Azure SQL Data Warehouse

L300 deck

Paul Duffett (paduffet@Microsoft.com)
Twitter: @paulduffett

Mar2017

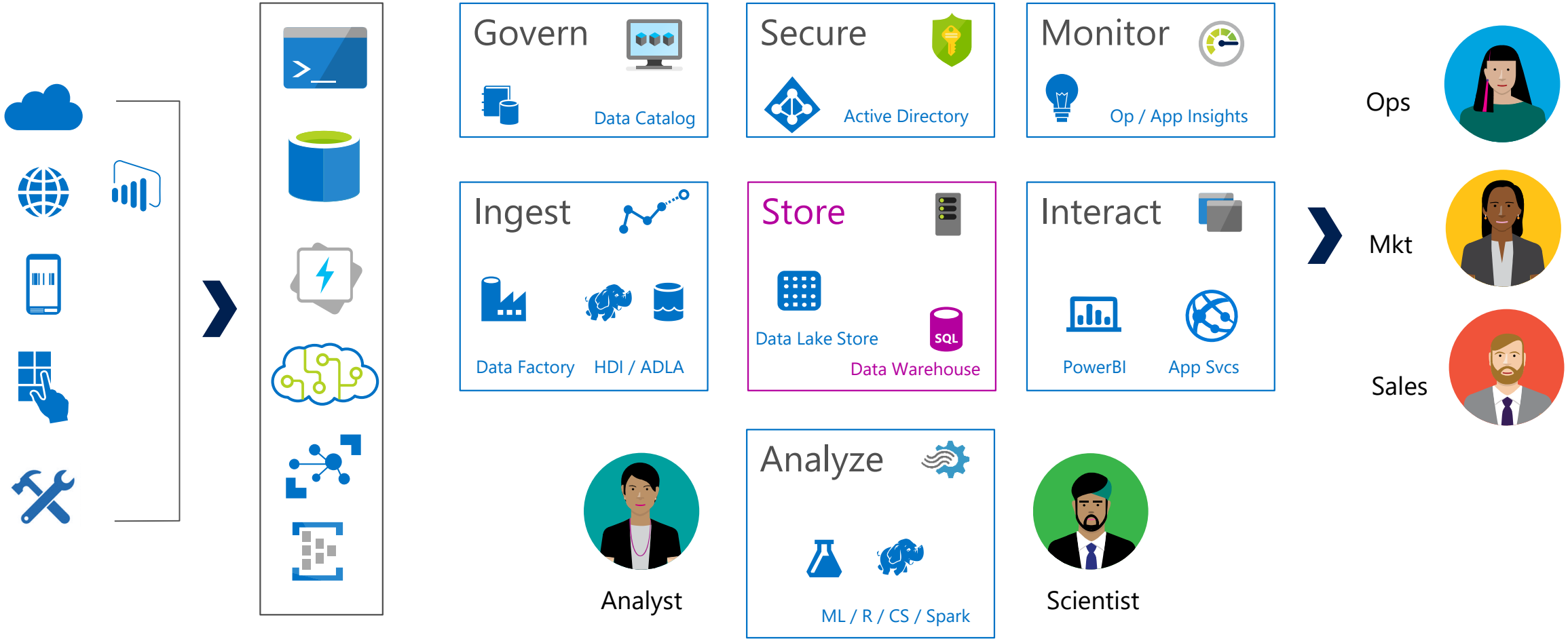
Contents

In presentation mode click on any box to jump to that section

Introduction

Where does a data warehouse fit? Everywhere!

Data & service architecture



Changes in Enterprise Data Warehouse space

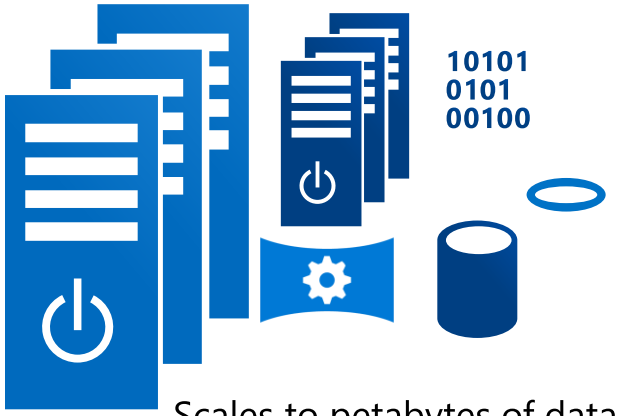
Organizations are changing with increasing demand to:

- Integrate with new or unstructured data
- Drive to the cloud
- Reduce or remove hardware renewal
- Reduction in support costs

Introducing Azure SQL Data Warehouse

A relational **data warehouse-as-a-service**, fully managed by Microsoft.
Industries first **elastic** cloud data warehouse with proven SQL Server capabilities.
Support your **smallest to your largest** data storage needs.

