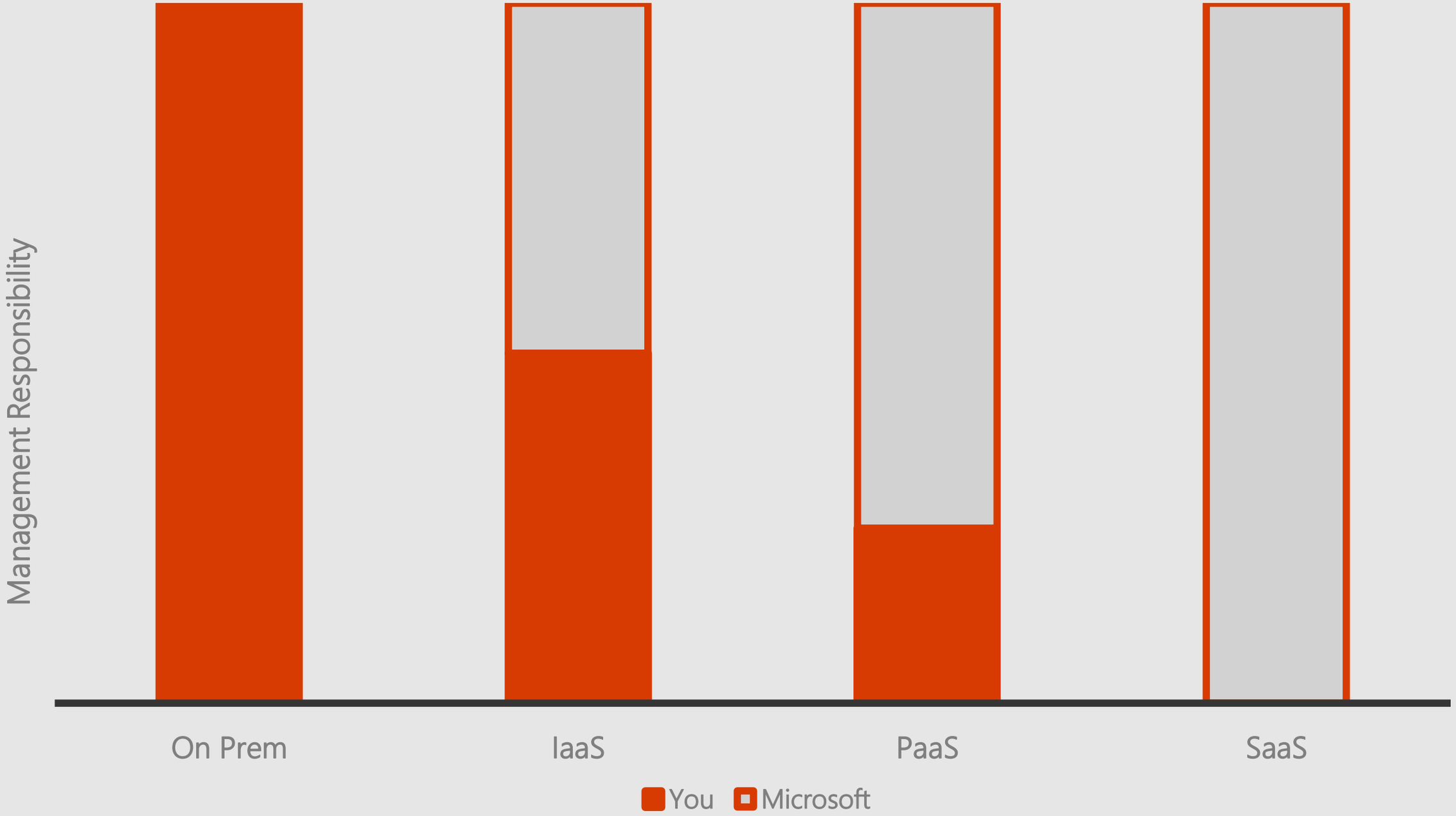

# SMP

Symmetric Multi-Processing
(ie: A Standard SQL Server)