Elastic scale & performance



Scales to petabytes of data

Massively Parallel Processing

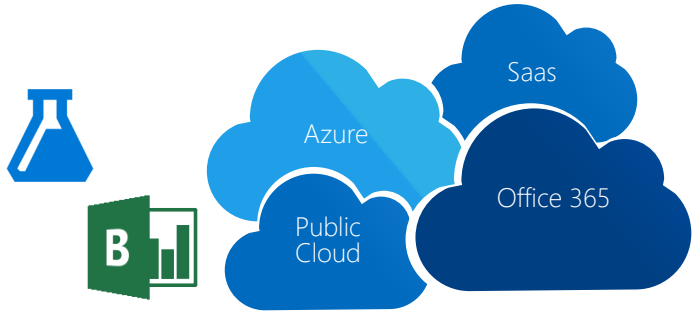
Instant-on compute scales in seconds

Query Relational / Non-Relational

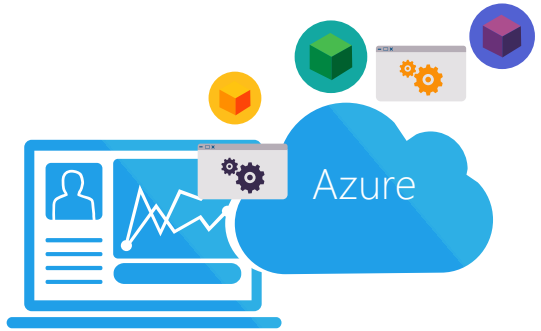
Powered by the Cloud

Get started in minutes

Integrated with Azure ML, PowerBI & ADF



Market Leading Price & Performance

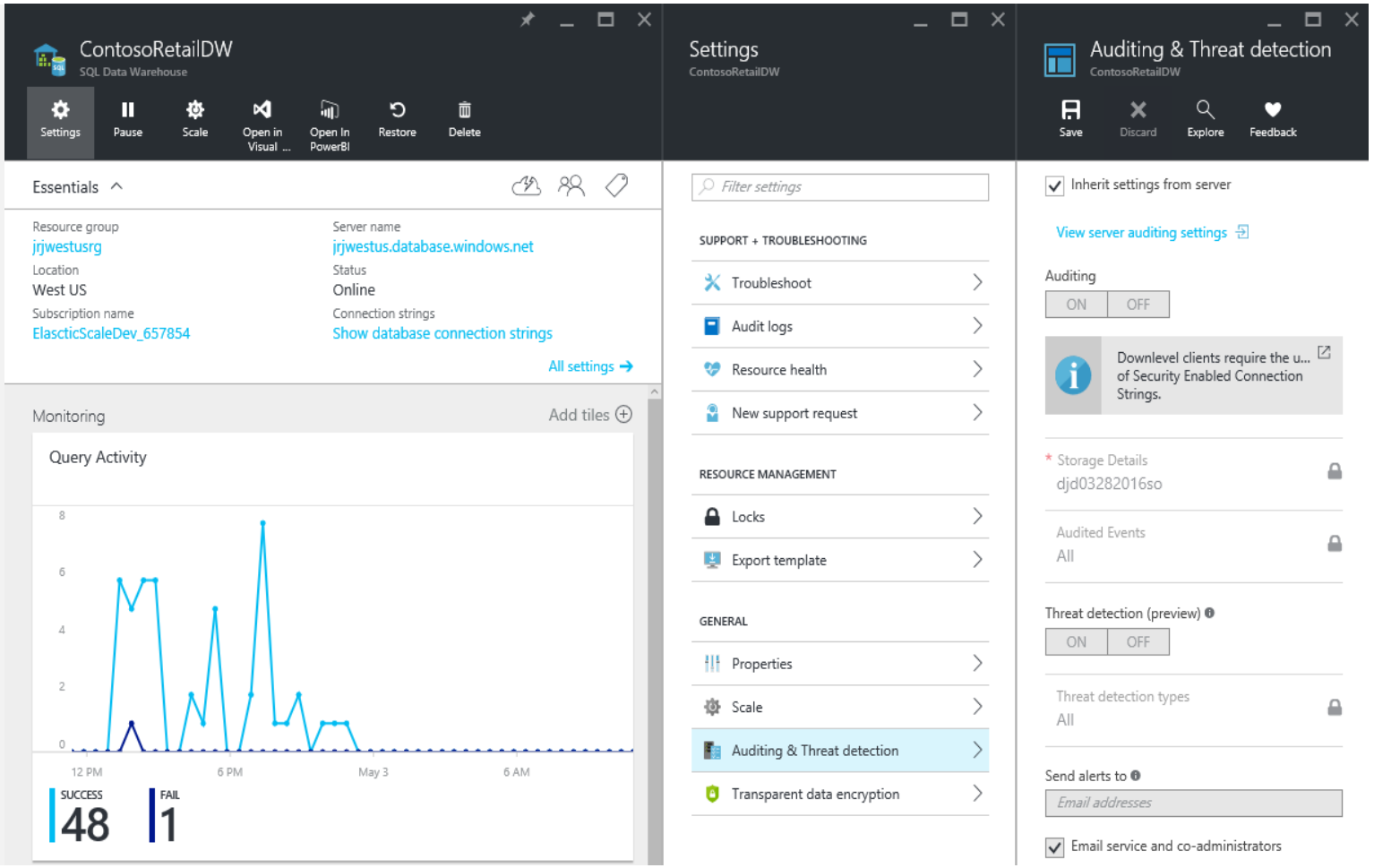


Simple billing compute & storage

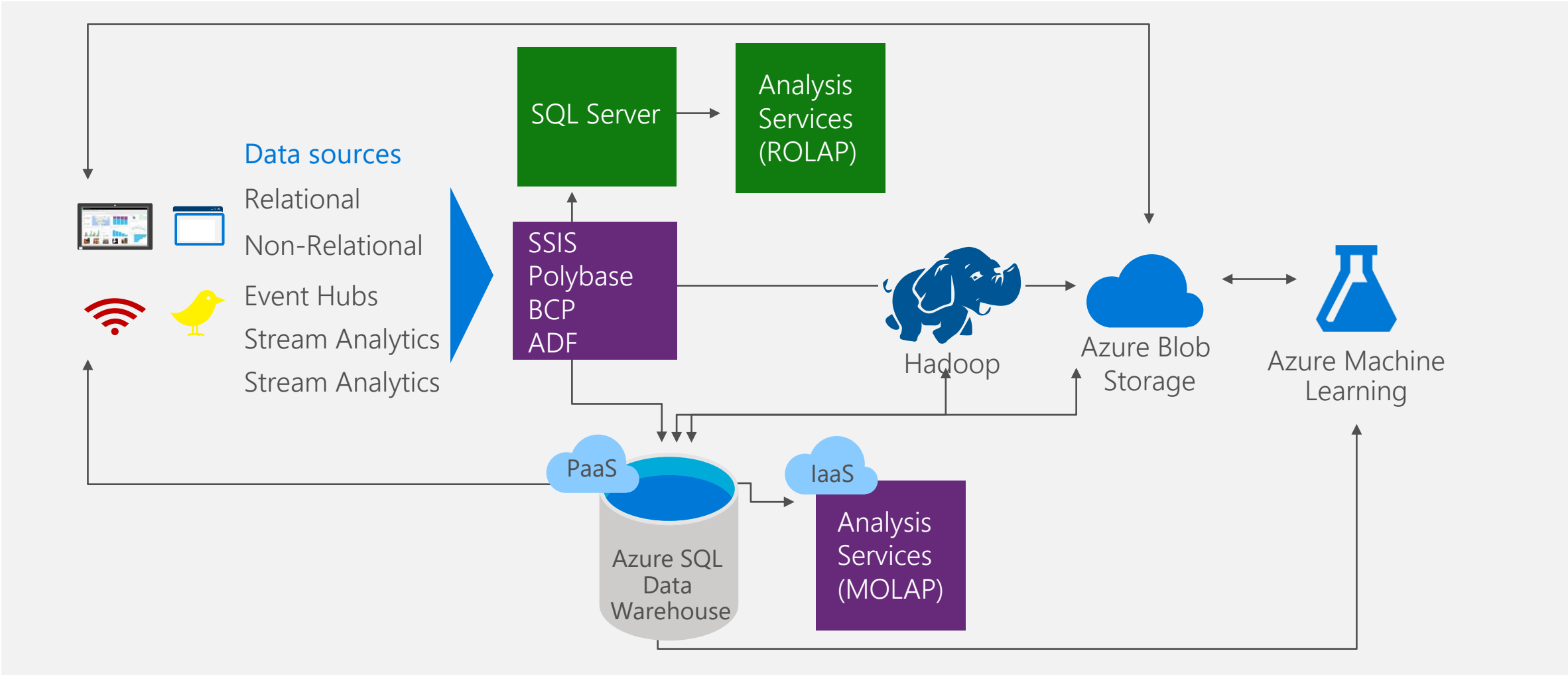
Pay for what you need, when you need it with dynamic pause

A fully managed Platform-as-a-Service

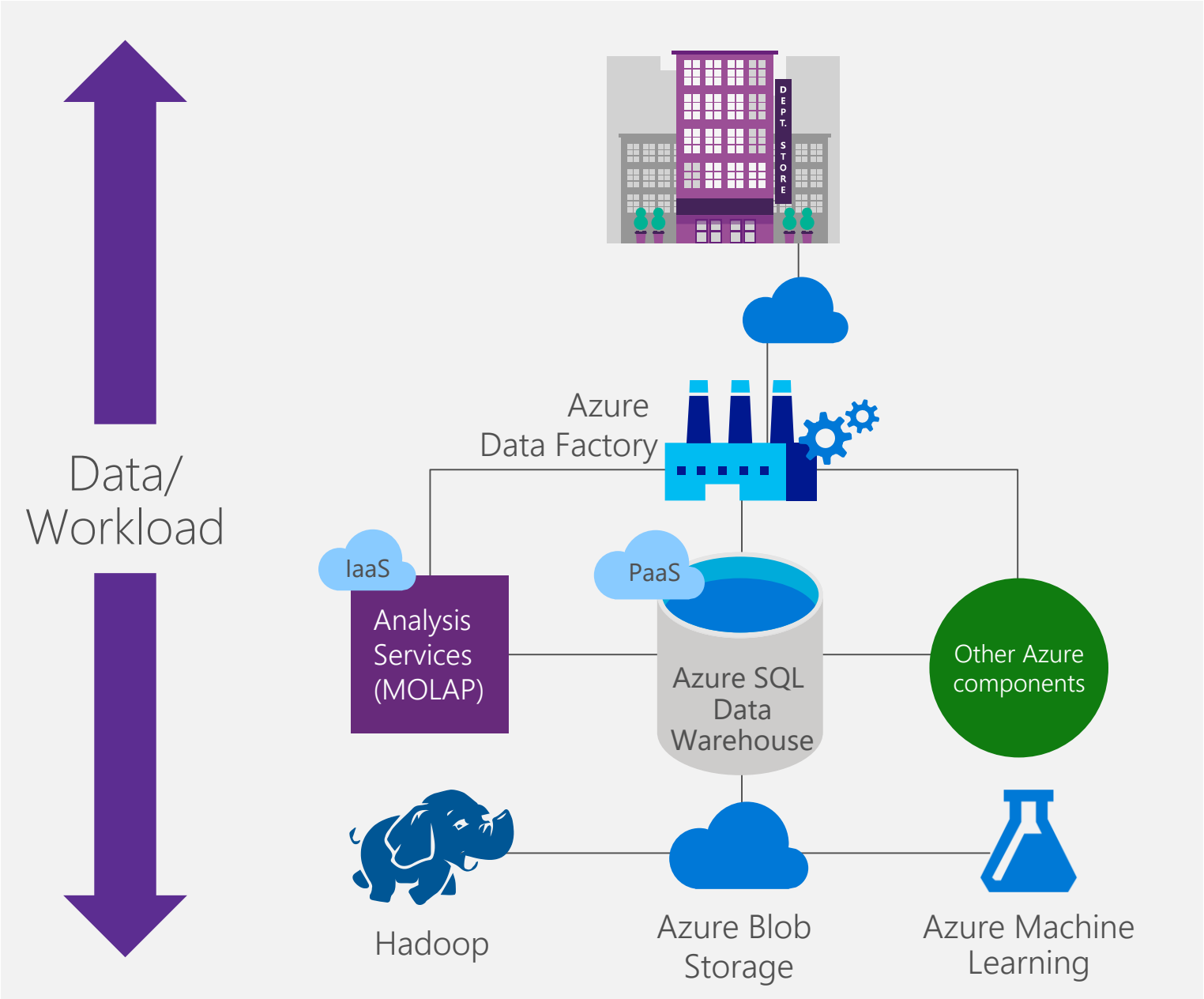
- Azure cloud data warehouse service
- Elastic scale
- Separate storage and compute
- Use existing tools and skills
- Deploy and use in minutes!



Integrates with existing processes



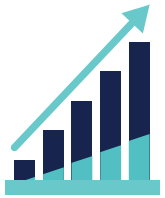
Supports data ingestion from literally anywhere…



Technical capabilities

SQL data warehouse: An elastic solution

Industry's **first** enterprise-class cloud data warehouse that can **grow, shrink, and pause** in seconds

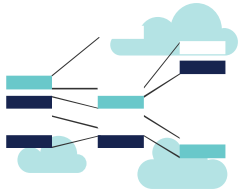


Petabyte scalability with massive parallel processing

Full enterprise-class SQL Server experience

Independent scale of compute and storage in seconds

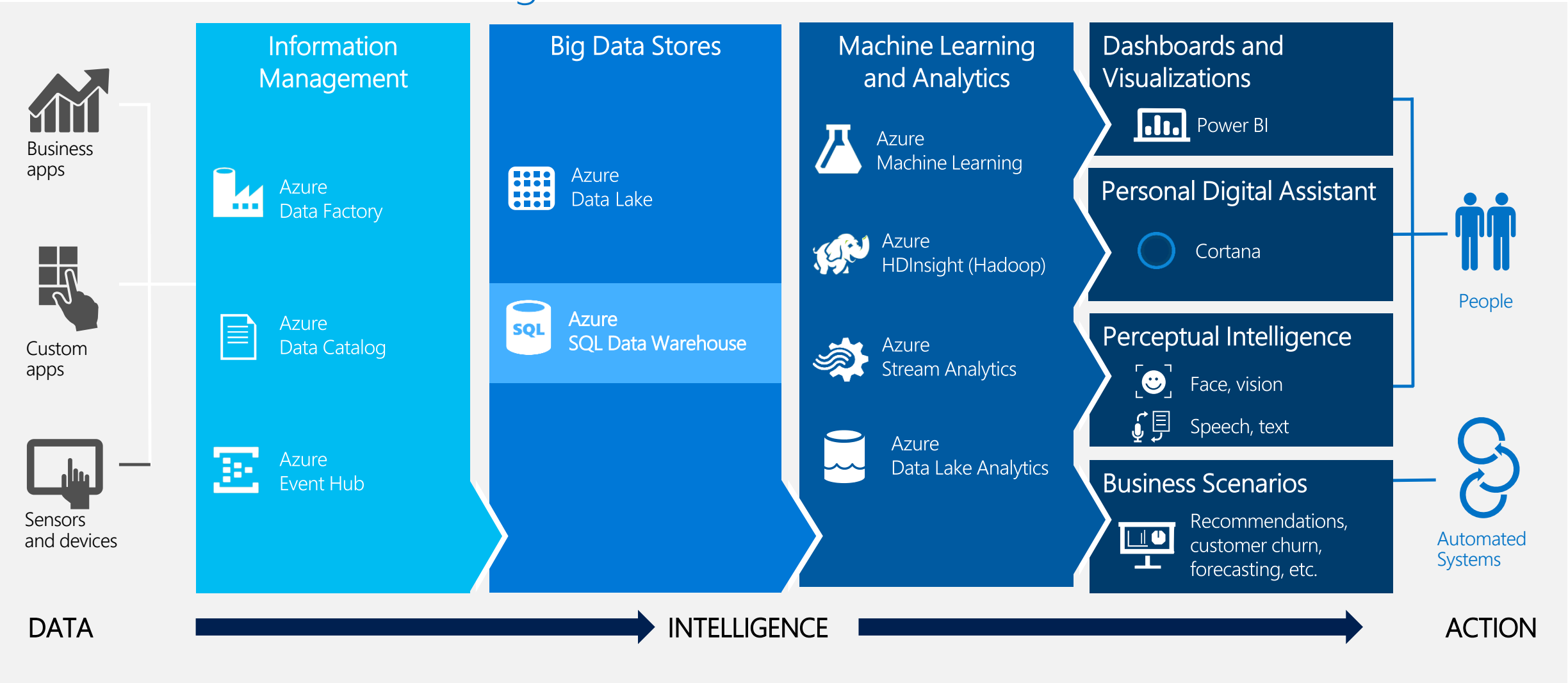
Seamless compatibility with Power BI, Azure Machine Learning, HDInsight, and Azure Data Factory