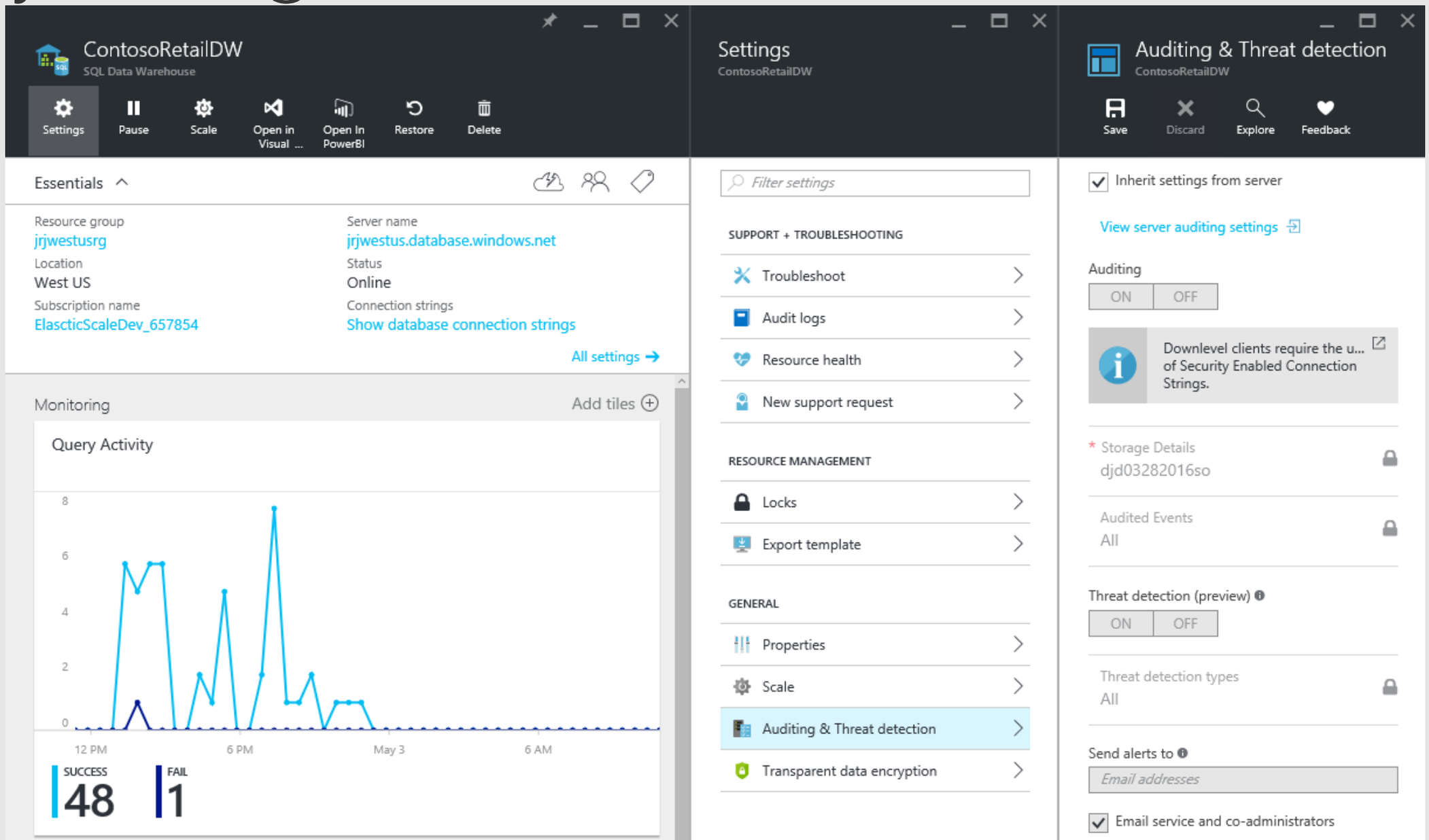
# MPP

Massively Parallel Processing




Management Responsibility
On Prem
IaaS
PaaS
SaaS
You
Microsoft

# Fully managed PaaS



# Connectivity

Windows or Linux

ODBC

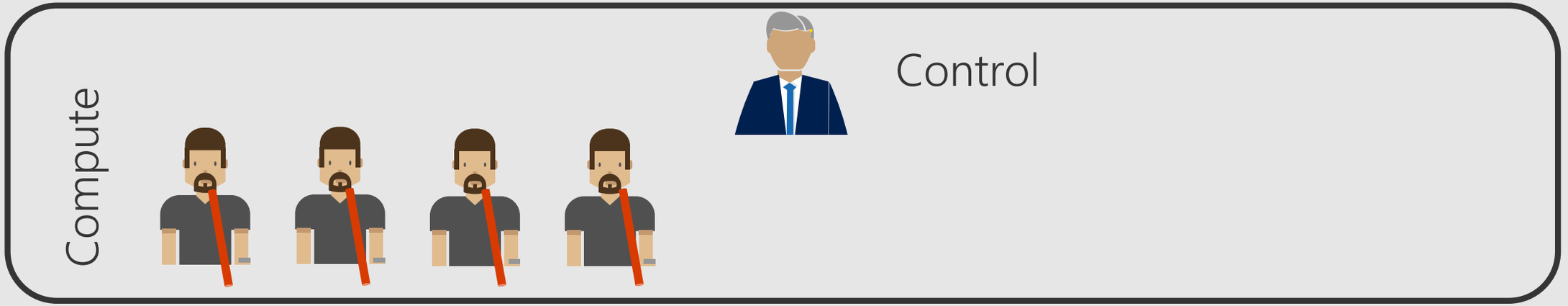
OLEDB

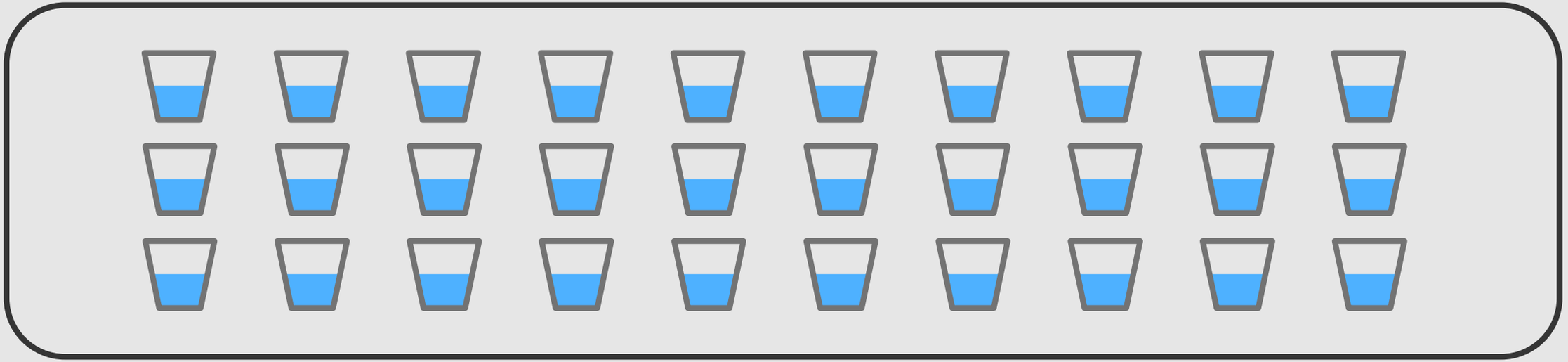
JDBC

ADO.NET

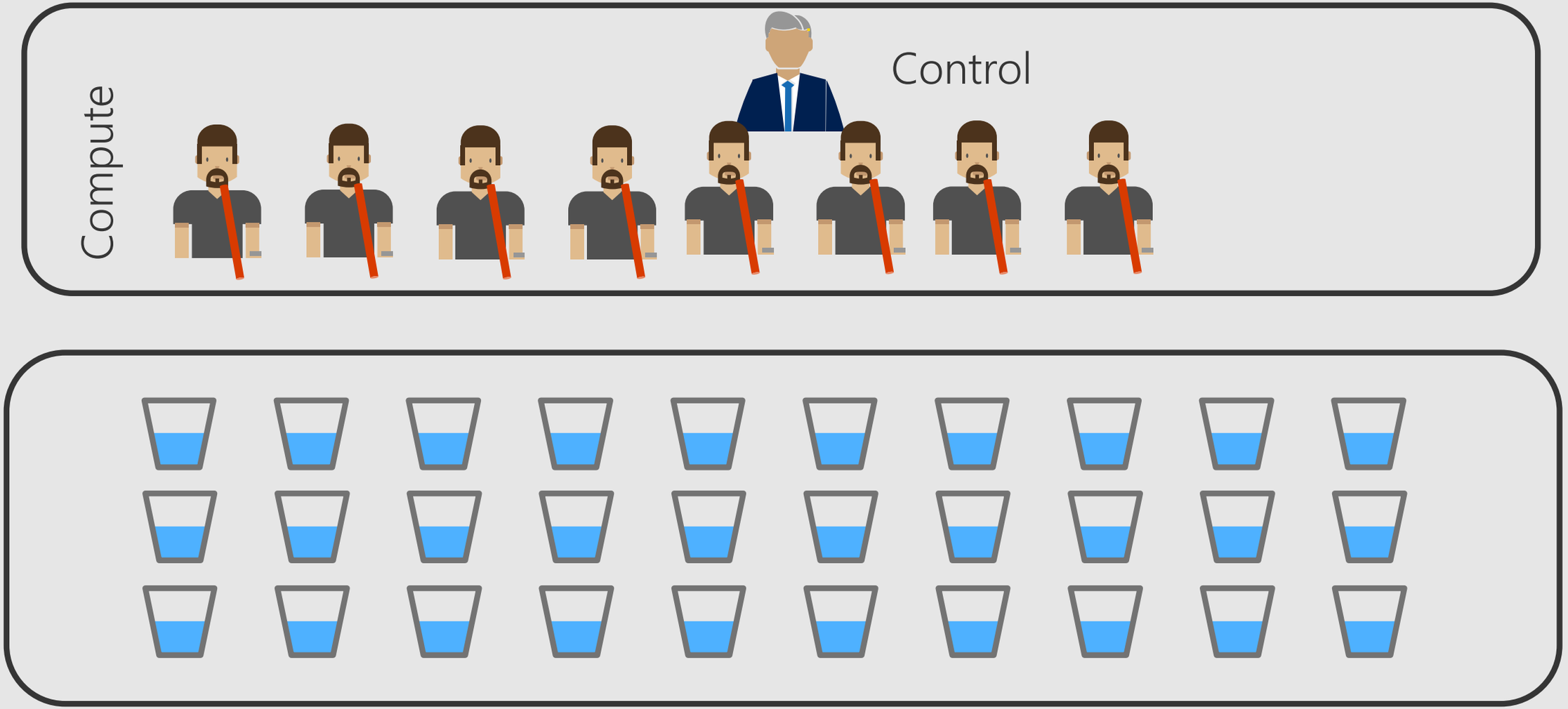
PHP

# Separation of compute from storage

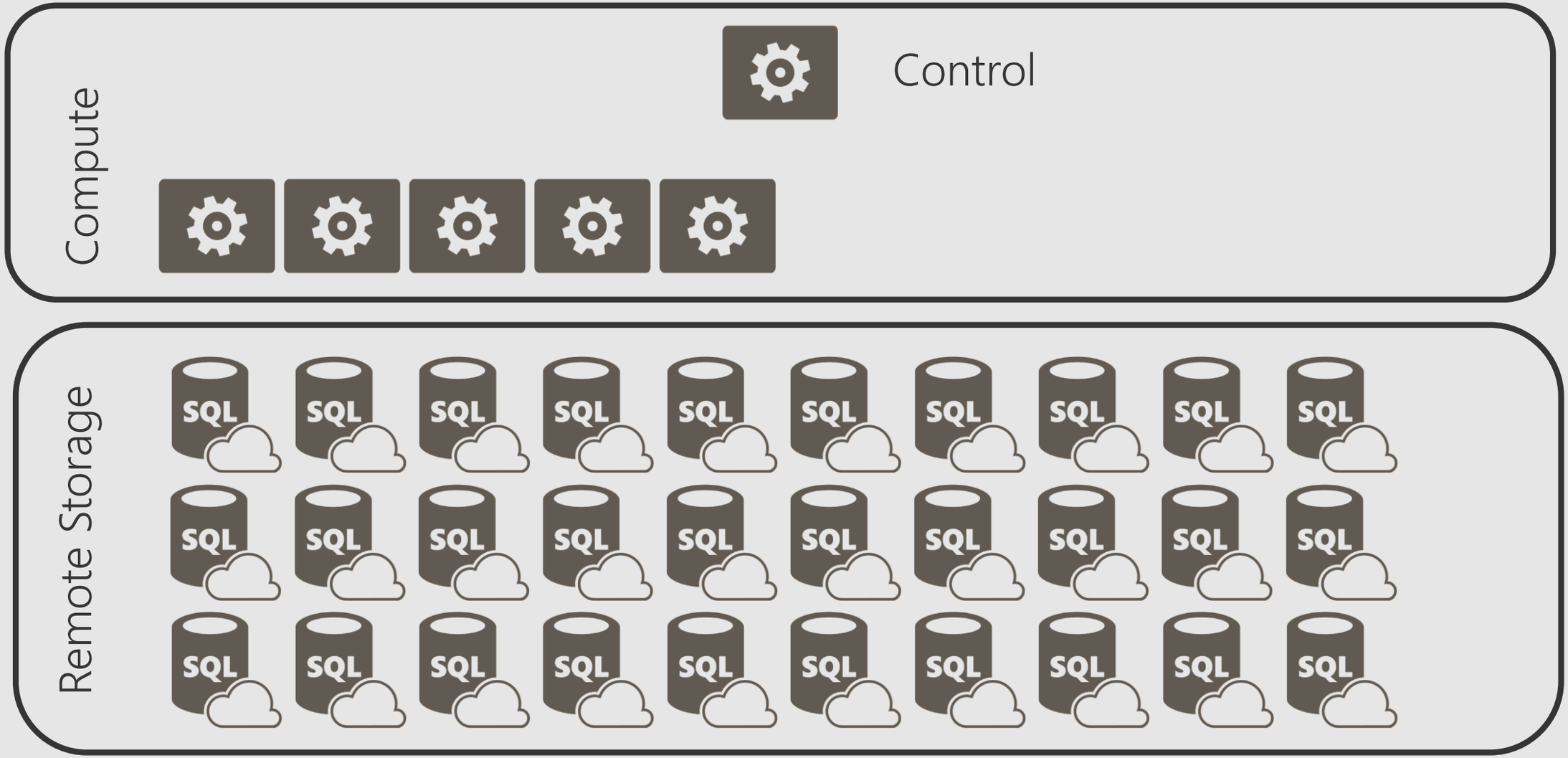



# Separation of compute from storage

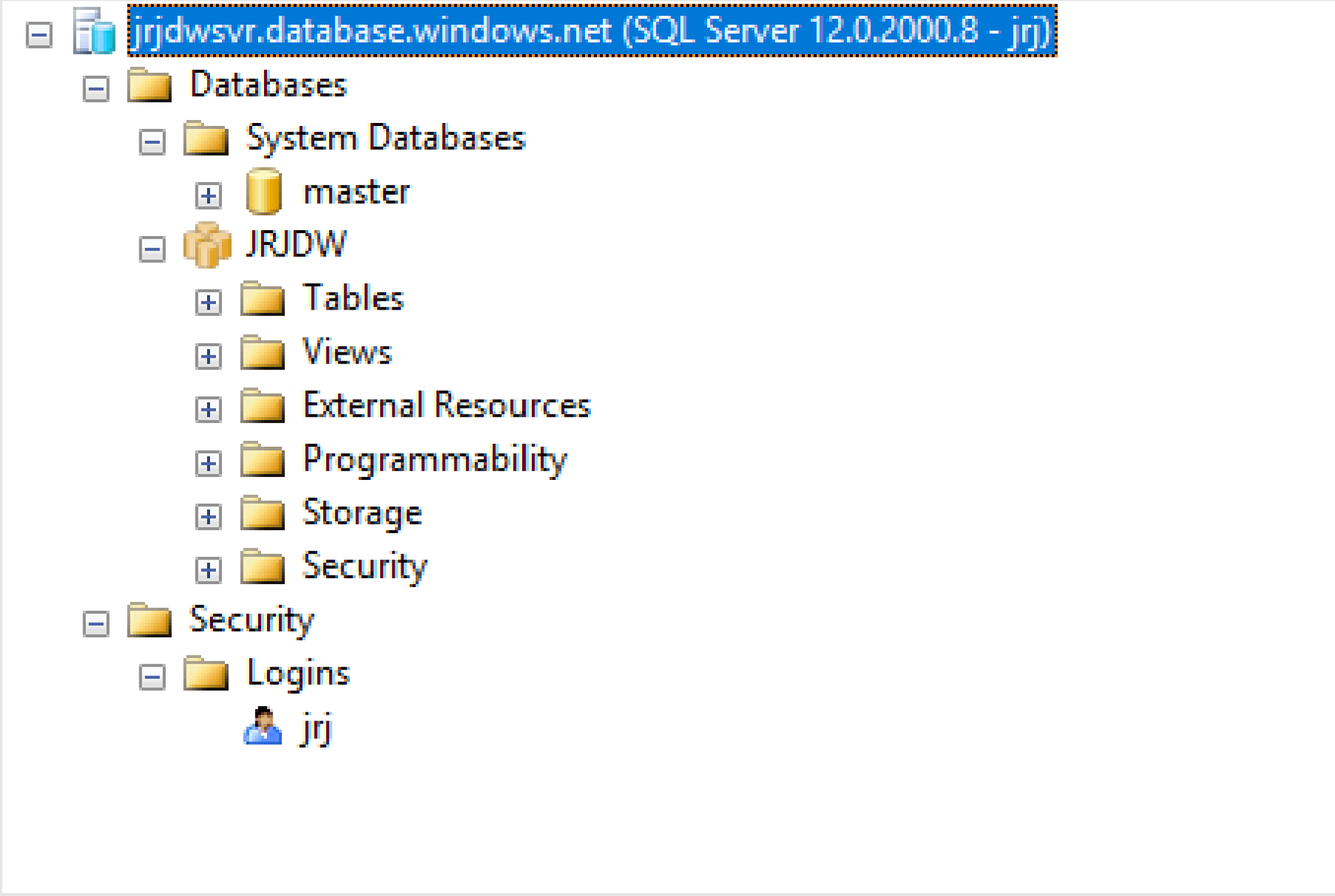



# Separation of compute from storage



# Instance

- Logical server
- Database
  - Schemas
  - Tables
  - Views
  - Stored Procedures
  - UDFs
- Security
  - Logins
  - Users
  - Certificates



# Demo: Creating & Connecting to Azure SQLDW

# Nodes:

```
SELECT   [pdw_node_id]    AS node_id
,        [type]           AS node_type
,        [name]           AS node_name
FROM     sys.[dm_pdw_nodes]
;
```

# Distributions:

```
SELECT   [distribution_id]    AS dist_id
,        [pdw_node_id]        AS node_id
,        [name]               AS dist_name
,        [position]           AS dist_position
FROM     sys.[pdw_distributions]
;
```

# Lab 001 – Connecting to SQLDW
*10 mins*

# Elastic performance & scalability

# Scale up architecture

One bucket (motherboard)

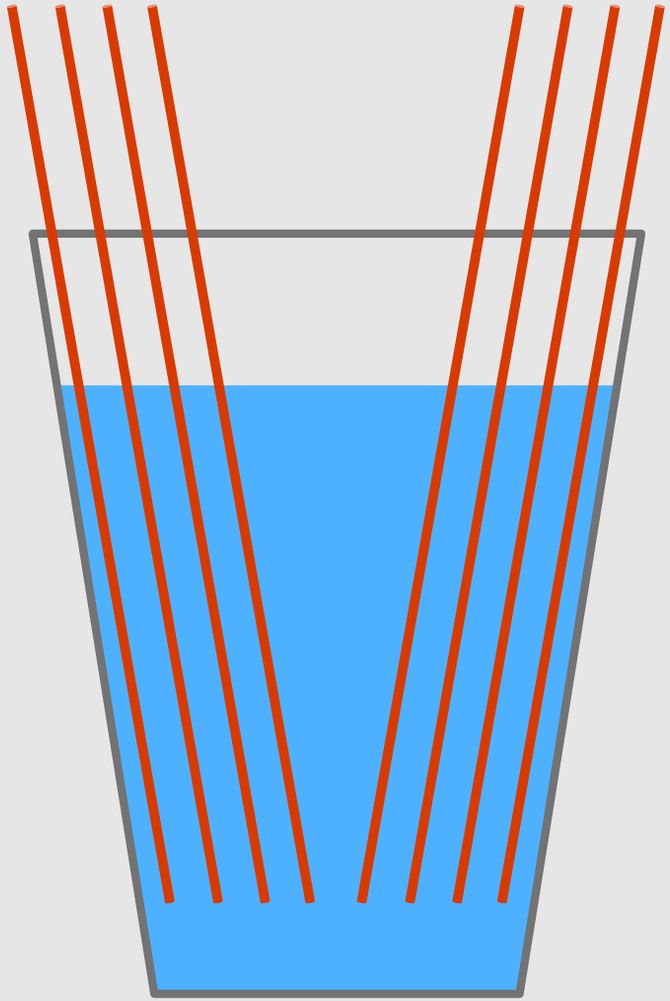
Contains all the water (resources)

Drinking through straws (logical procs)

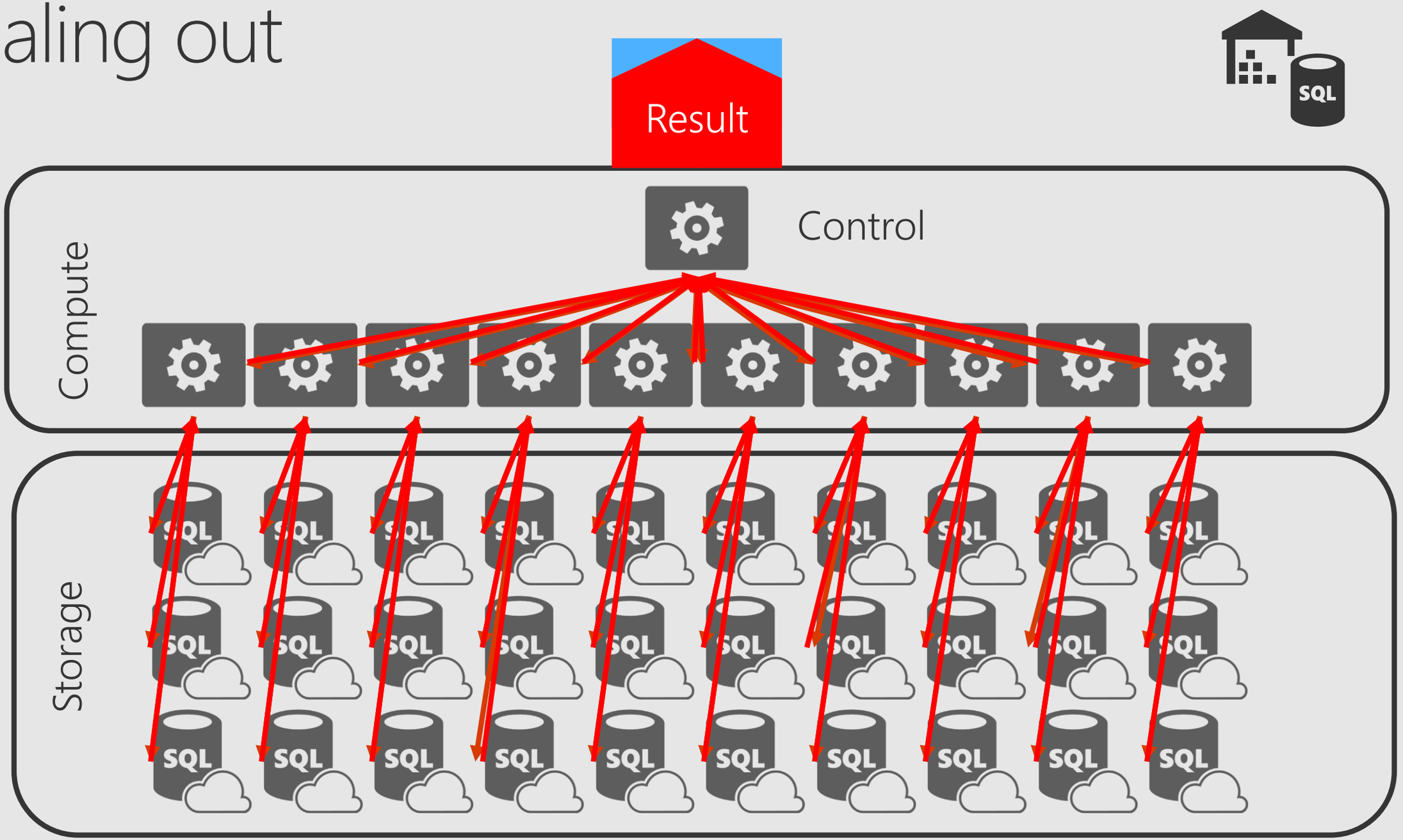
Want more resources?

- Buy a bigger bucket

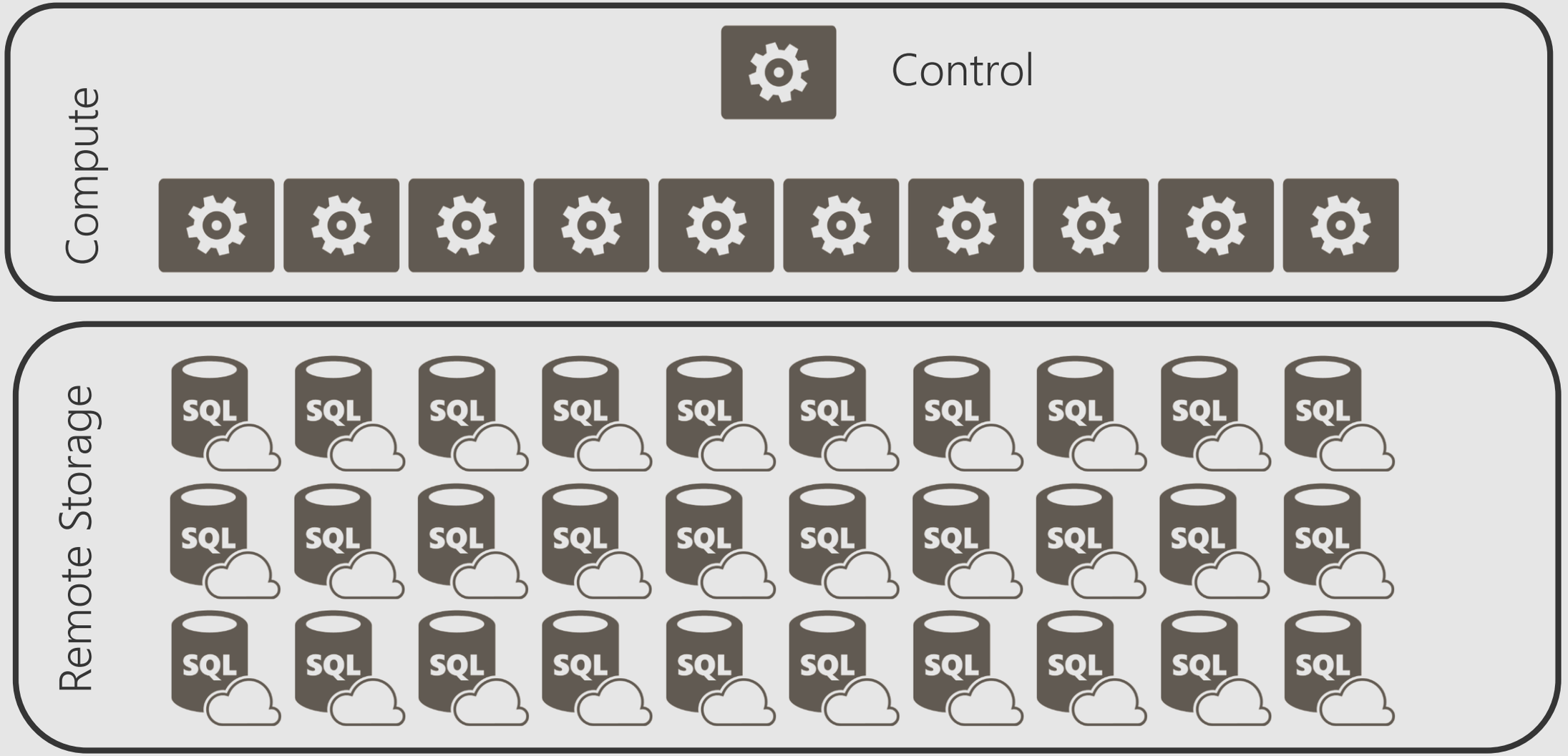
Sometimes you only get one straw…



# Scaling out



# Independently scale compute



# Data Warehouse Units

Normalized amount of compute

Converts to billing units i.e. what you pay


Relates directly to number of compute nodes

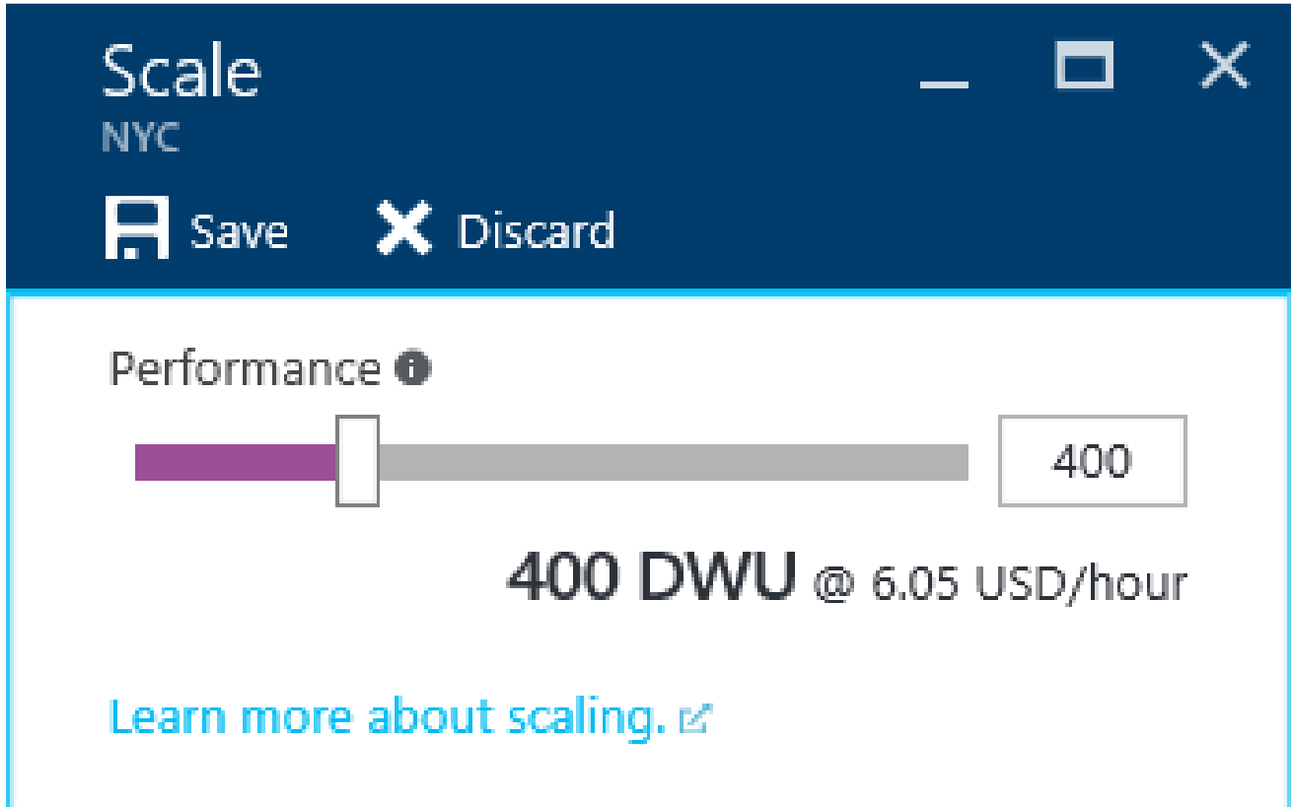
| DWU |
| --- |
| 100 |
| 200 |
| 300 |
| 400 |
| 500 |
| 600 |
| 1000 |
| 1200 |
| 1500 |
| 2000 |
| 3000 |
| 6000 |