Transaction of SQL queries across relational and non-relational data in Hadoop with PolyBase

Cortana Intelligence Suite includes SQL DW

Transform data into intelligent action



Use case, patterns and adoption

SQL DW is good for analytical workloads. Why?

- ✓ 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"

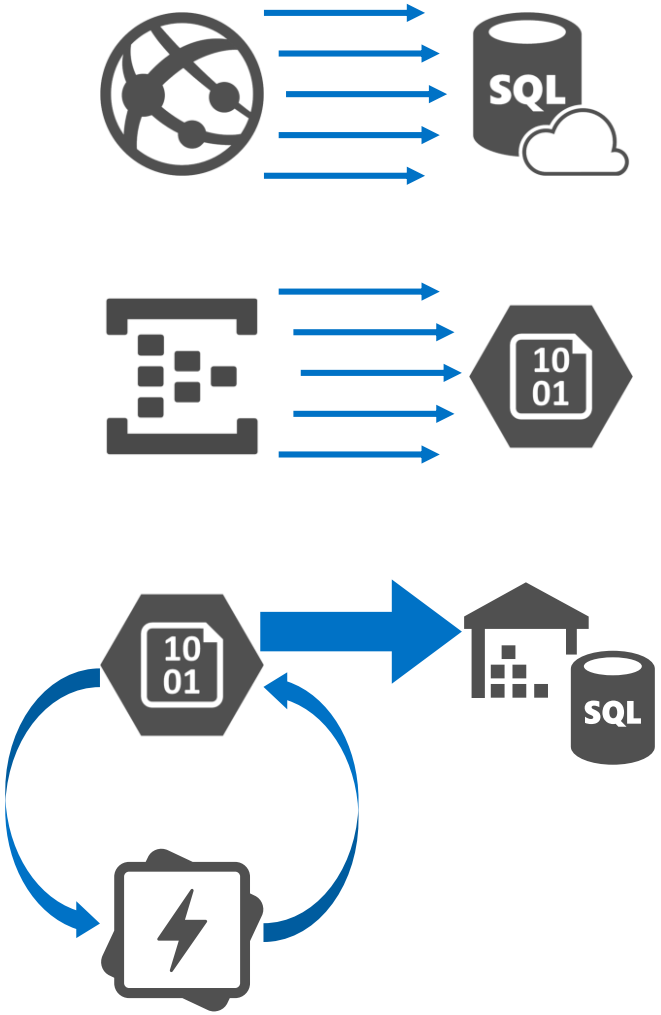
Unsuitable workloads for SQL DW

Operational workloads (OLTP)

- High frequency reads and writes.
- Large numbers of singleton selects.
- High volumes of single row inserts.

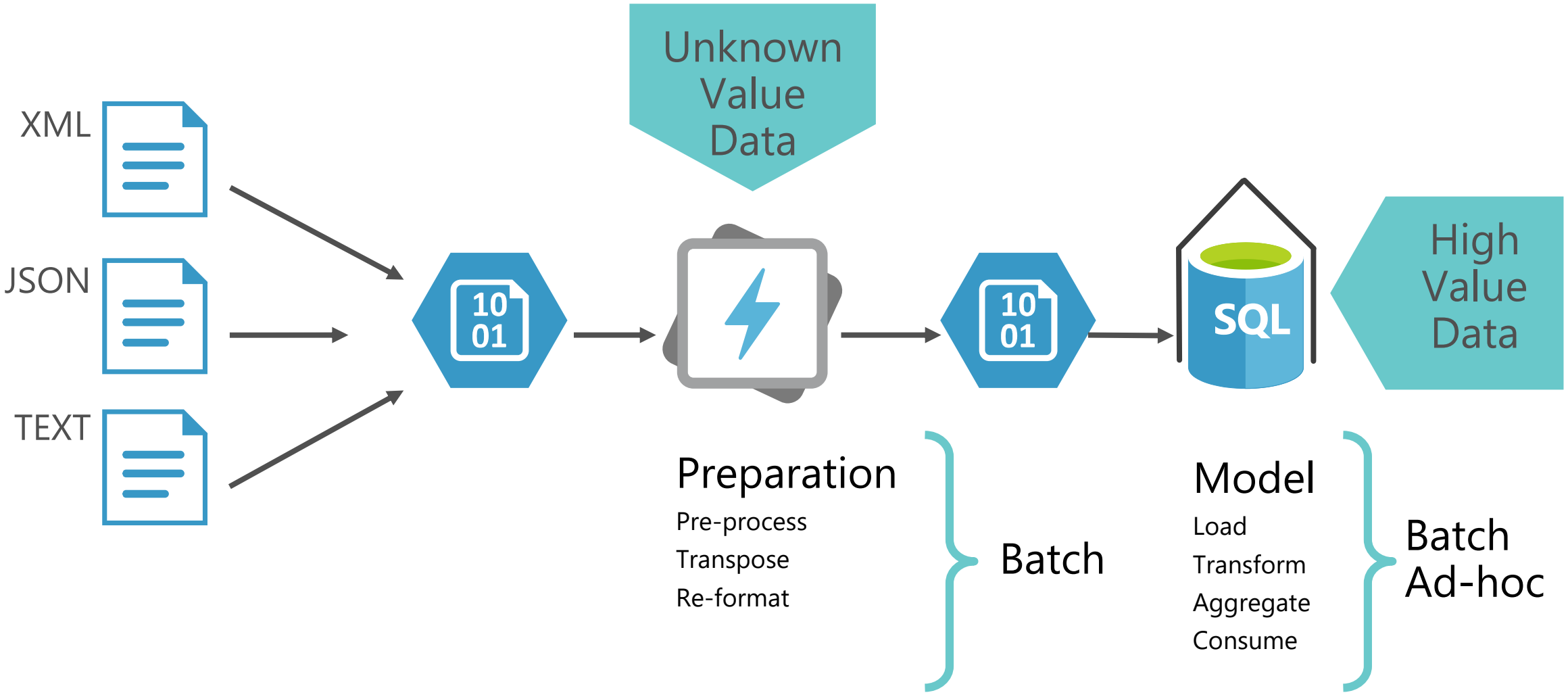
Data Preparation

- Row by row processing needs.
- Incompatible formats (JSON, XML).



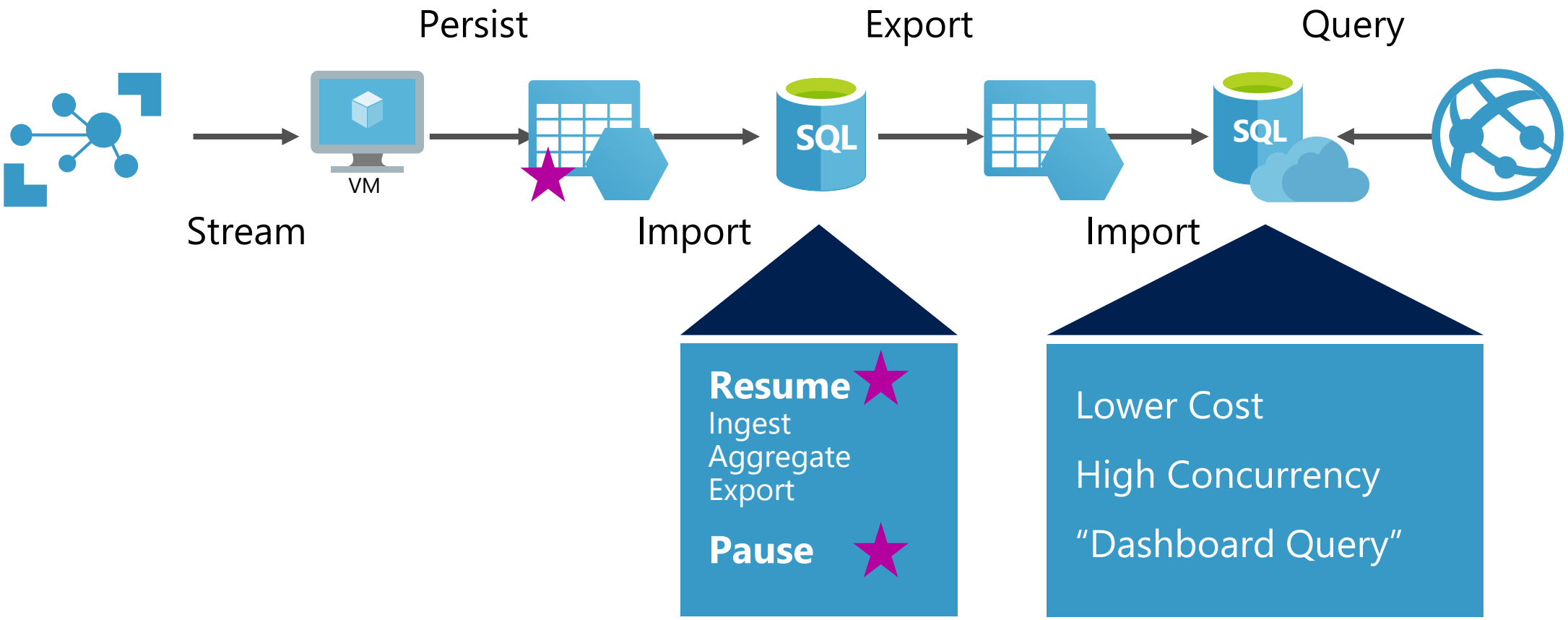
Pattern

Big Data Warehouse



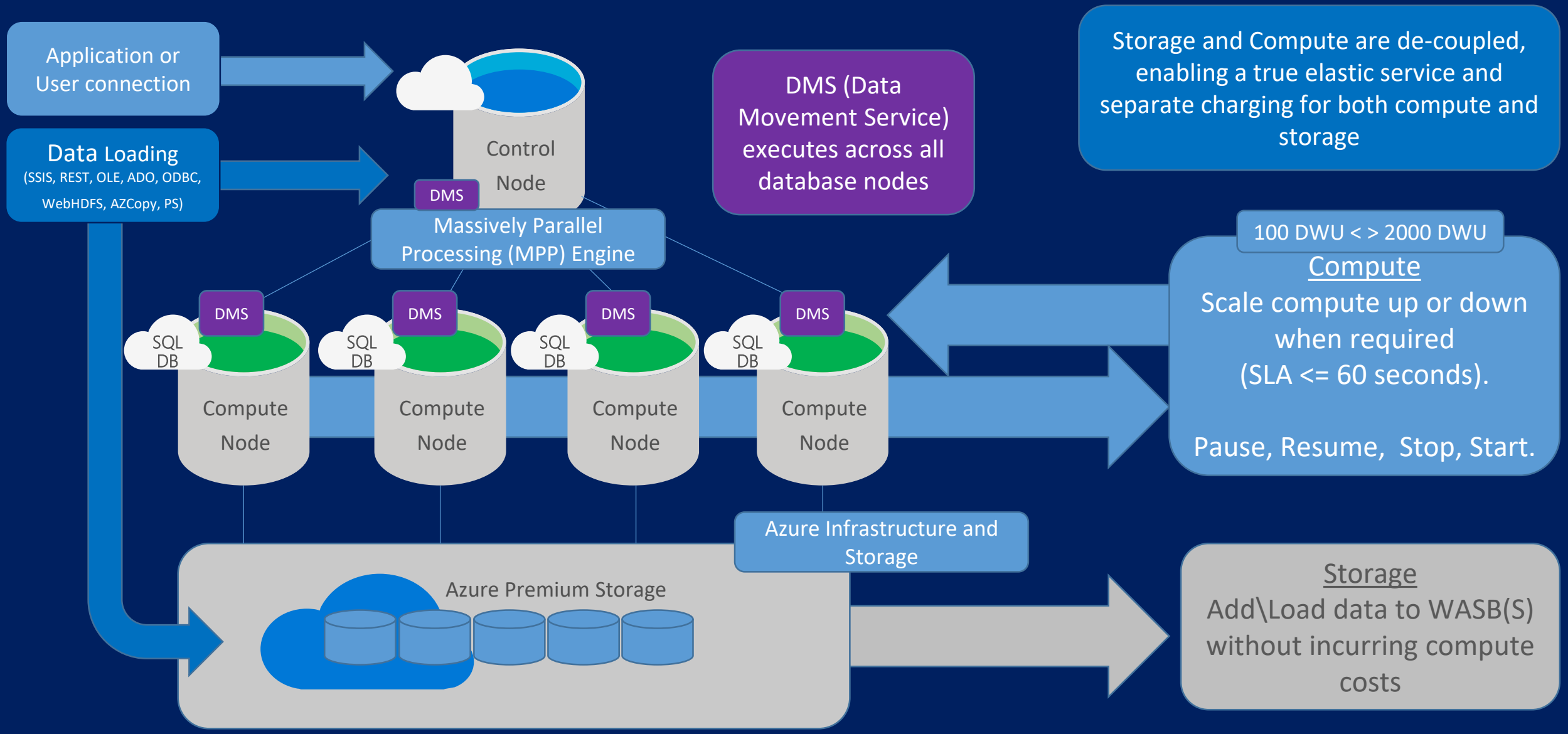
Pattern

ELASTIC consumption

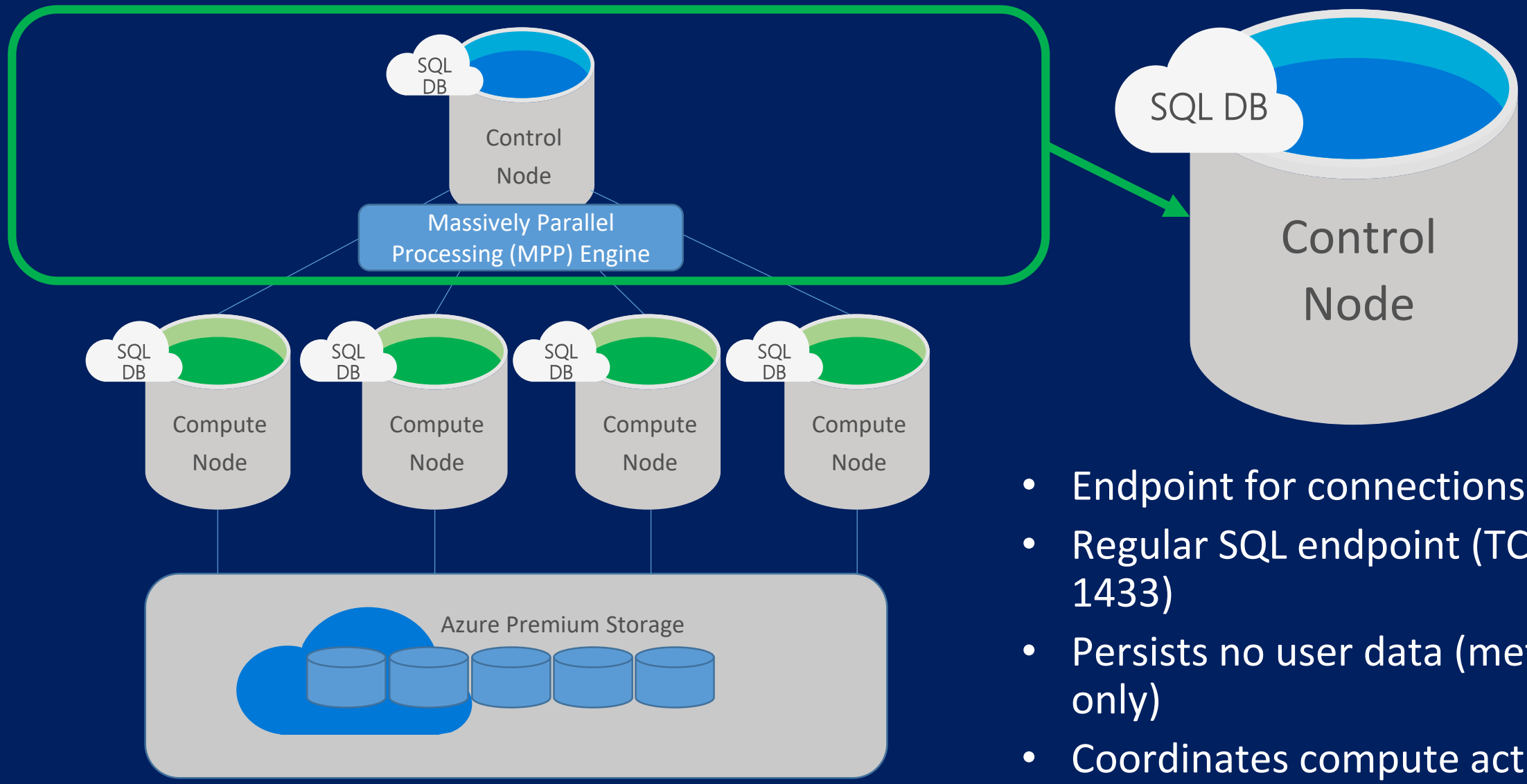


Architectural overview

Azure SQL Data Warehouse Architecture