# Service objective

```
SELECT   db.[name]                      AS [db_name]
,        ds.[edition]                   AS [db_edition]
,        ds.[service_objective]         AS [db_slo]
FROM     sys.[database_service_objectives]   AS ds
JOIN     sys.[databases]                     AS db
ON       ds.[database_id] = db.[database_id]
WHERE    ds.[edition]     = 'DataWarehouse'
;
```

# Changing Service Level Objectives (SLO)

```
ALTER DATABASE ContosoDW
MODIFY
(service_objective = 'DW1000'
)
;
```



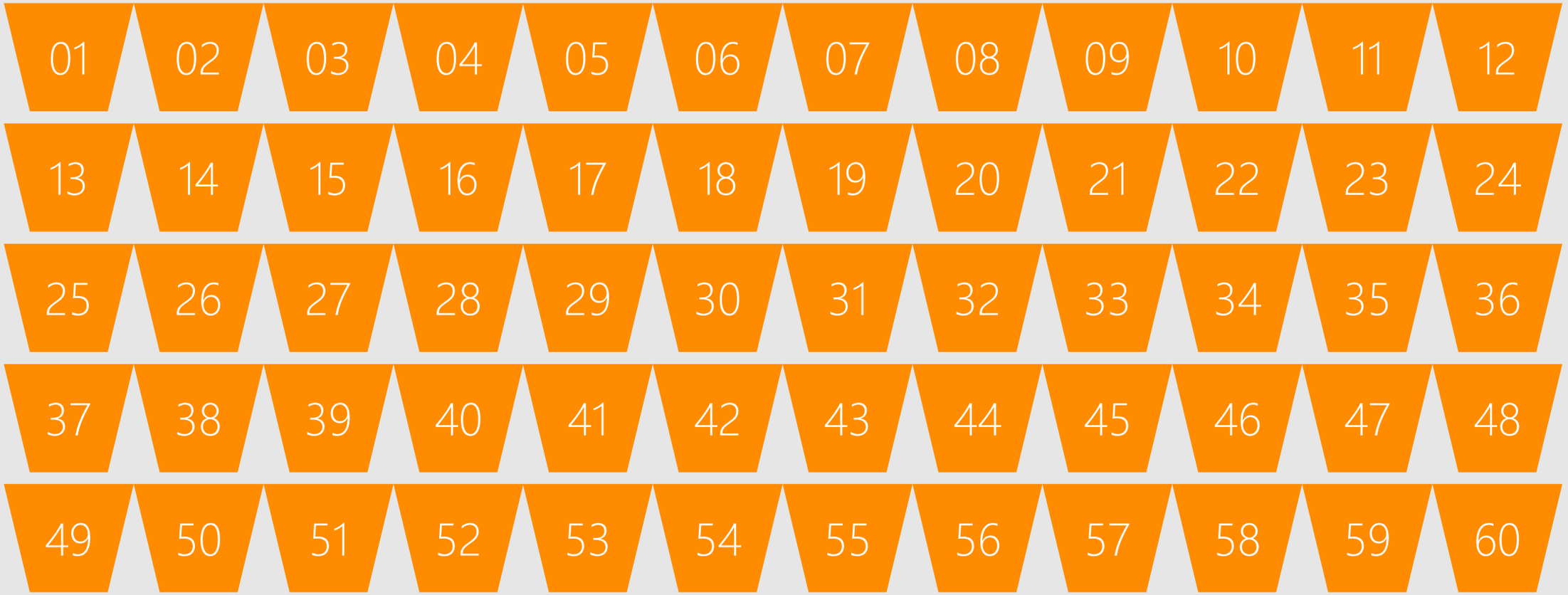
# Changing Service Level Objectives (SLO)

```
Set-AzureRmSqlDatabase
        -DatabaseName “Database"
        -ServerName “Server"
        -RequestedServiceObjectiveName "DW1000"
```

*PowerShell*

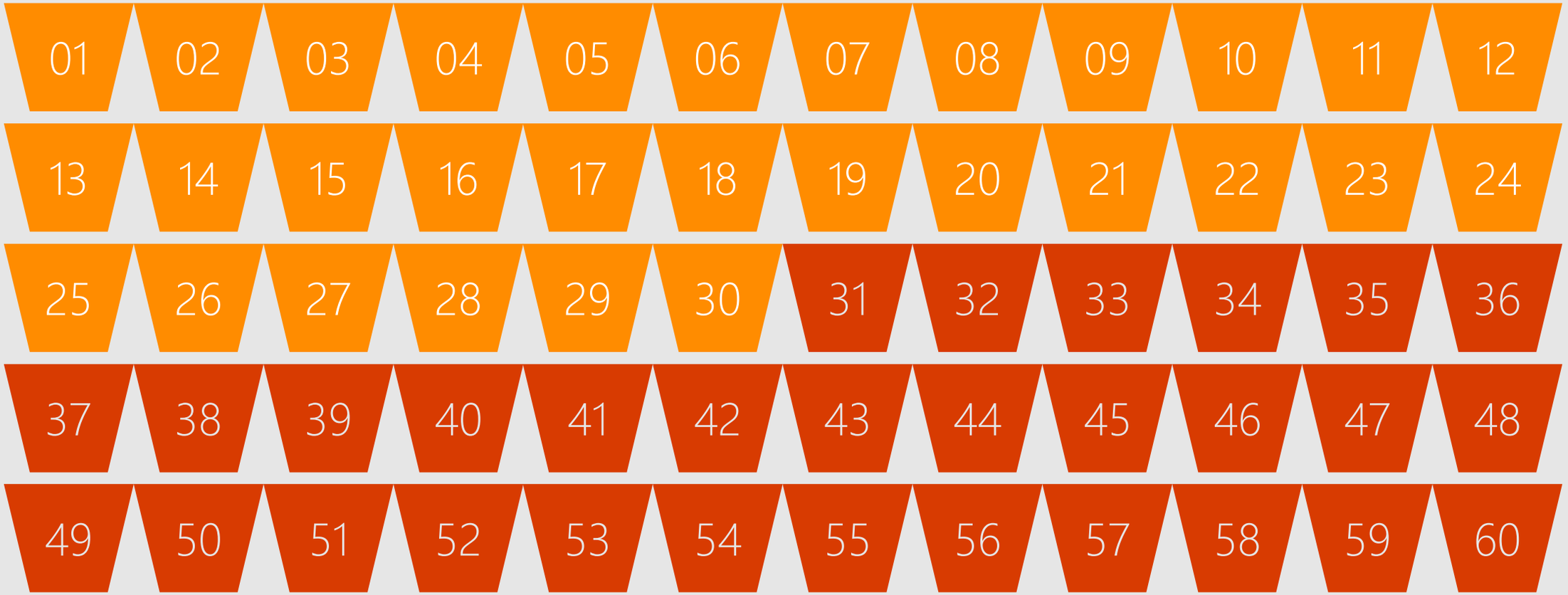
# Mapping Compute in SQLDW

DW100



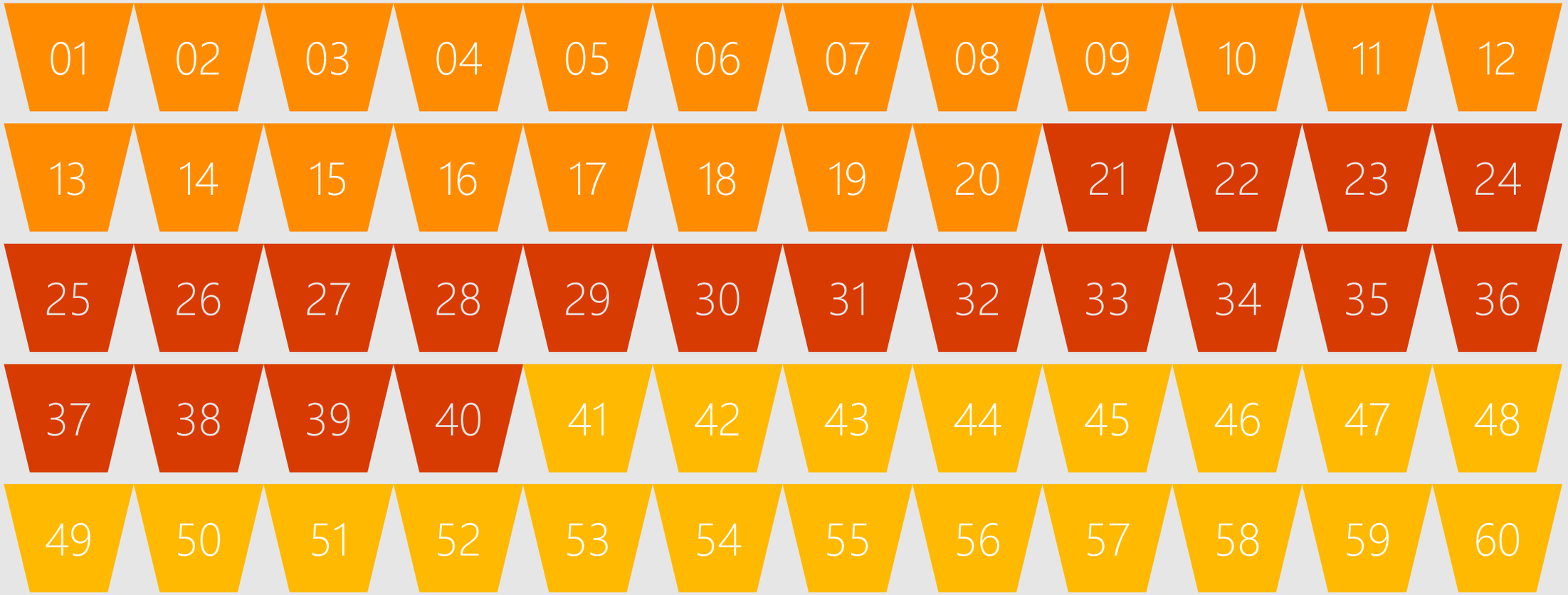
# Mapping Compute in SQLDW

DW200



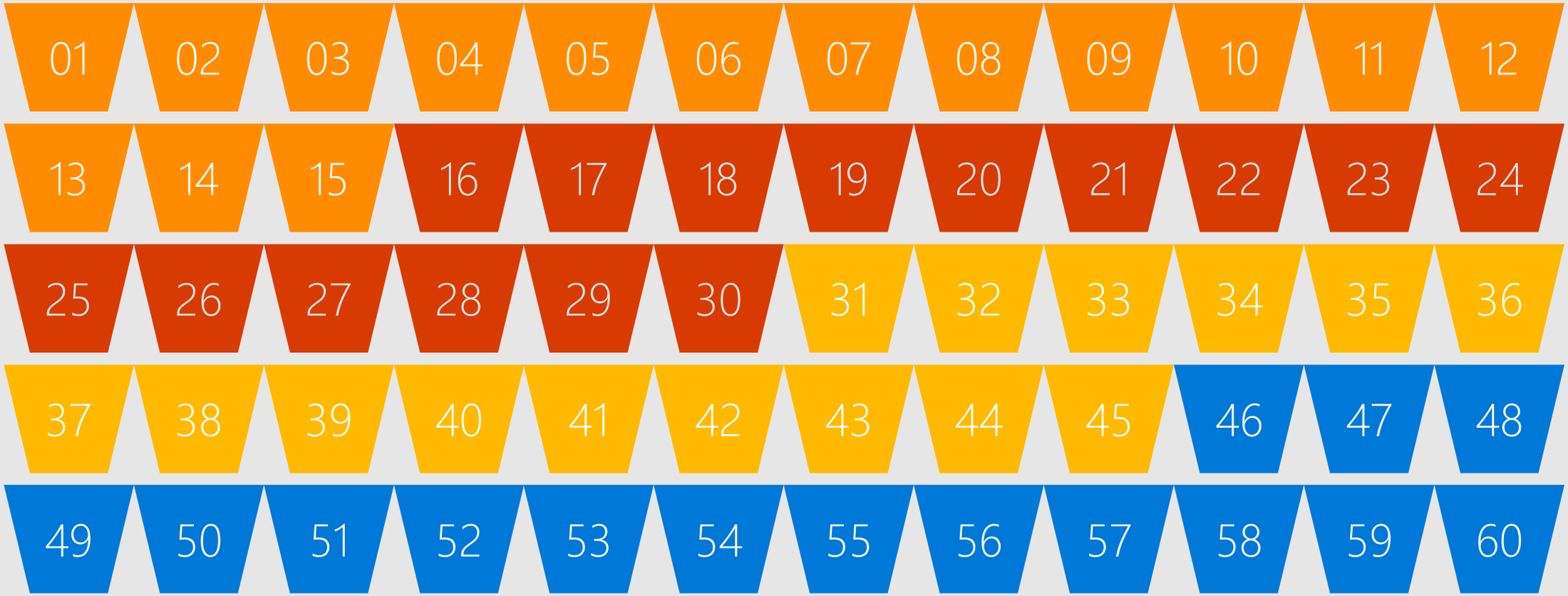
# Mapping Compute in SQLDW

DW300



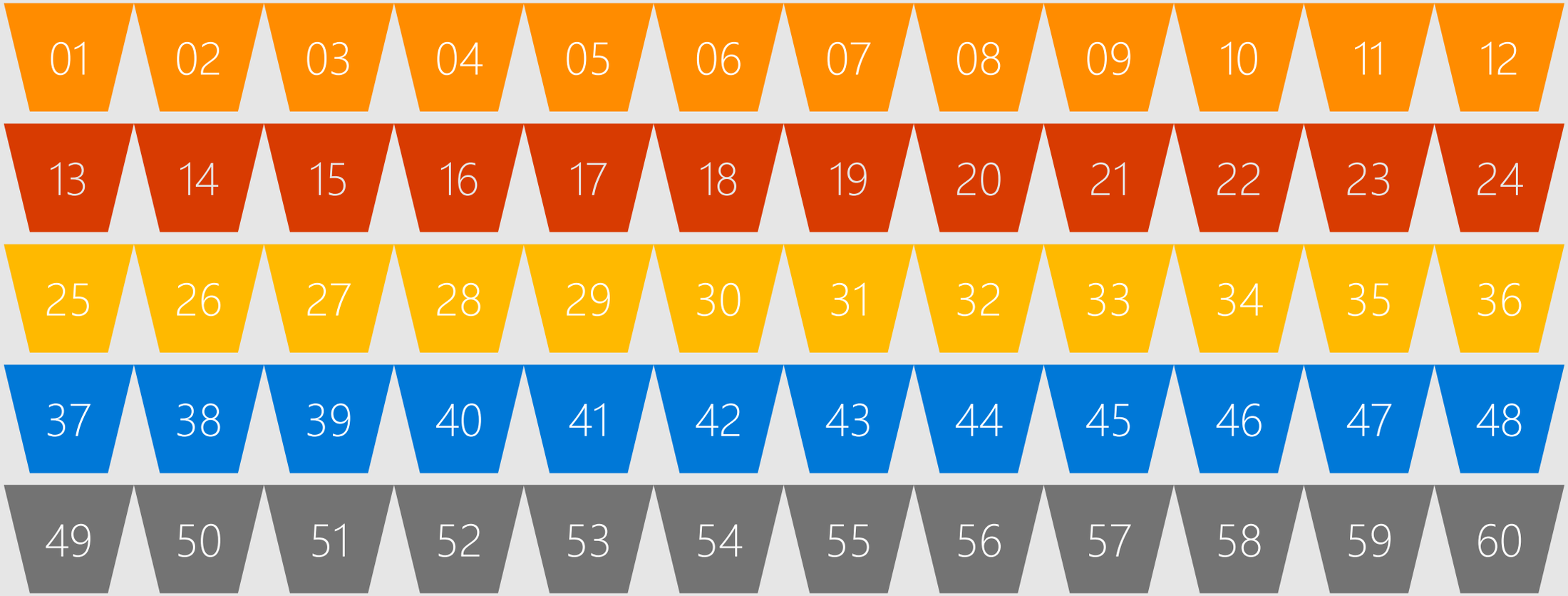
# Mapping Compute in SQLDW

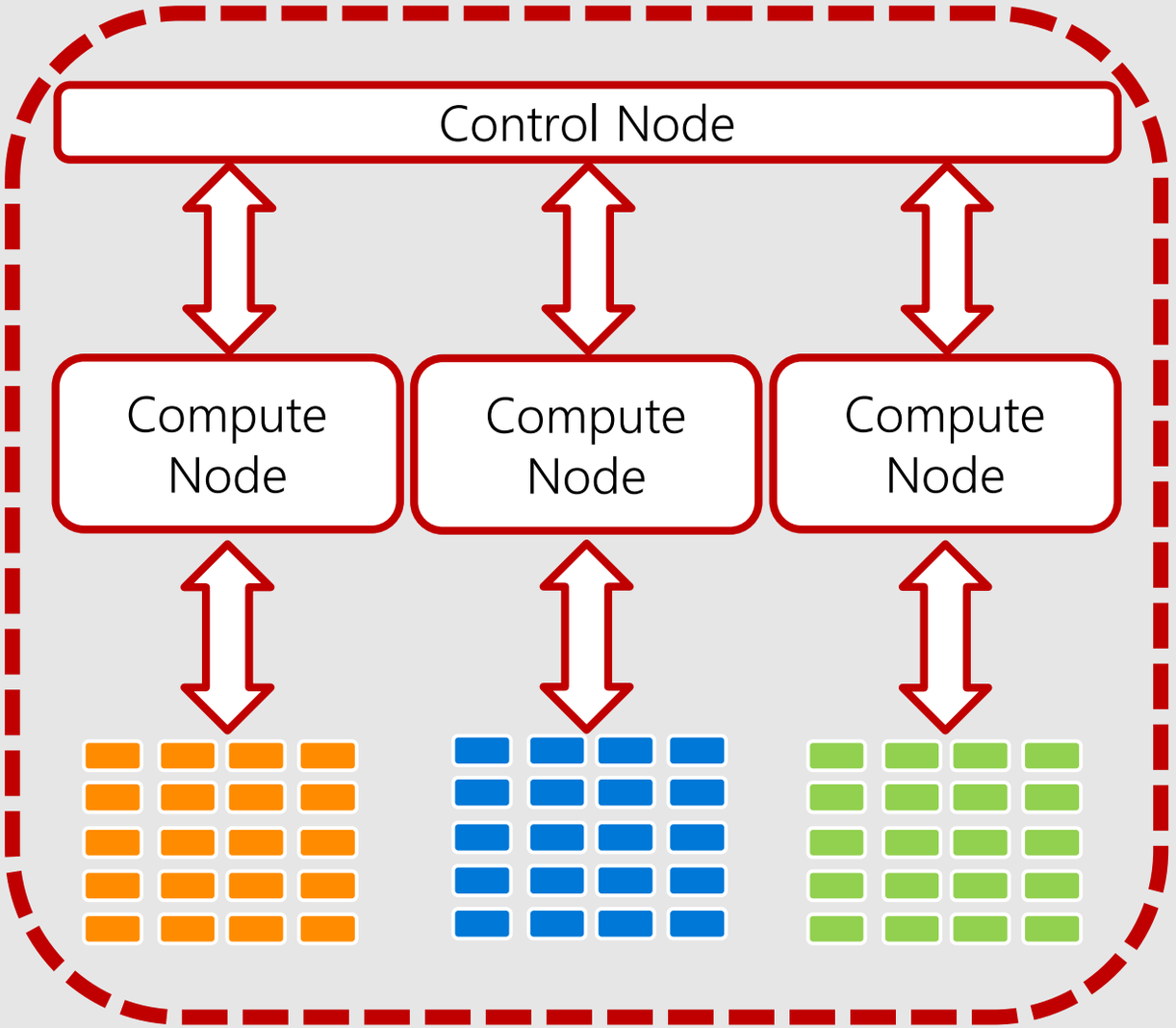
DW400



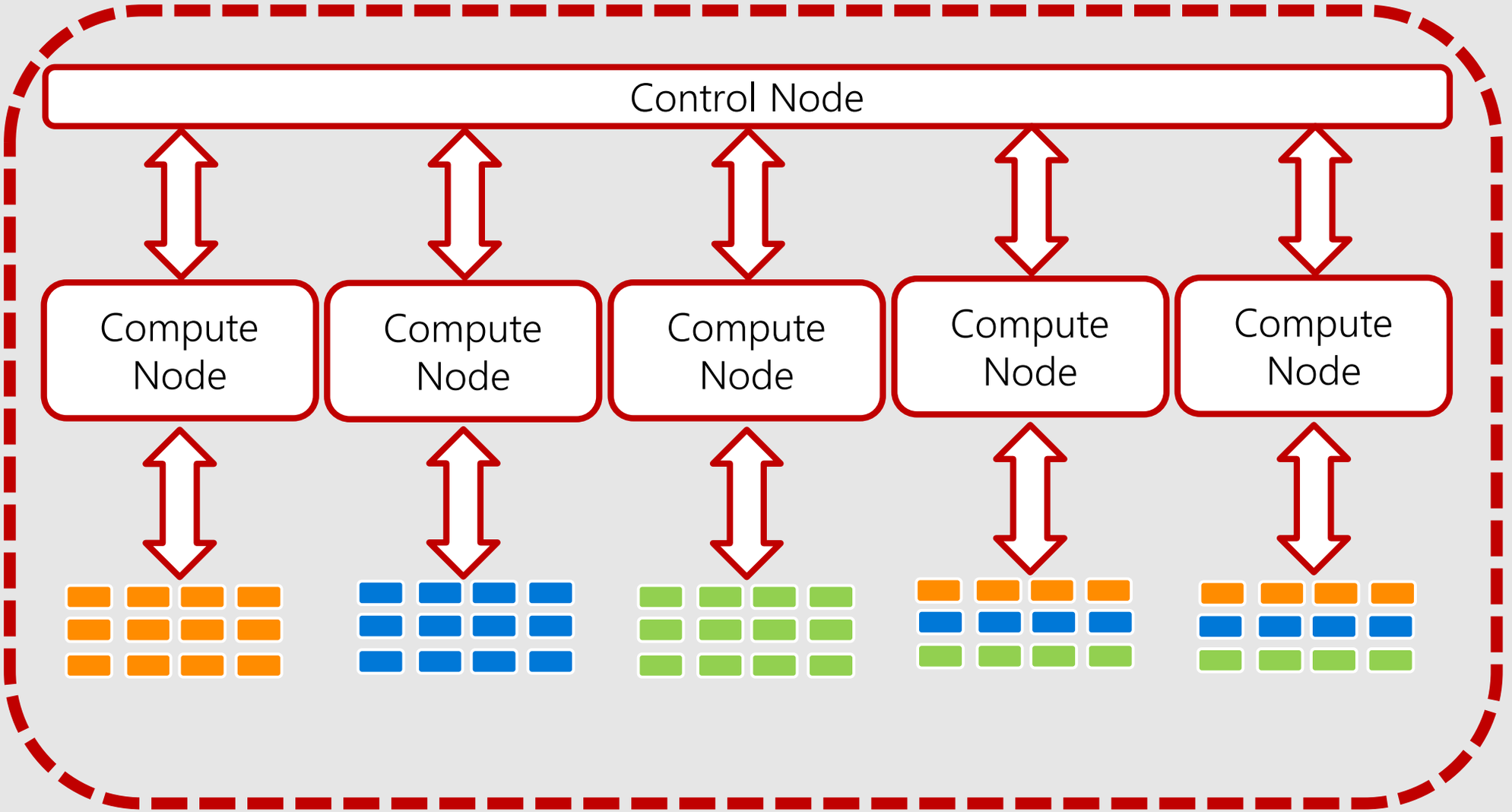
# Mapping Compute in SQLDW

DW500



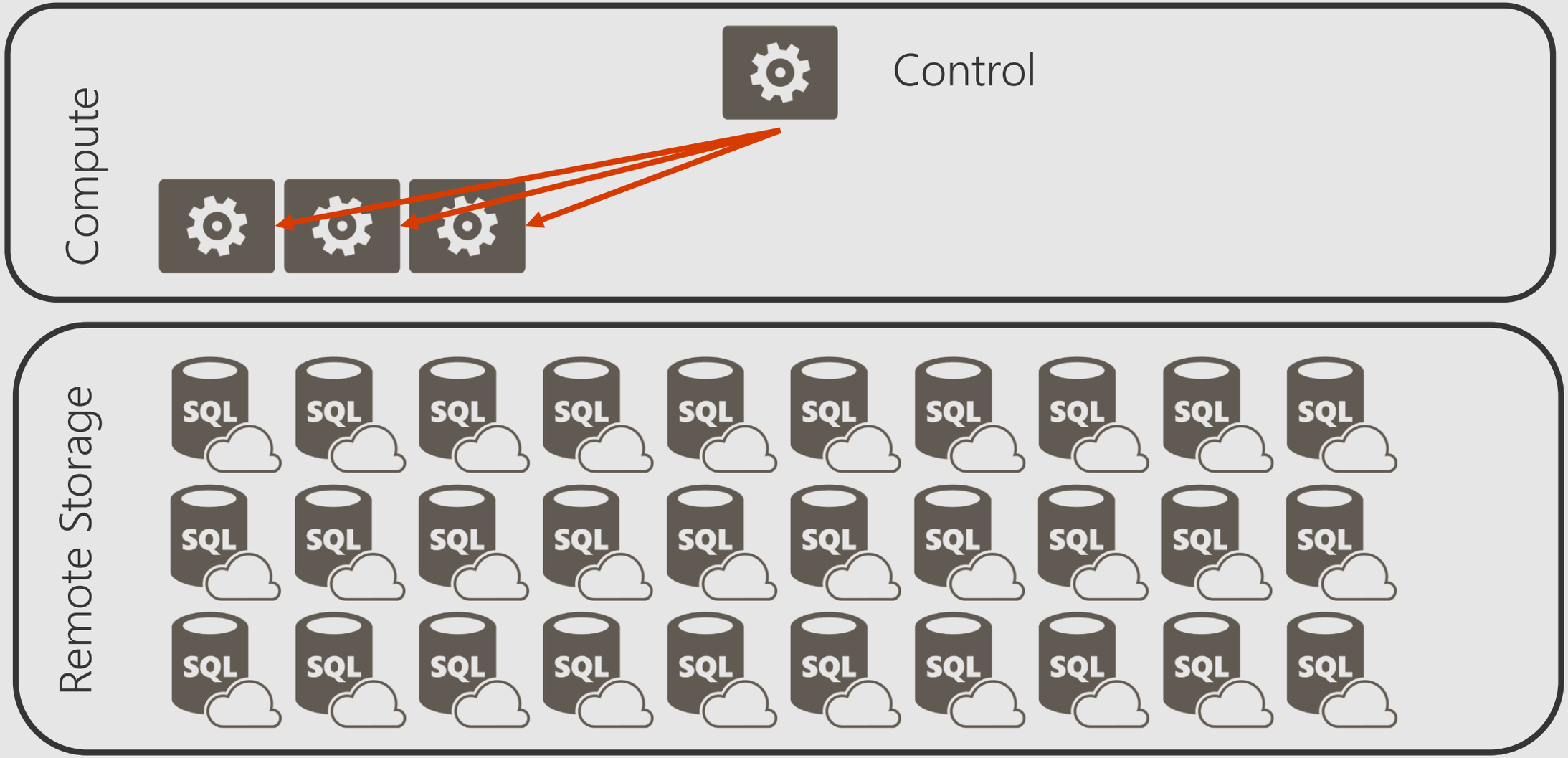



300 DWUs As An MPP diagram



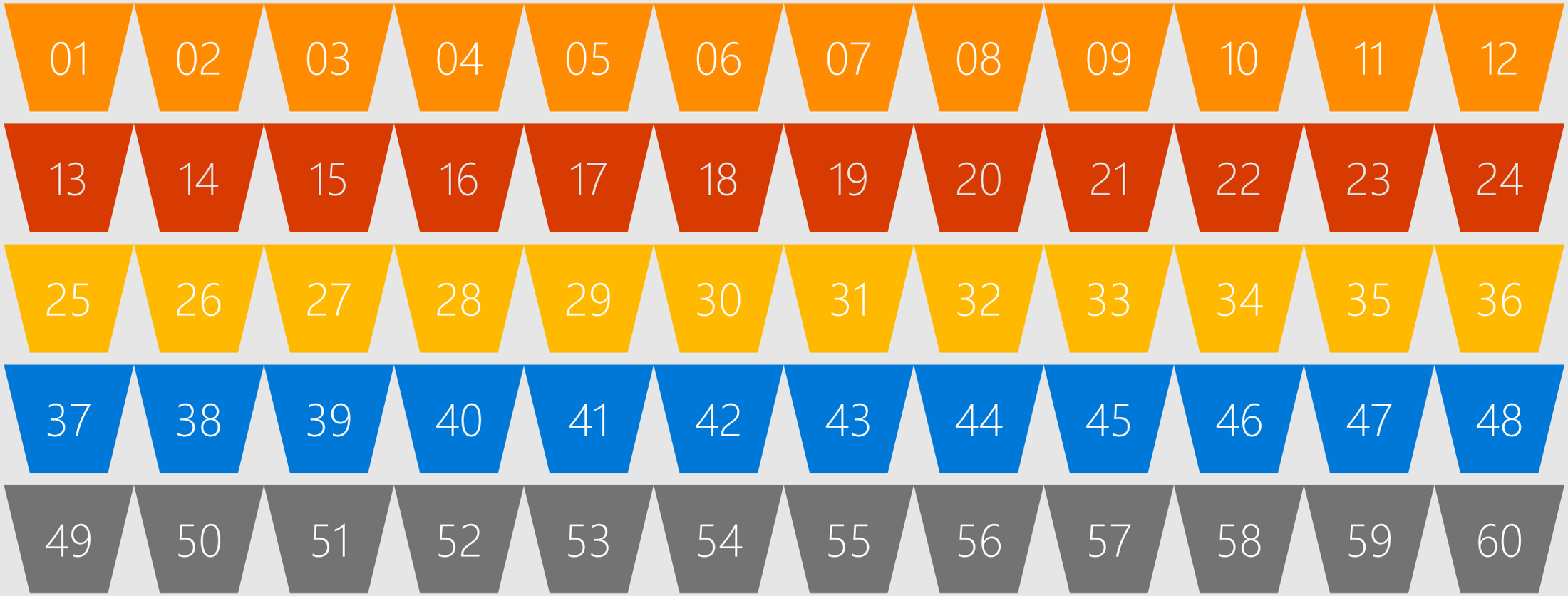

500 DWUs As An MPP diagram

# Pause and resume workload



# Pausing compute in SQLDW

DW500



# Pausing compute in SQLDW

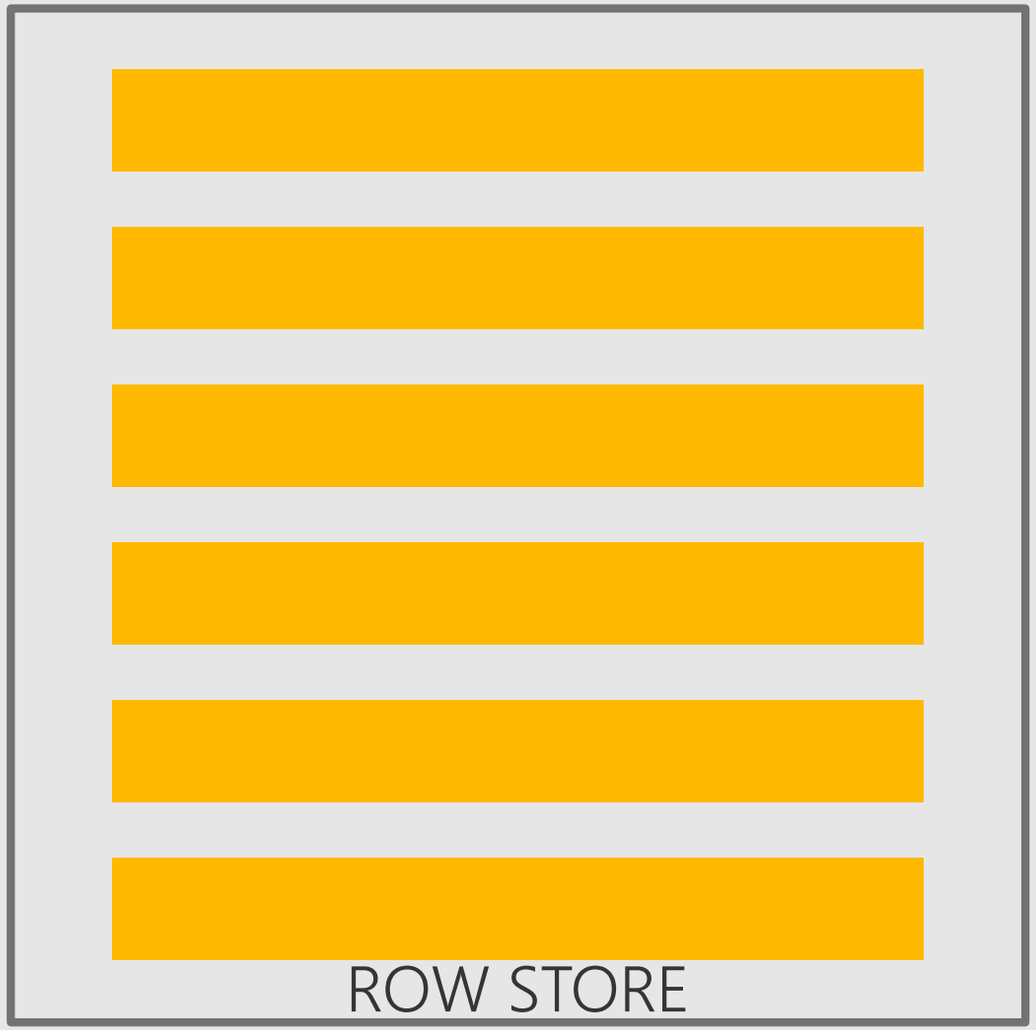
```
Suspend-AzureRmSqlDatabase
    –ResourceGroupName "ResourceGroup"
    –ServerName "Server"
    –DatabaseName "Database"
```

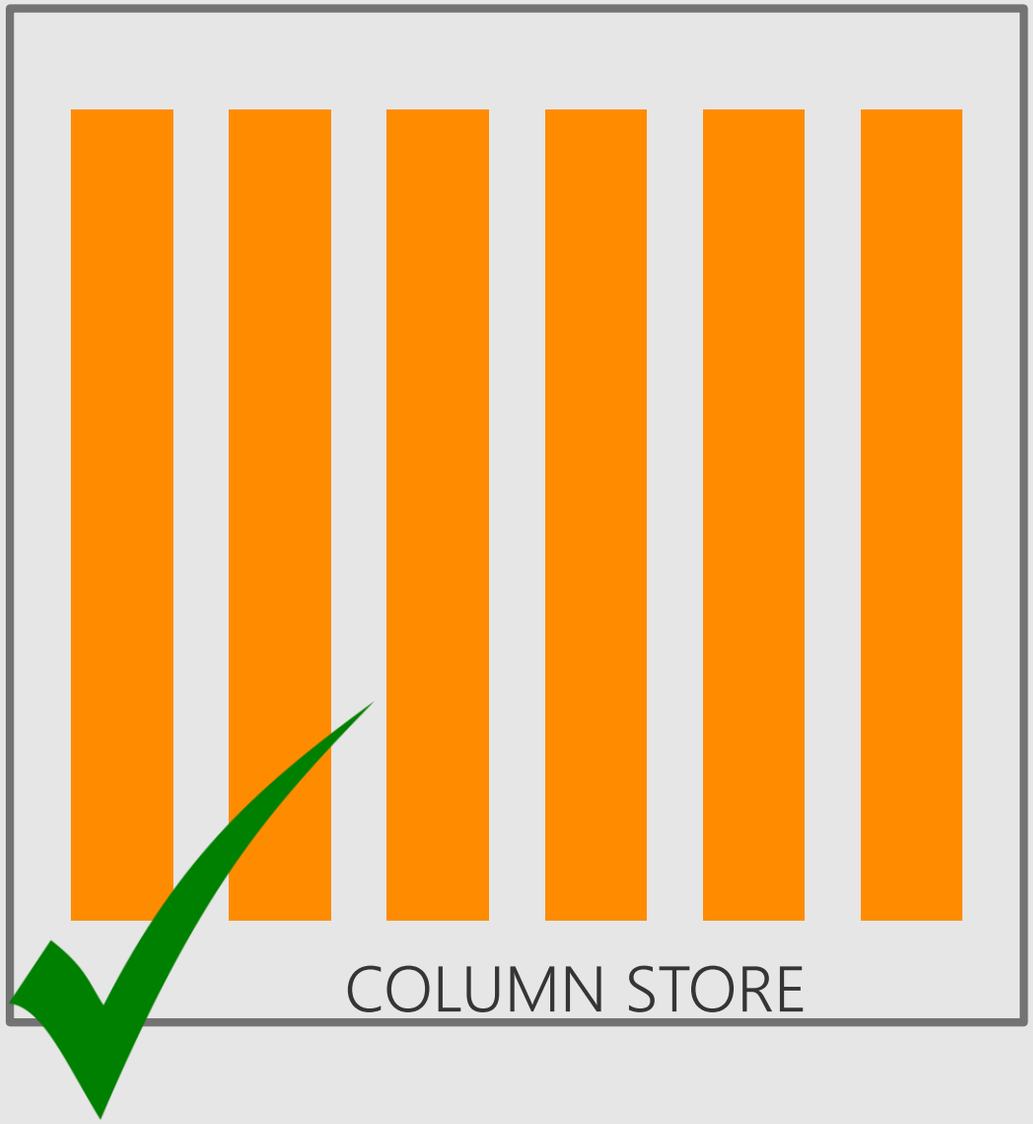
```
Resume-AzureRmSqlDatabase
    –ResourceGroupName "ResourceGroup"
    –ServerName "Server"
    –DatabaseName "Database"
```