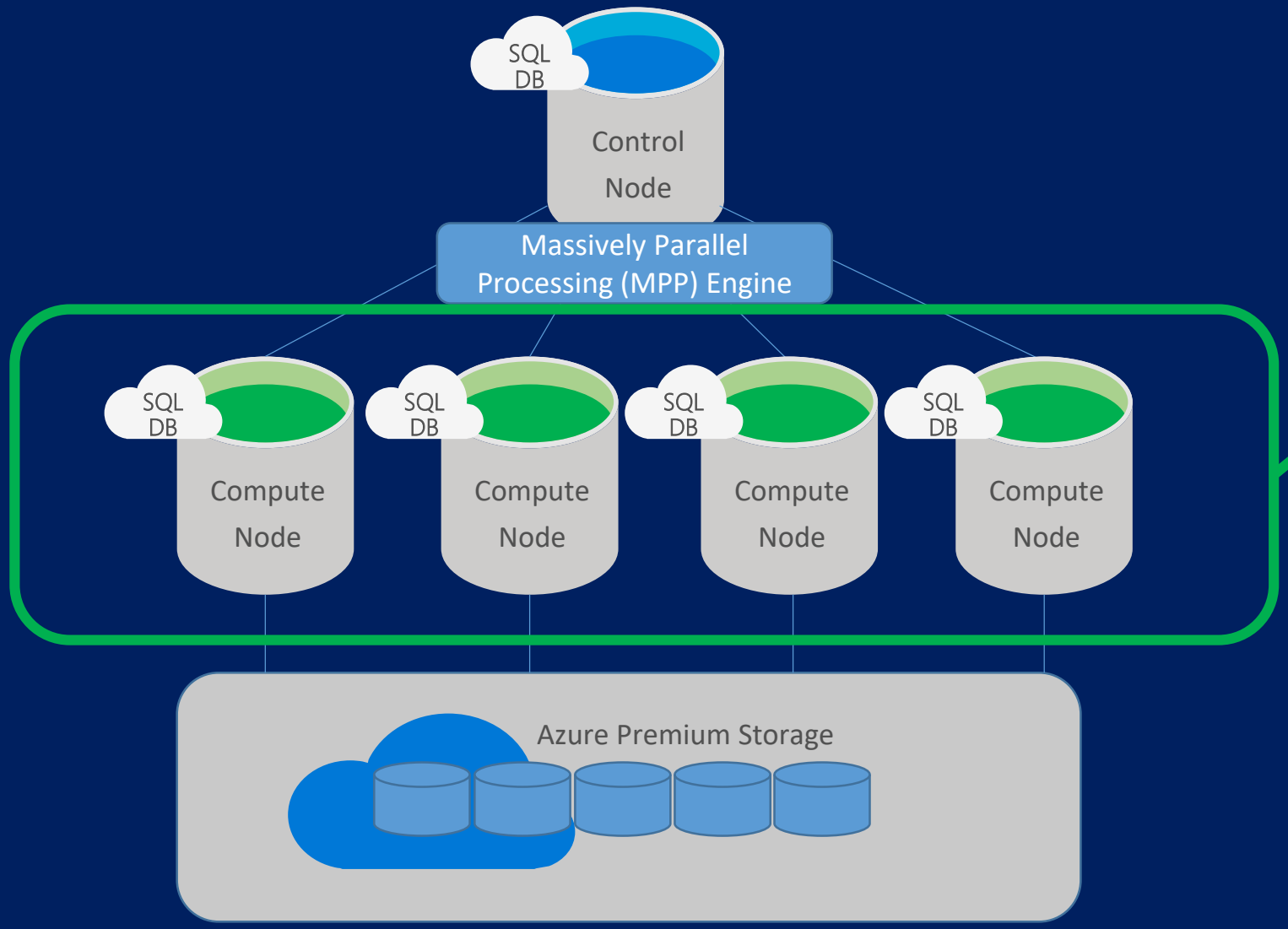


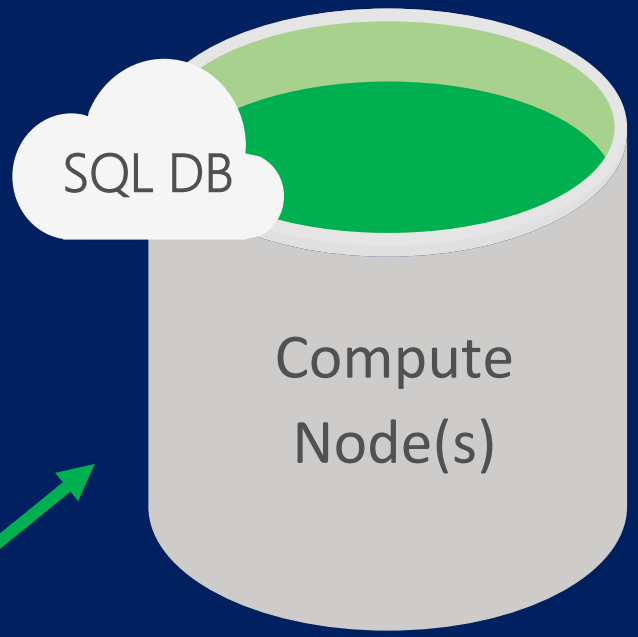
Azure SQL Data Warehouse – Control Node



- Endpoint for connections
- Regular SQL endpoint (TCP 1433)
- Persists no user data (metadata only)
- Coordinates compute activity using MPP