*PowerShell*

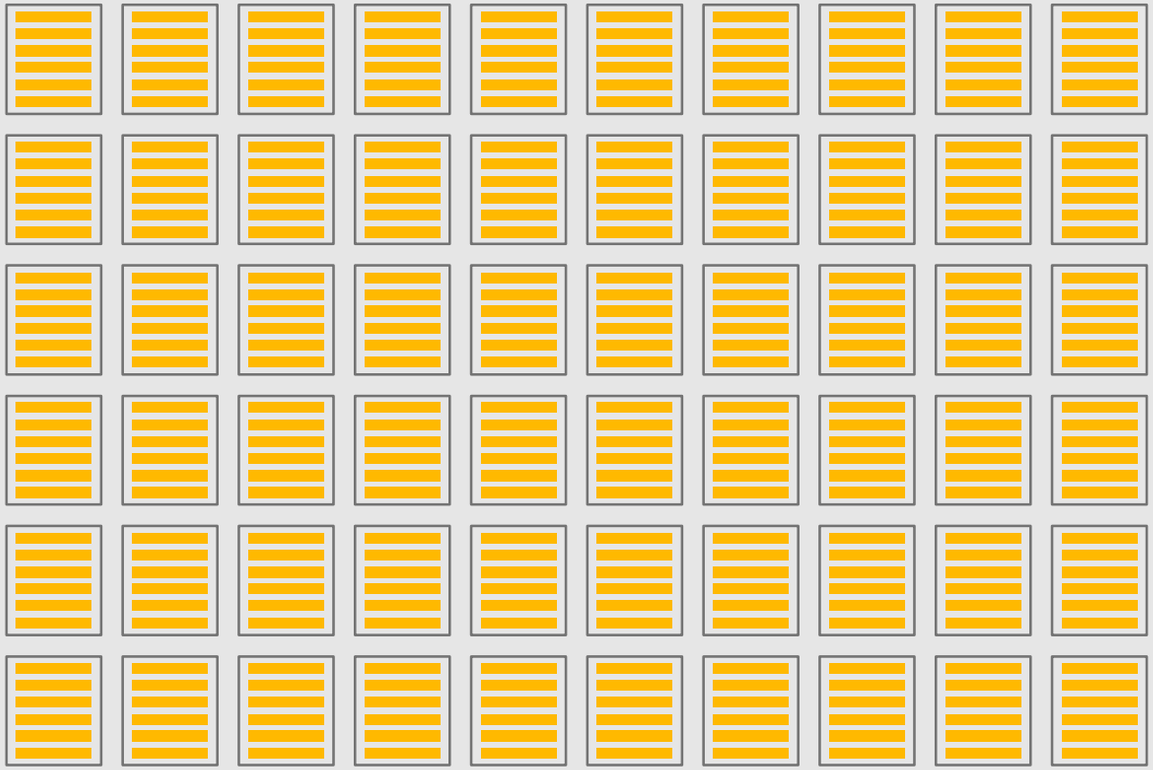
# Table Storage

# Table Storage: Row store & Column store



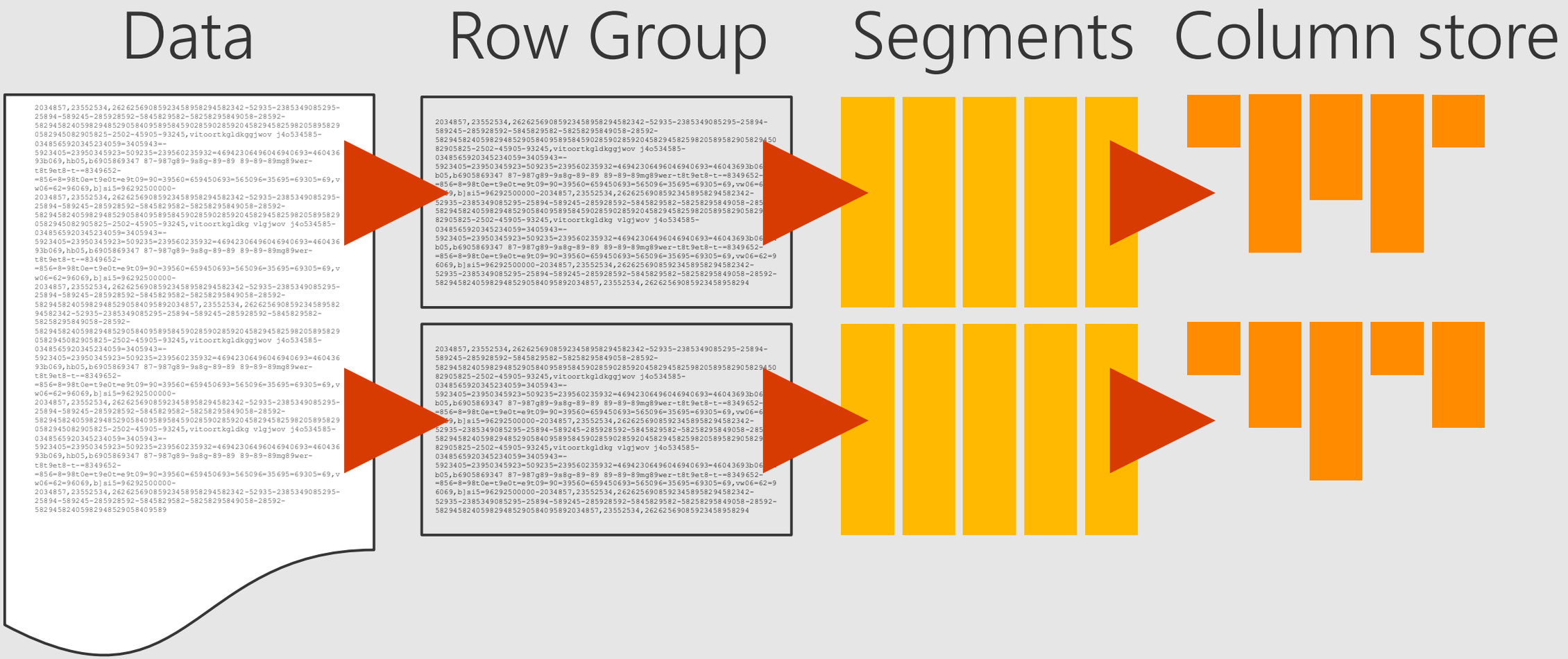
# Scaling out: Impact of distributions on tables



ROW STORE

COLUMN STORE

# Column store



# Why does ColumnStore help?

| Name | Species |
|---|---|
| Hedwig | Owl |
| Mrs. Norris | Cat |
| Crookshanks | Cat |
| Pigwidgeon | Owl |
| Nagini | Snake |
| Errol | Owl |
| Scabbers | Rat |
| Fawkes | Phoenix |
| Hermes | Owl |
| Fluffy | Dog |

| Name | Species |
|---|---|
| Mrs. Norris | Cat |
| Crookshanks | Cat |
| Fluffy | Dog |
| Hedwig | Owl |
| Pigwidgeon | Owl |
| Errol | Owl |
| Hermes | Owl |
| Fawkes | Phoenix |
| Scabbers | Rat |
| Nagini | Snake |

| Name | Species |
|---|---|
| Mrs. Norris: 1 | Cat: 2 |
| Crookshanks: 1 | Dog: 1 |
| Fluffy: 1 | Owl: 4 |
| Hedwig: 1 | Phoenix: 1 |
| Pigwidgeon: 1 | Rat: 1 |
| Errol: 1 | Snake: 1 |
| Hermes: 1 | |
| Fawkes: 1 | |
| Scabbers: 1 | |
| Nagini: 1 | |

Run Length Encoding

# Indexes

## Primary Indexing

Heap = Base Row Store

Clustered Index (CI) = Base Row Store maintained as a B-Tree

Clustered Columnstore Index (CCI) = Base Column Store

## Secondary Indexing

Non Clustered Index (NCI) = Secondary B-Tree Index

NCI can be on Heap

NCI can be on Clustered Columnstore Index (NCI on CCI)

# Business Continuity and Disaster Recovery

# Storage Snapshots

## System RPO

8 hours

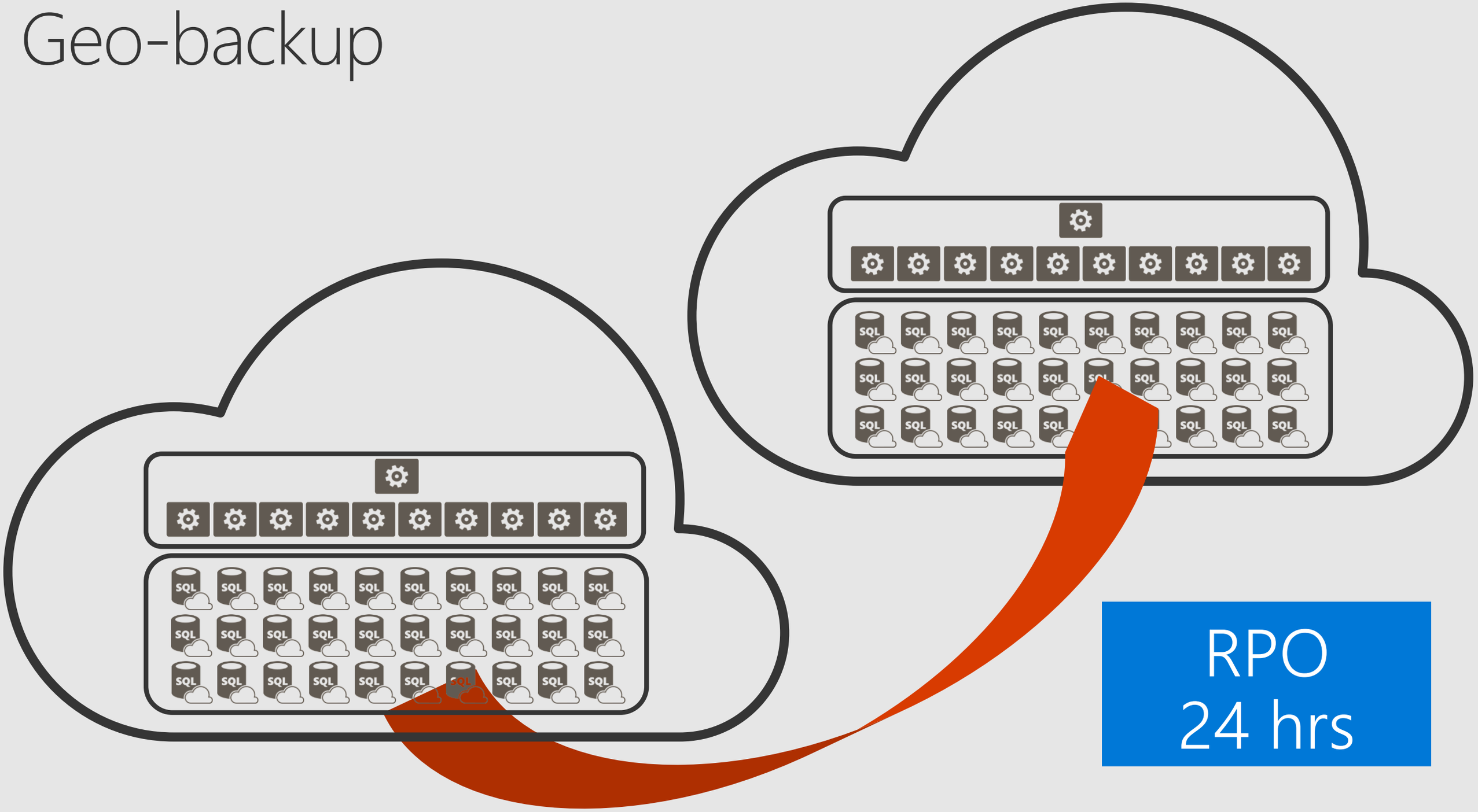
## Snapshot frequency

Every few hours (generally 4)

## Snapshot retention

7 days

# Geo-backup



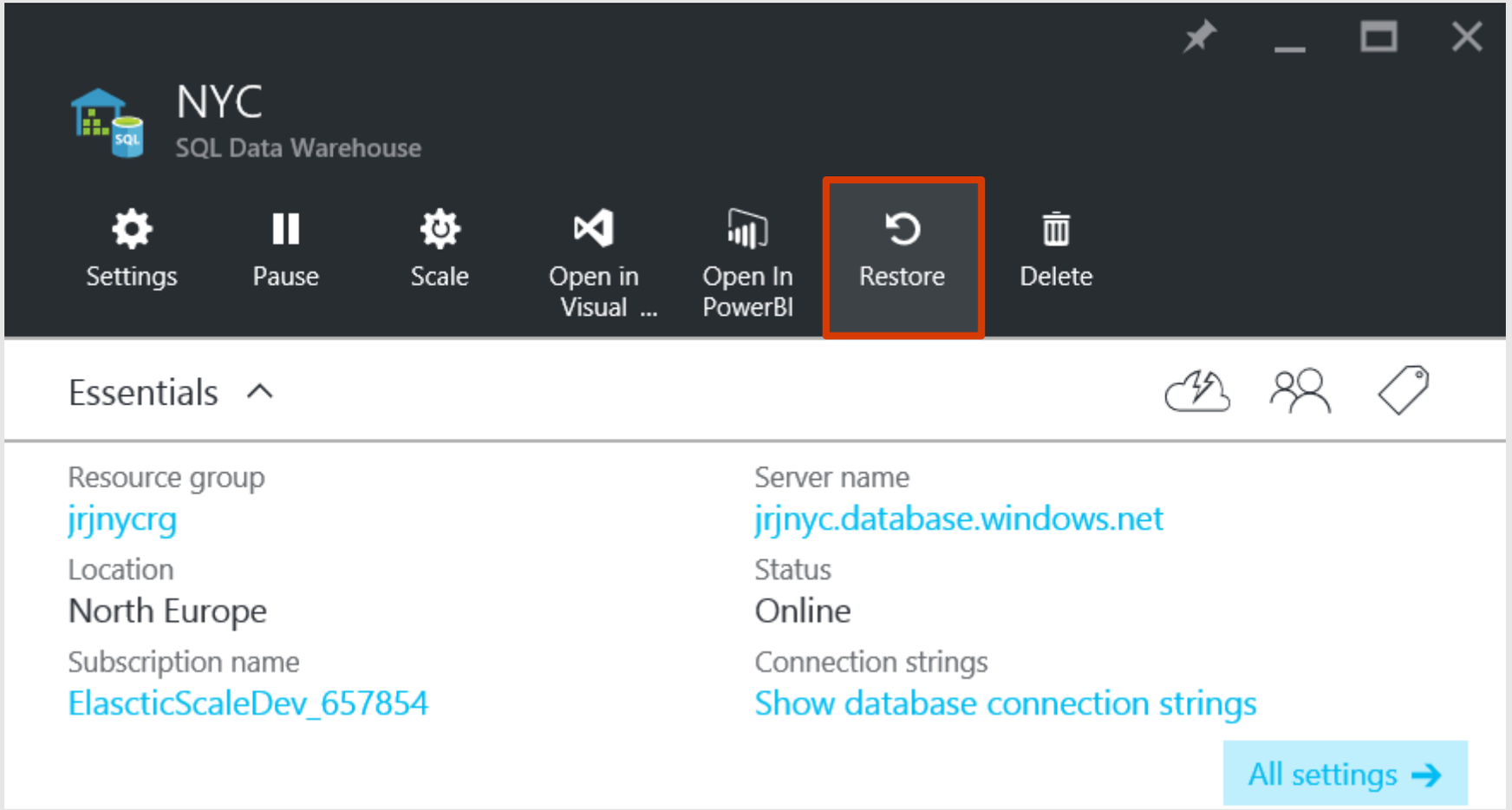
# Restore

Same server overwrite (fastest)

In-region

Geo restore

# Restoring in the Portal



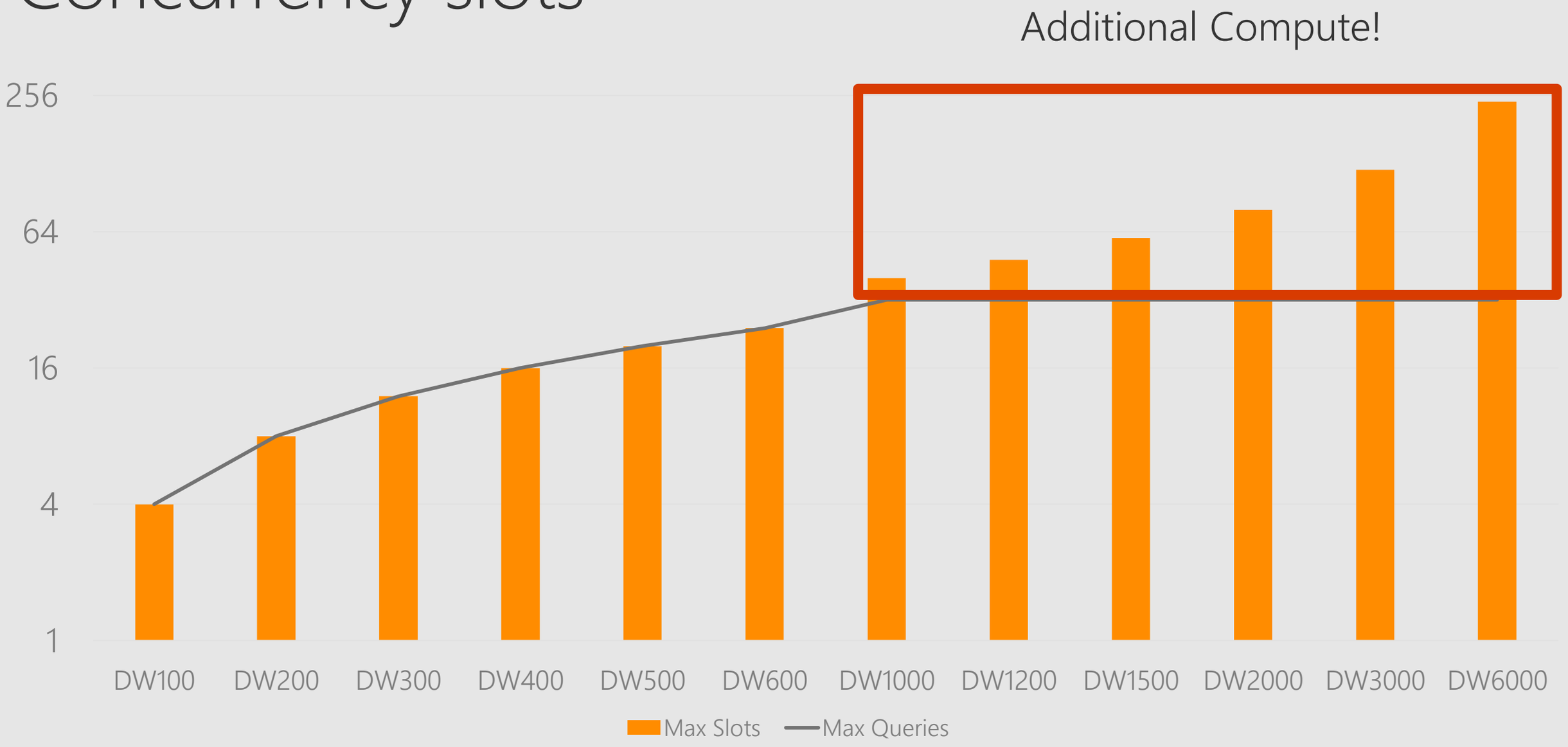
# Snapshots

```
SELECT  [run_id]                    AS bkup_run_id
,       [session_id]                AS session_id
,       [request_id]                AS request_id
,       [name]                      AS bkup_name
,       [submit_time]               AS bkup_submit_time
,       [start_time]                AS bkup_start_time
,       [end_time]                  AS bkup_end_time
,       [total_elapsed_time]        AS bkup_duration_ms
,       [total_elapsed_time]/1000.0 AS bkup_duration_sec
FROM    sys.pdw_loader_backup_runs
;
```

# Workload Management

# Concurrent queries

# Concurrency slots



# Resource classes

## Dynamic

- Increases resource consumption as you scale
- No increase in concurrency as you scale

## Static

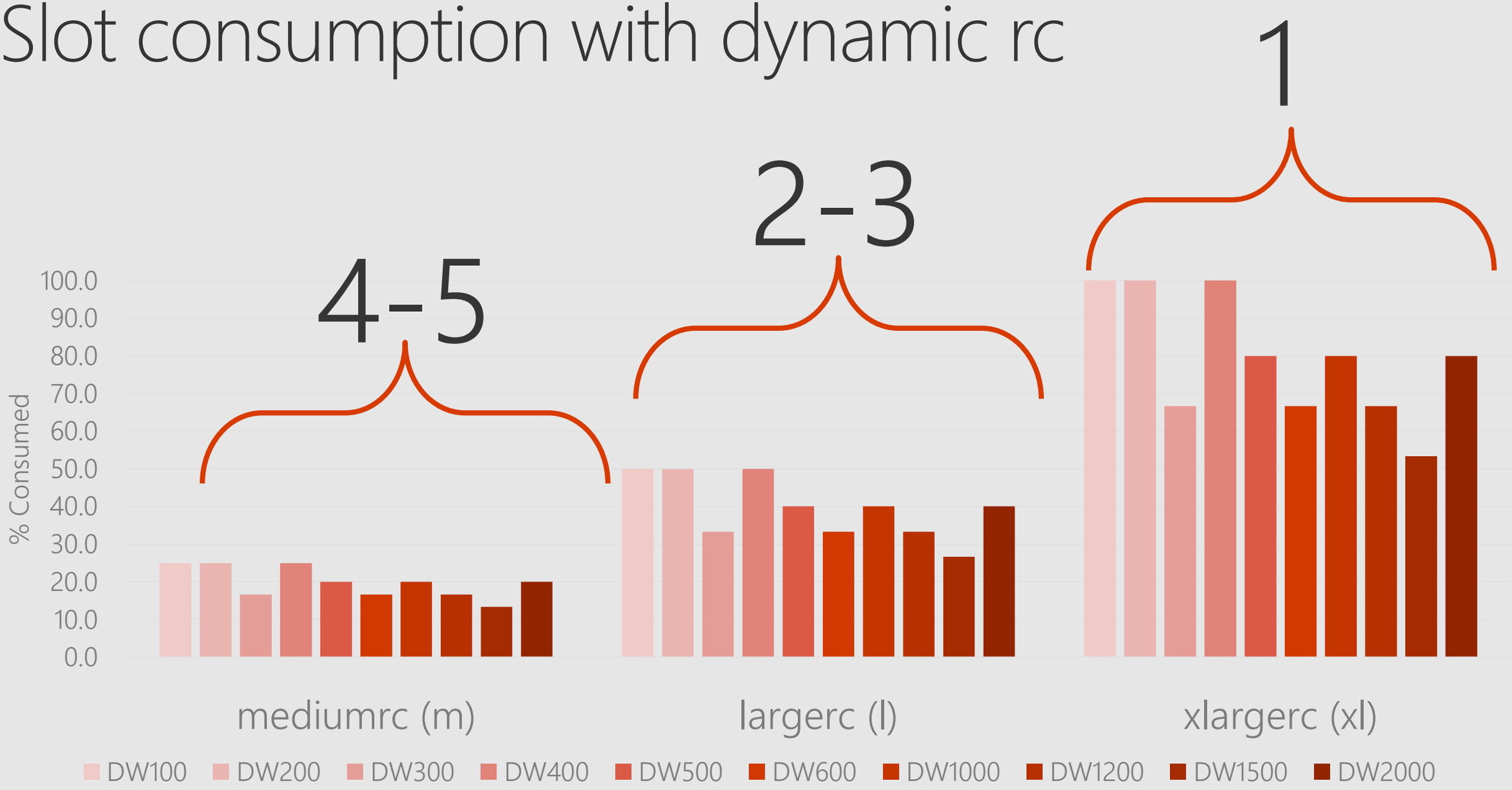
- Maintain resource consumption as you scale
- Increase concurrent queries as you scale

| Consume Slots | Increase memory<br>Isolate resources |
| --- | --- |

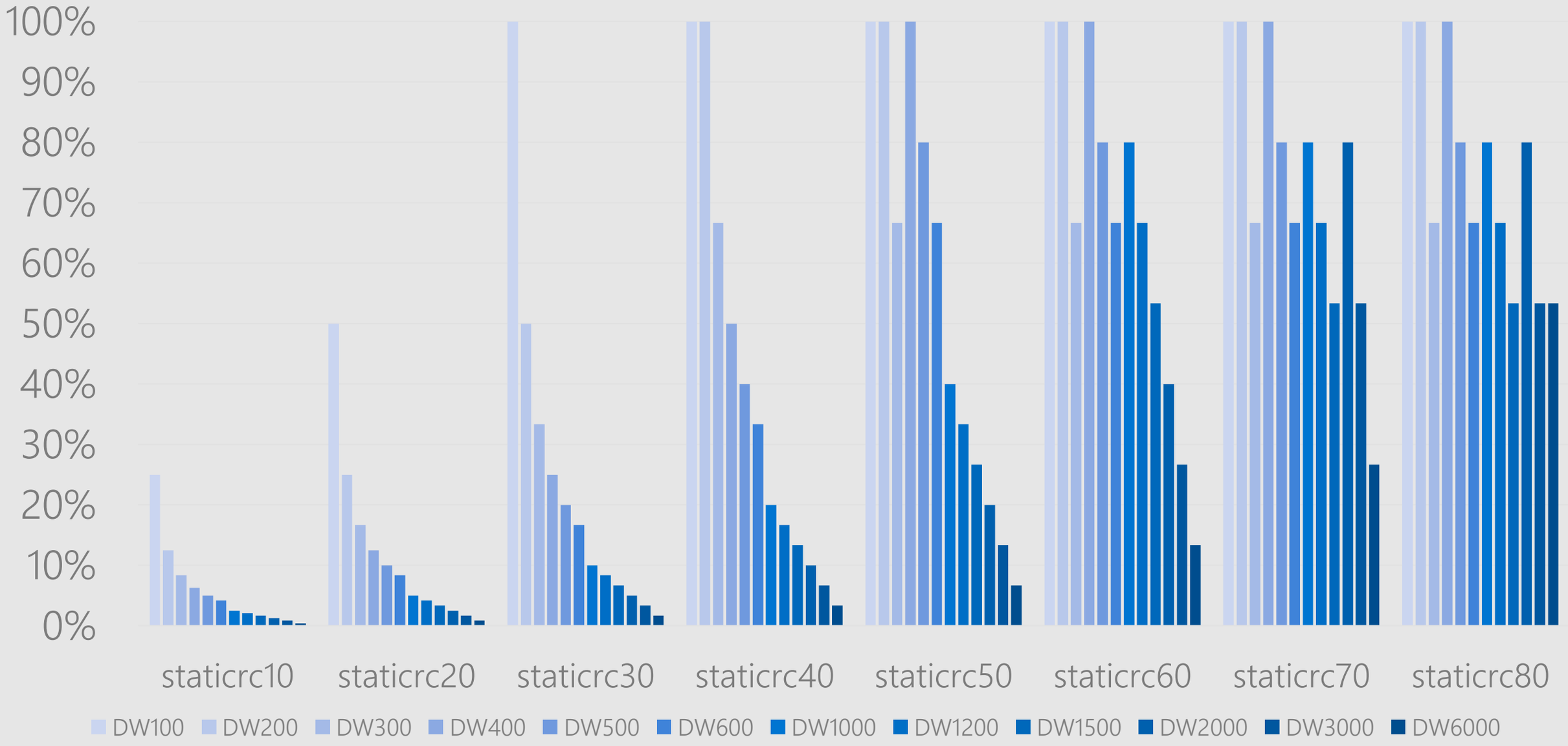
# Resource classes

```
-- Pre-req
CREATE USER data_loader FOR LOGIN data_loader
;
-- Add user to static rc
EXEC sp_addrolemember 'staticrc40','data_loader'
;
-- Remove user from static rc
EXEC sp_droprolemember 'staticrc40','data_loader'
;
```