Azure SQL Data Warehouse - Compute Nodes

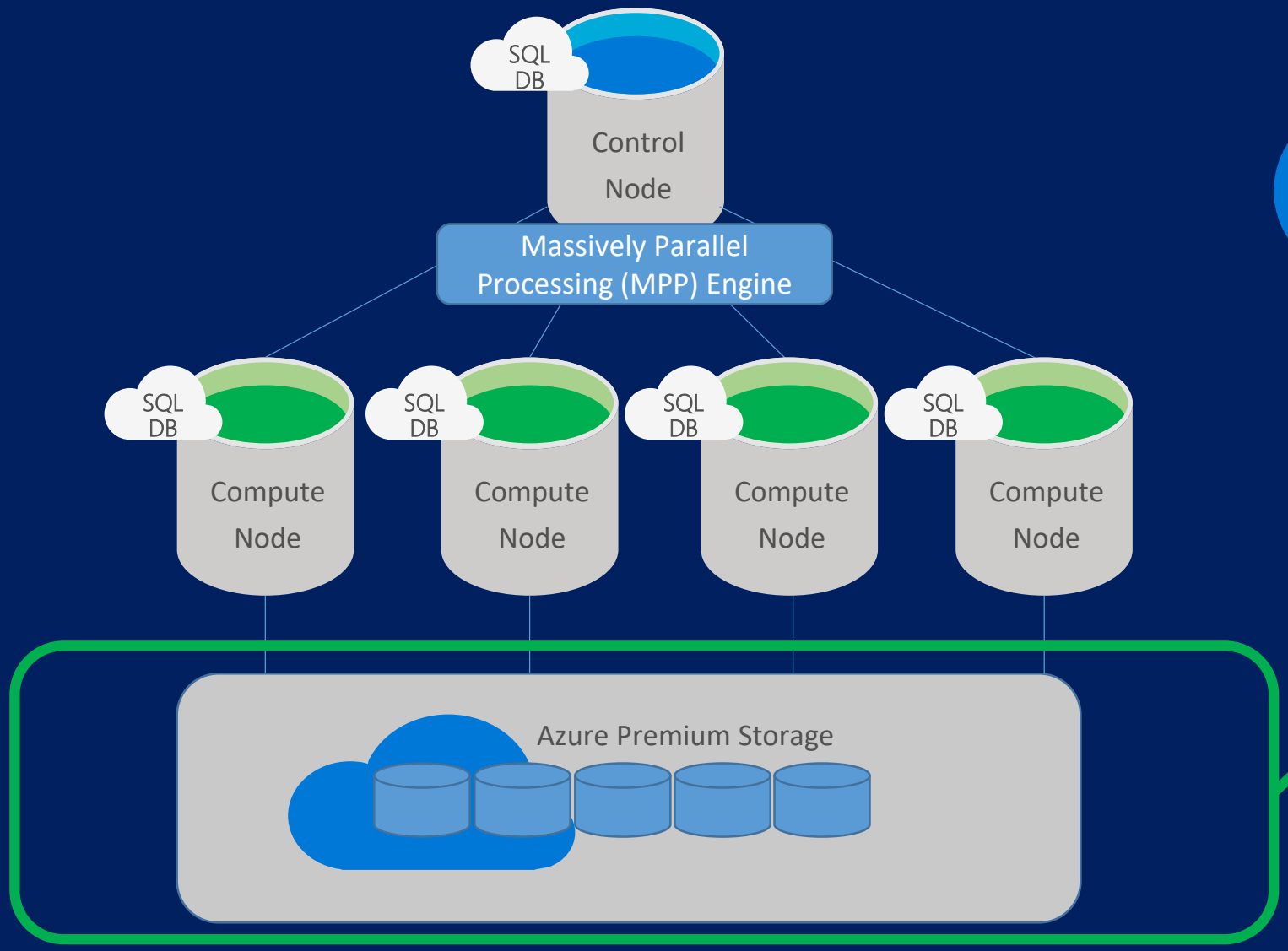


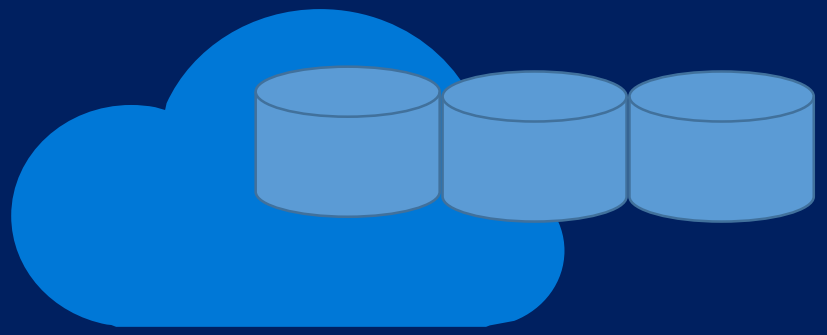


Azure SQL Database

An increase of DWU will increase the number of compute nodes