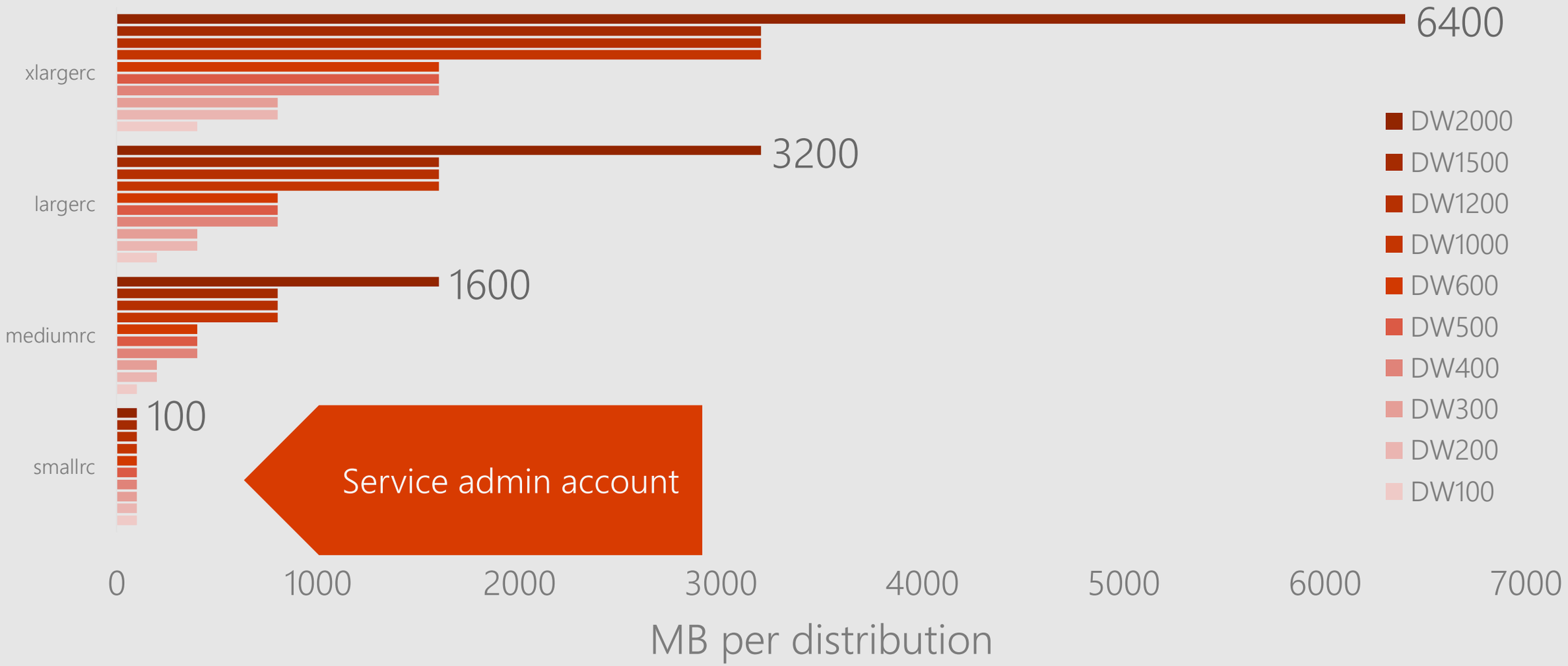
# Slot consumption with dynamic rc



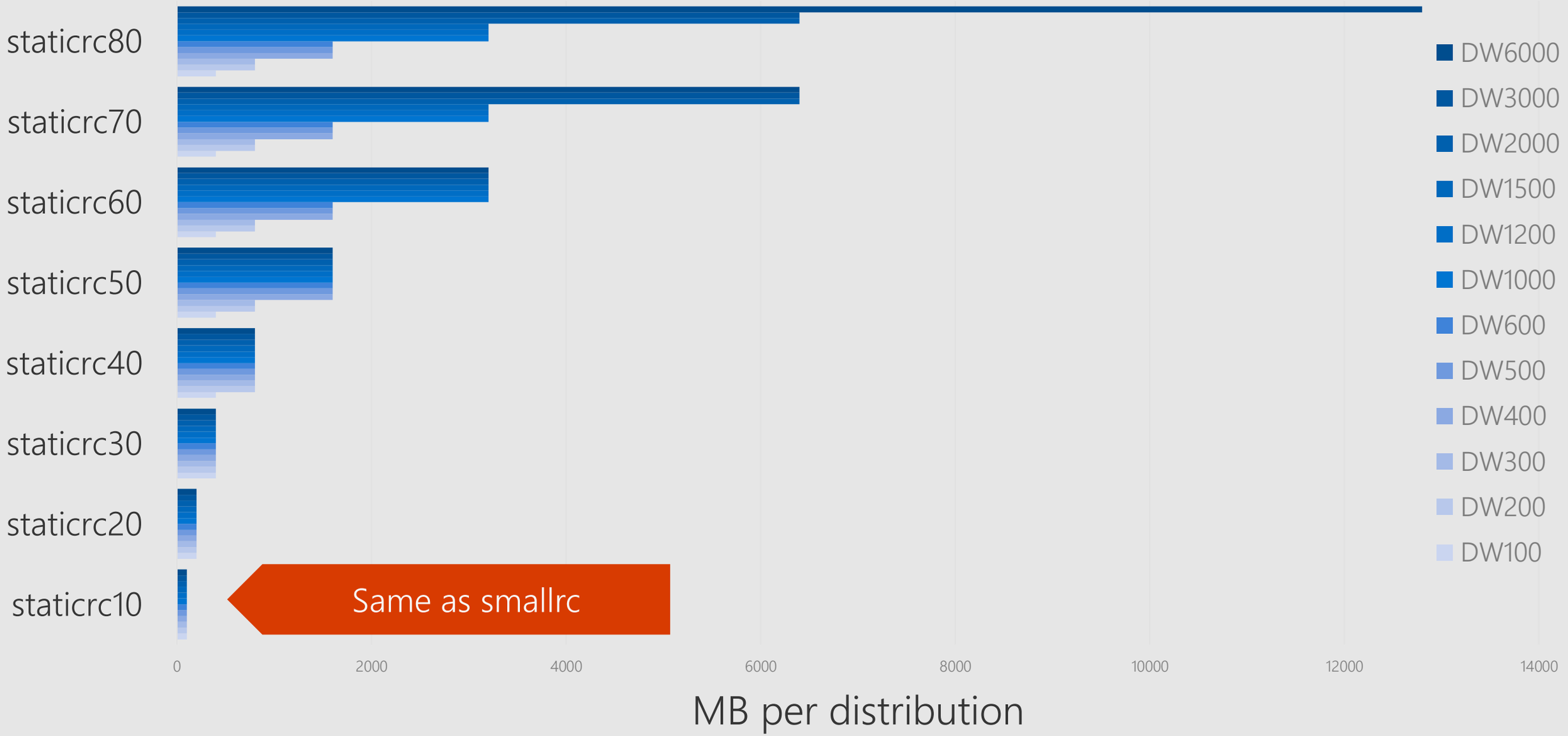
# Slot consumption with static rc



# Memory Allocation for dynamic rc



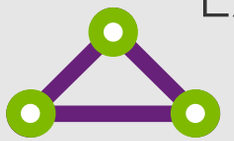
# Memory allocation for staticrc



# Integration with other services

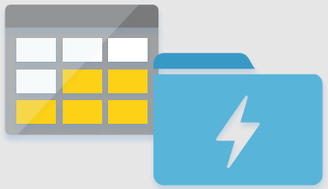
# Common Integration points

Data Factory



ExpressRoute

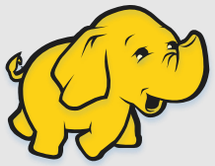
Azure Active Directory



Blob / ADLS

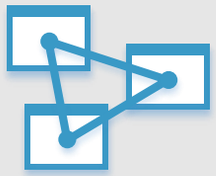


SQLDB



HDInsight

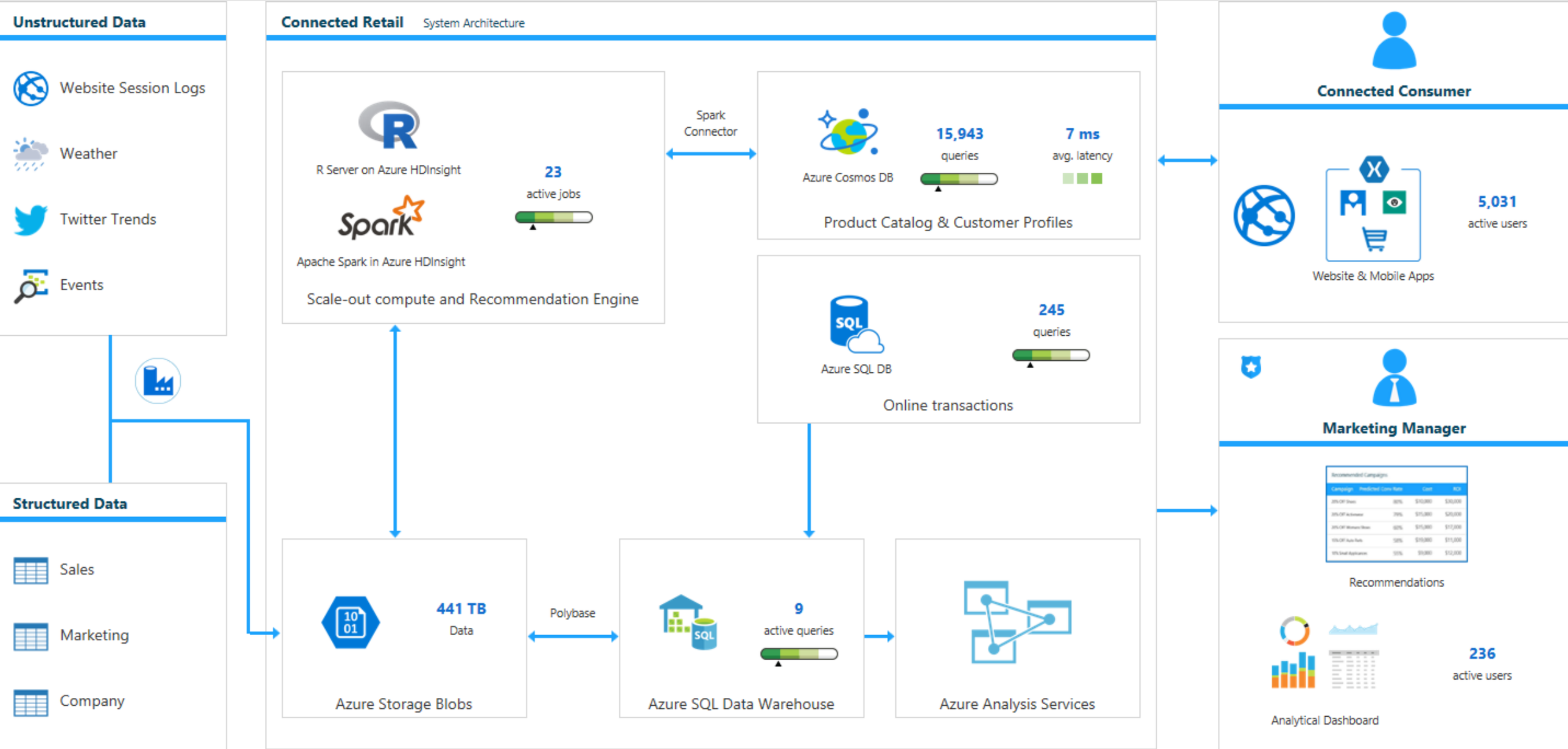
ADLA



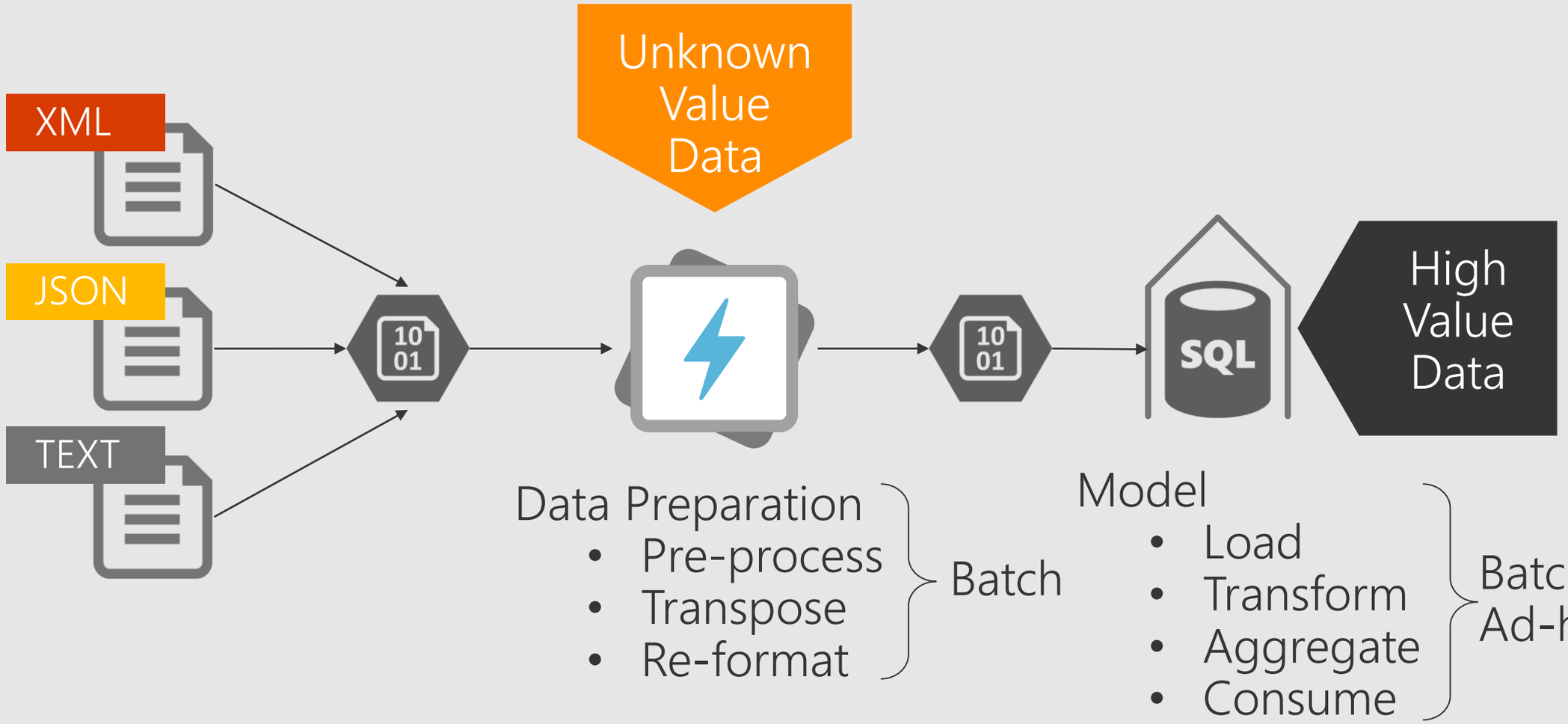
Analysis Services

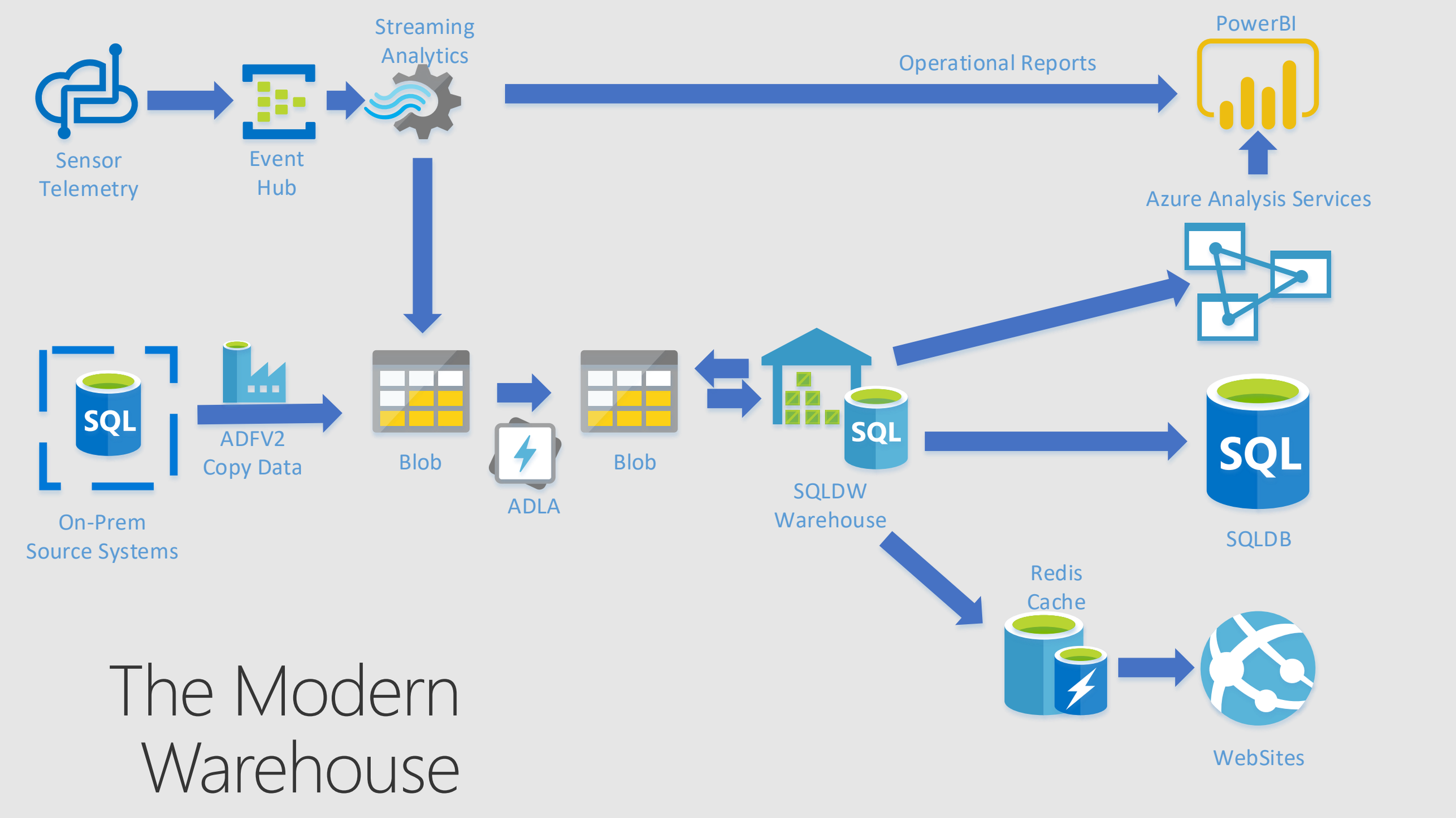
PowerBI

# Cloud Data Warehouse



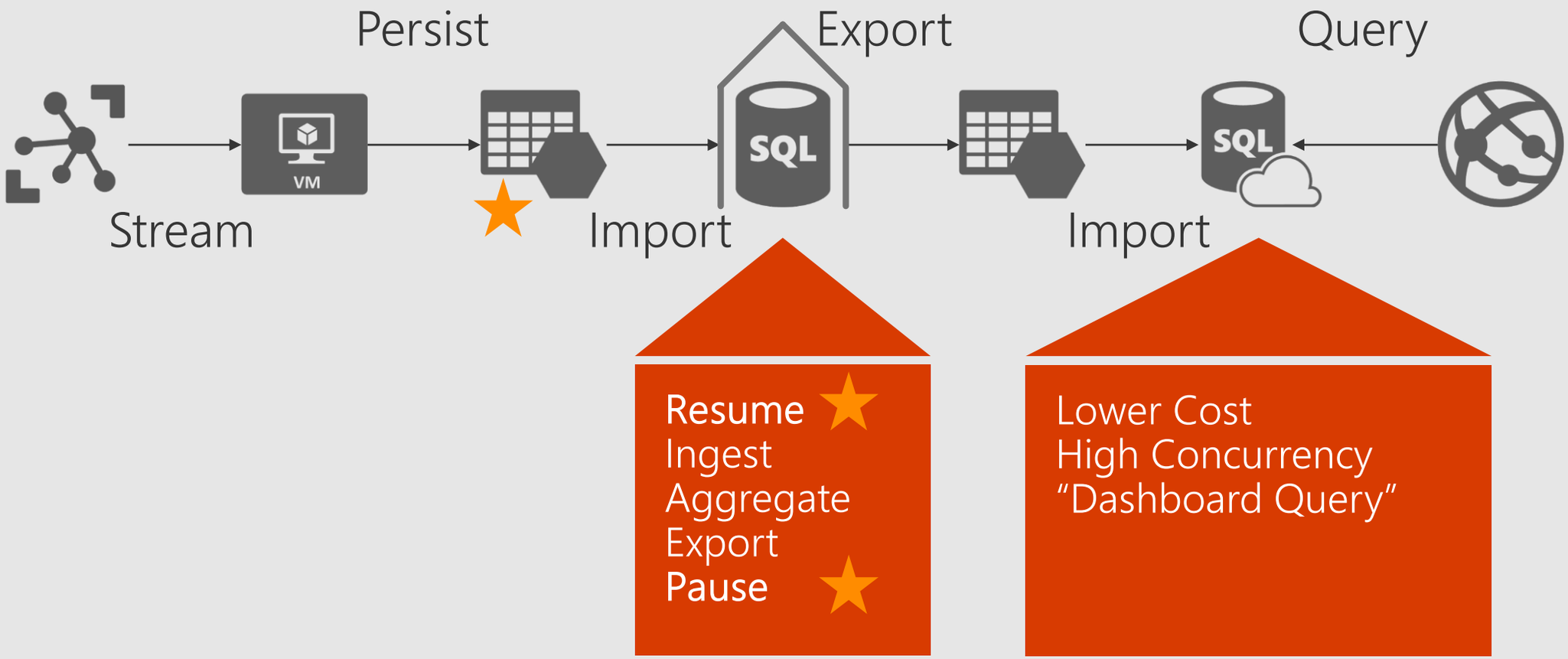
# ADL & SQLDW


Unknown
Value
Data
XML
JSON
TEXT
10
01
10
01
SQL
High
Value
Data
Data Preparation
• Pre-process
• Transpose
• Re-format
Batch
Model
• Load
• Transform
• Aggregate
• Consume
Batch
Ad-hoc

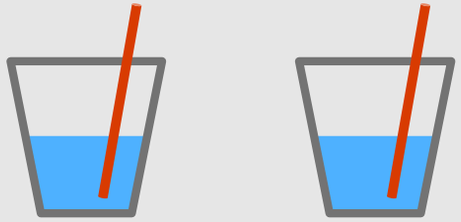



# The Modern Warehouse

# The On-Demand Processor



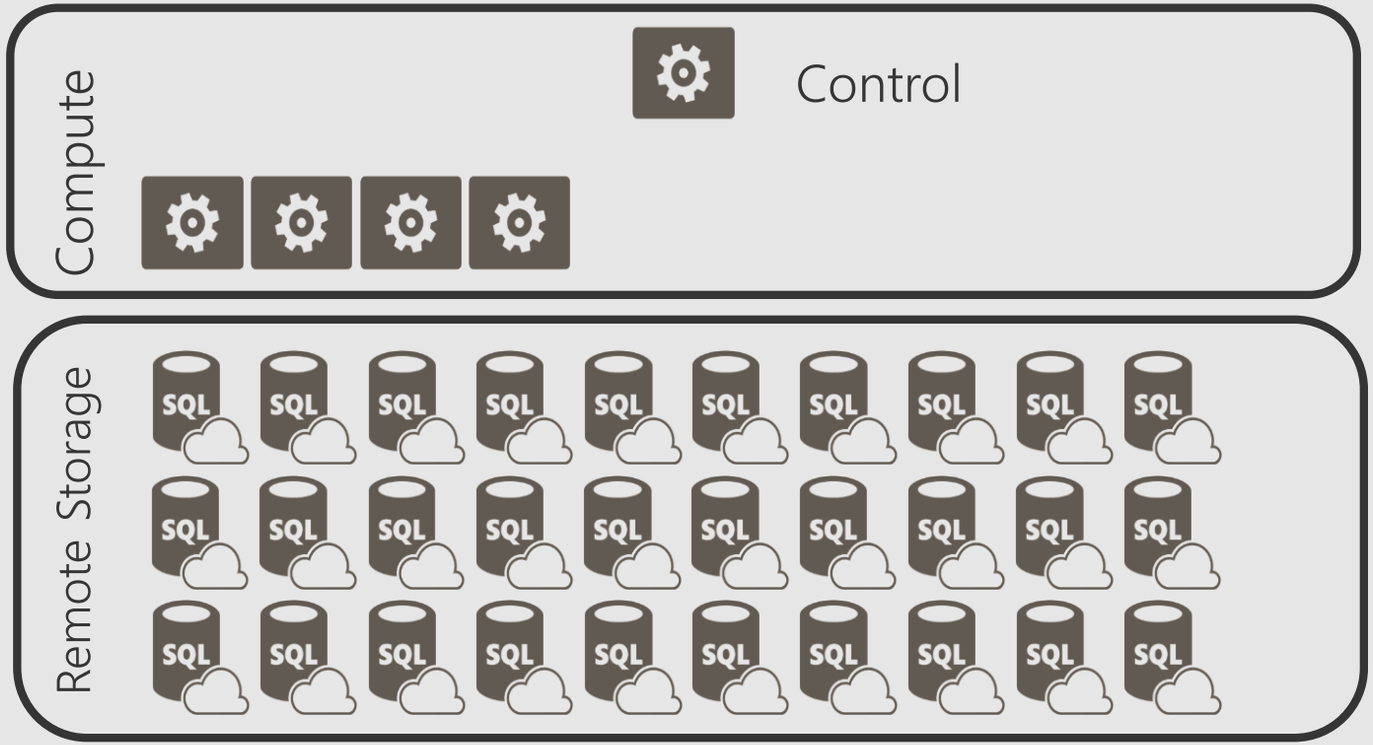
# Summary



# What did we learn?

Scale-out distributed query engine


Fully managed
Completely elastic
Platform as a Service (PaaS)




De-coupled storage from compute

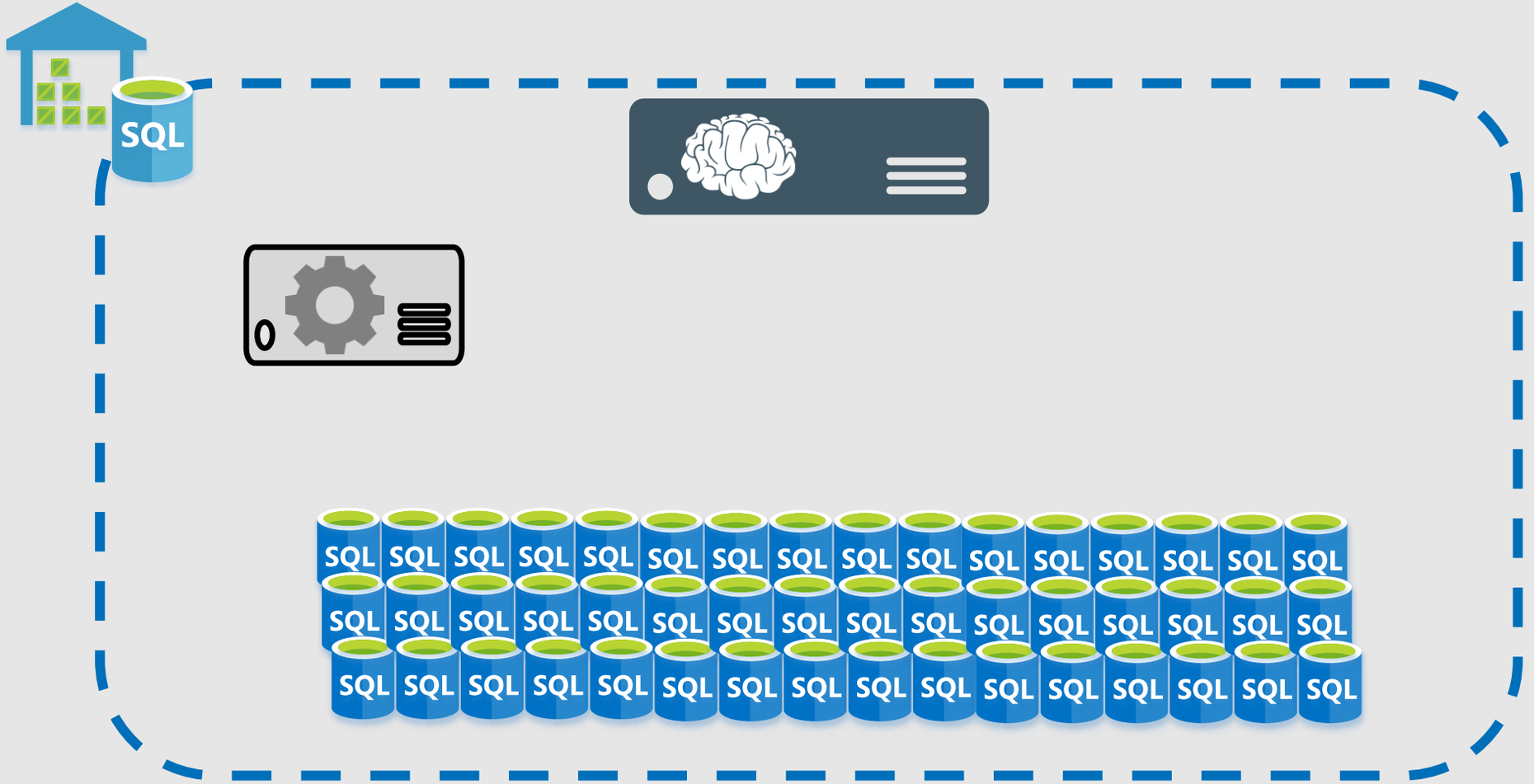
MagicWorks

# Will SQLDW Help?


- Distributed Scale-Out queries will speed up their analytics
- PetaByte scale storage can handle their growth
- Can elastically scale to handle unforeseen circumstances

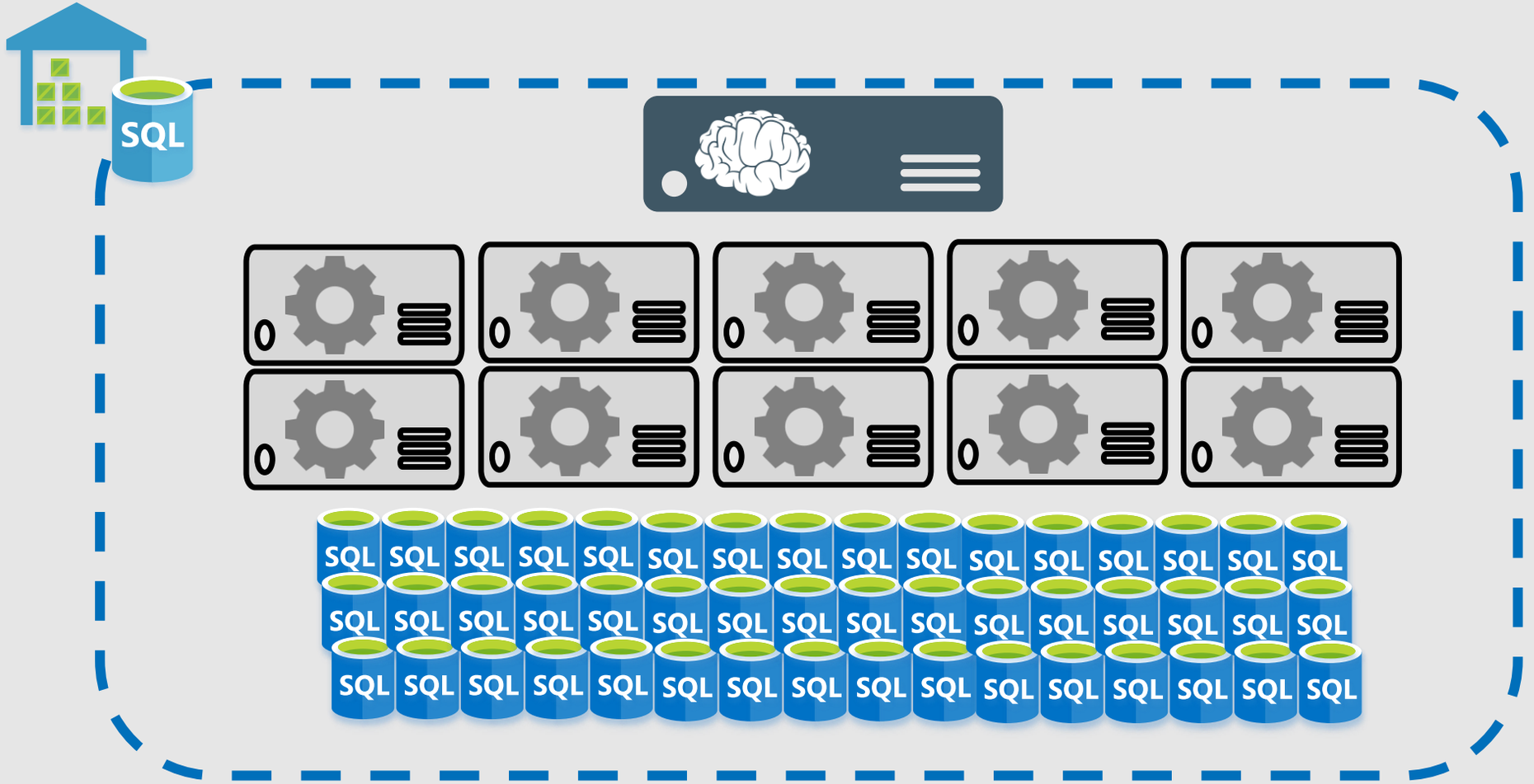
# Azure SQLDW Gen 1

# 100 DWUs

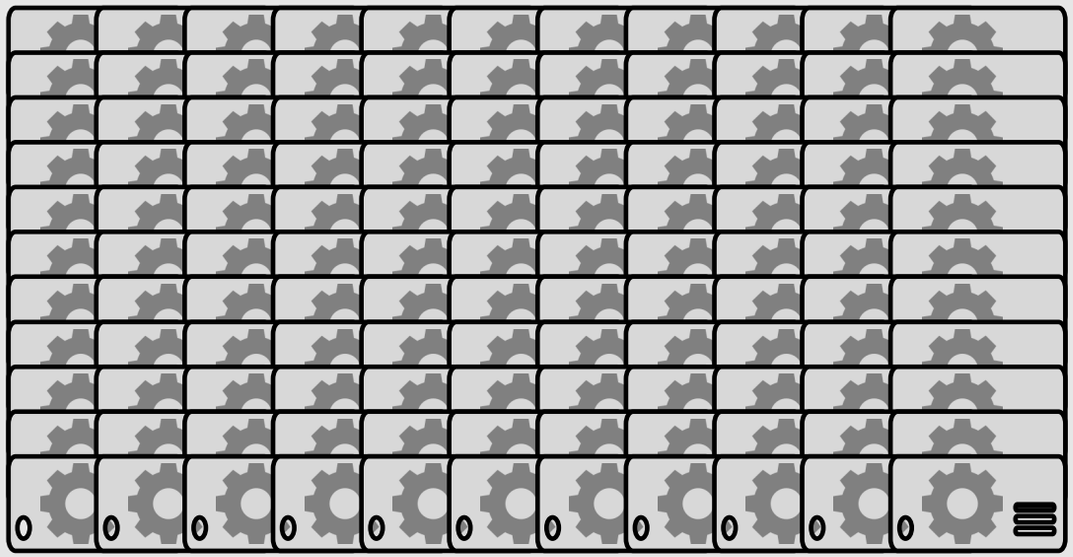

SQL
0
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL

# 100 DWUs = 1 Compute Node

# 1000 DWUs


SQL
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL
SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL SQL

# 6000 DWUs is a 1-to-1 mapping of compute to distributions

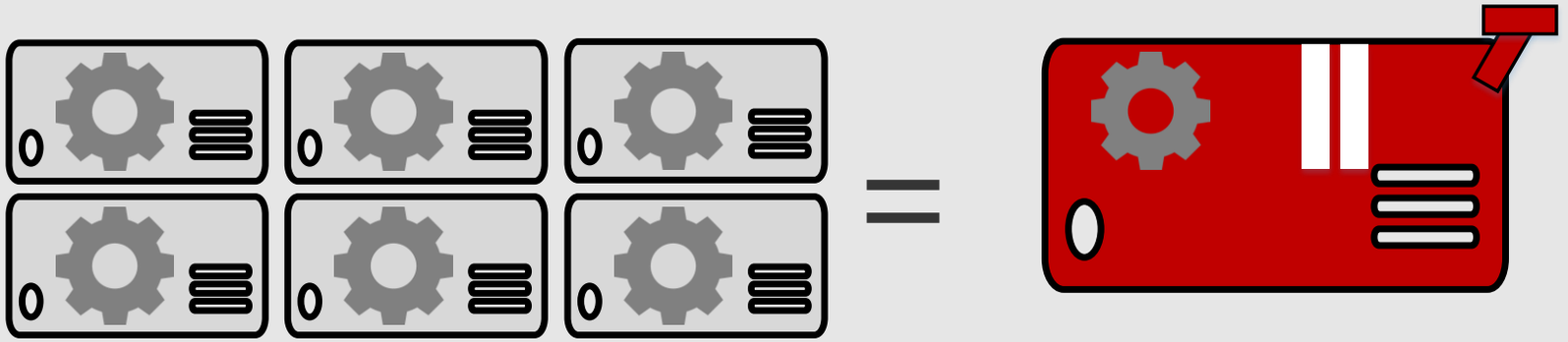


60x


60x

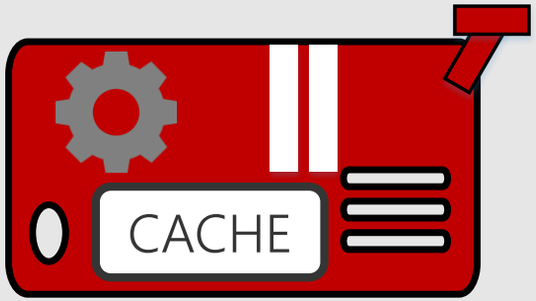
# Azure SQLDW Gen 2

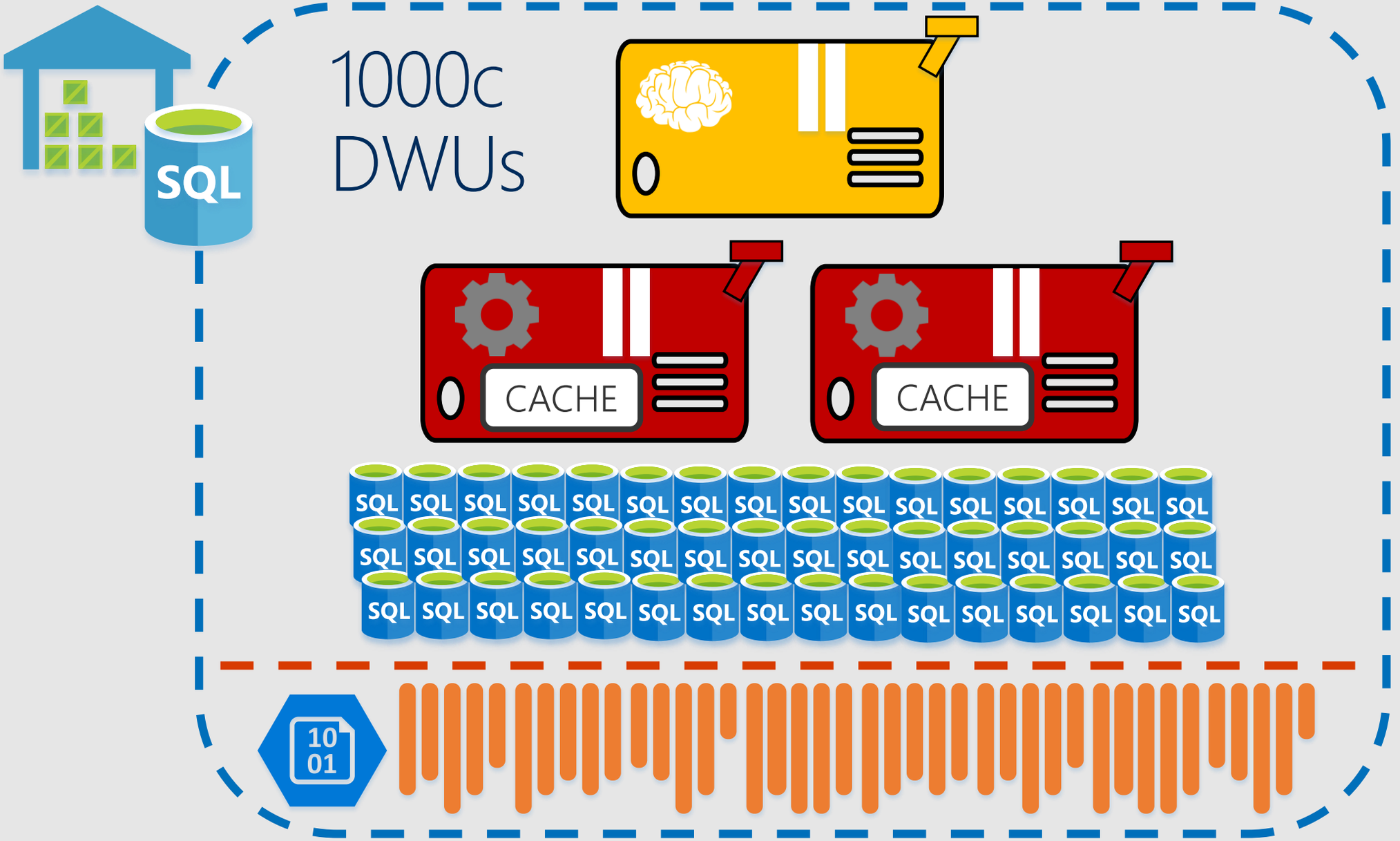
# Node Sizing



# Separate Column Store

# Caching




1000c
DWUs
SQL
CACHE
CACHE
SQL
10
01

| Gen 1 DWU | G1 Nodes | G1 Concurrency | Gen 2 DWU | G2 Nodes | G2 Concurrency |
|---|---|---|---|---|---|
| 100 | 1 | 4 | | | |
| 200 | 2 | 8 | | | |
| 300 | 3 | 12 | | | |
| 400 | 4 | 16 | | | |
| 500 | 5 | 20 | | | |
| 600 | 6 | 24 | | | |
| 1000 | 10 | 32 | 1000c | 2 | 32 |
| 1200 | 12 | 32 | | | |
| 1500 | 15 | 32 | 1500c | 3 | 32 |
| 2000 | 20 | 48 | 2000c | 4 | 48 |
| | | | 2500c | 5 | 48 |
| 3000 | 30 | 64 | 3000c | 6 | 64 |
| | | | 5000c | 10 | 64 |
| 6000 | 60 | 128 | 6000c | 12 | 128 |
| | | | 7500c | 15 | 128 |
| | | | 10000c | 20 | 128 |
| | | | 15000c | 30 | 128 |
| | | | 30000c | 60 | 128 |