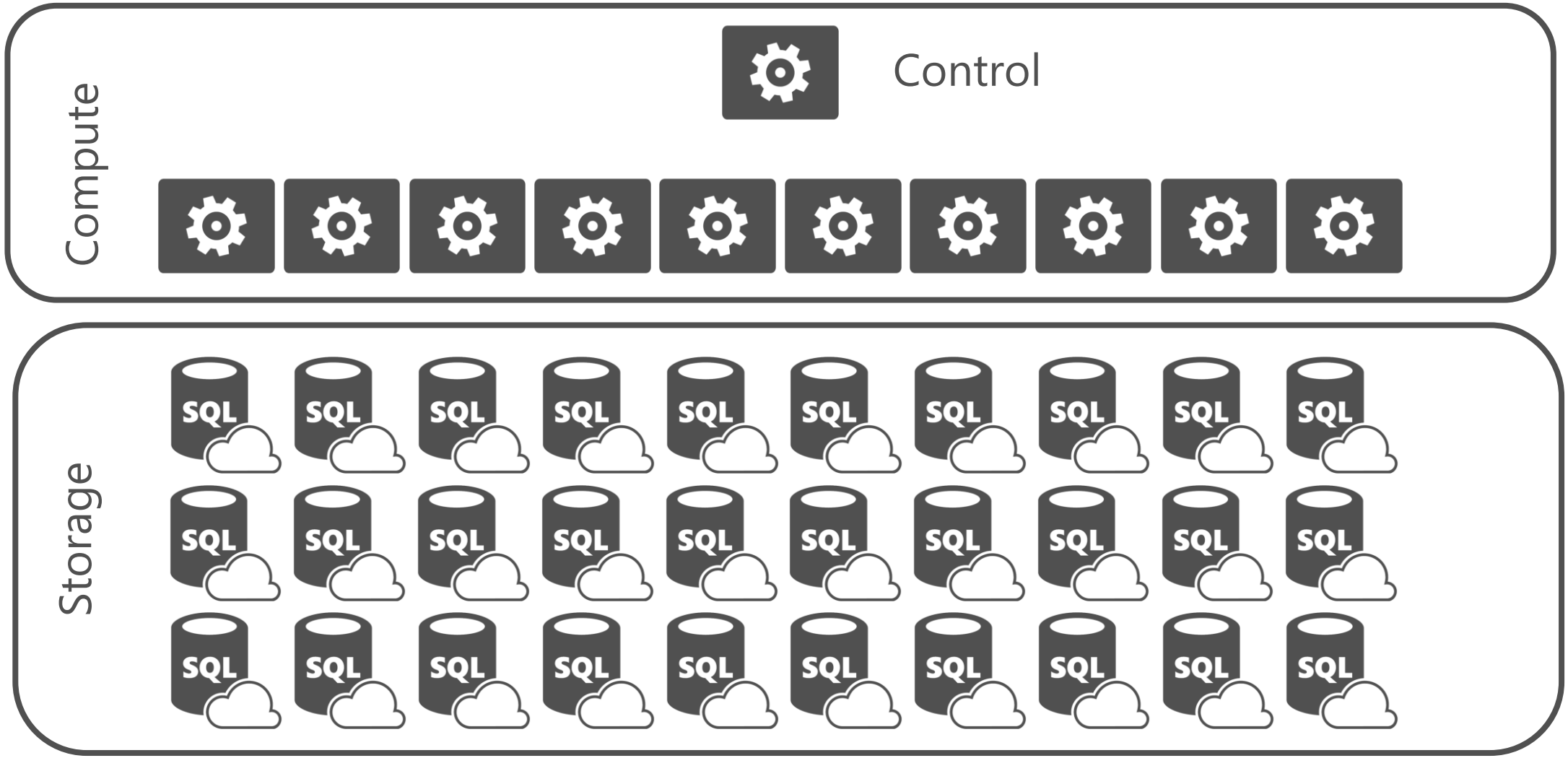
Azure SQL Data Warehouse – Blob storage





- RA-GRS storage
- +PB's of storage
- Ingest data without incurring compute costs

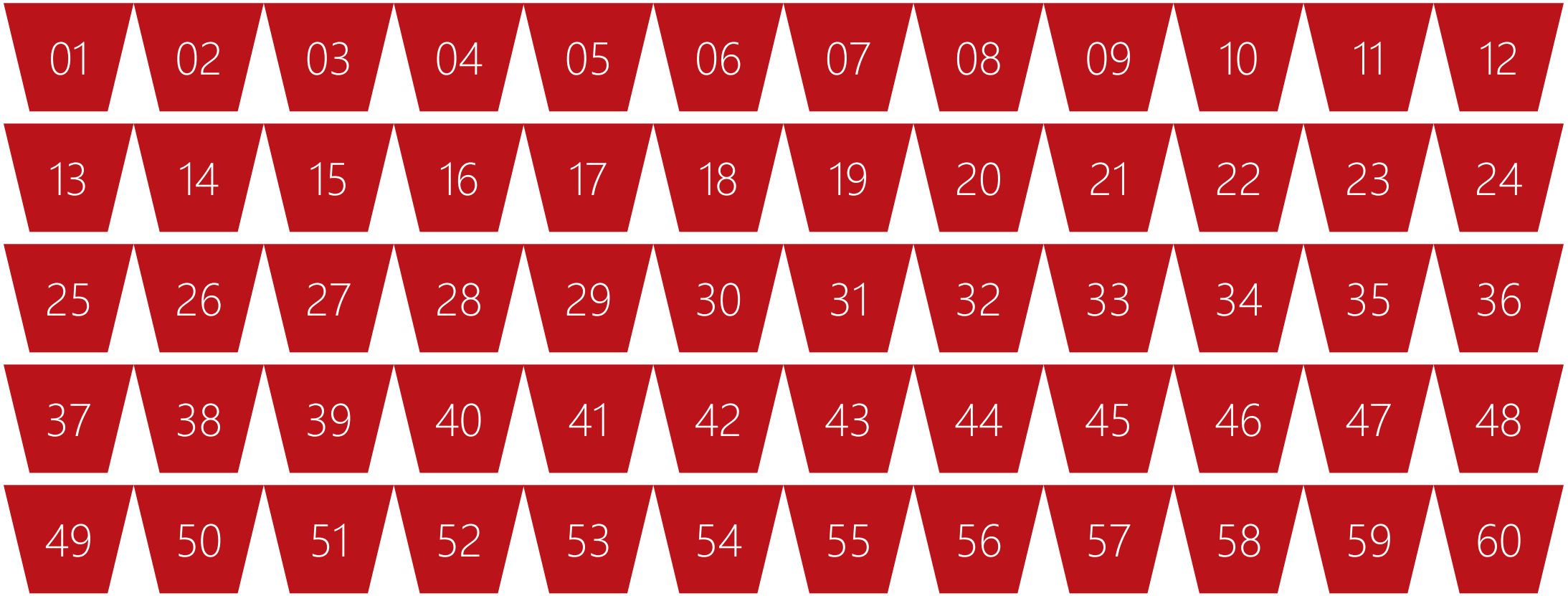
Logical overview



Scale operations

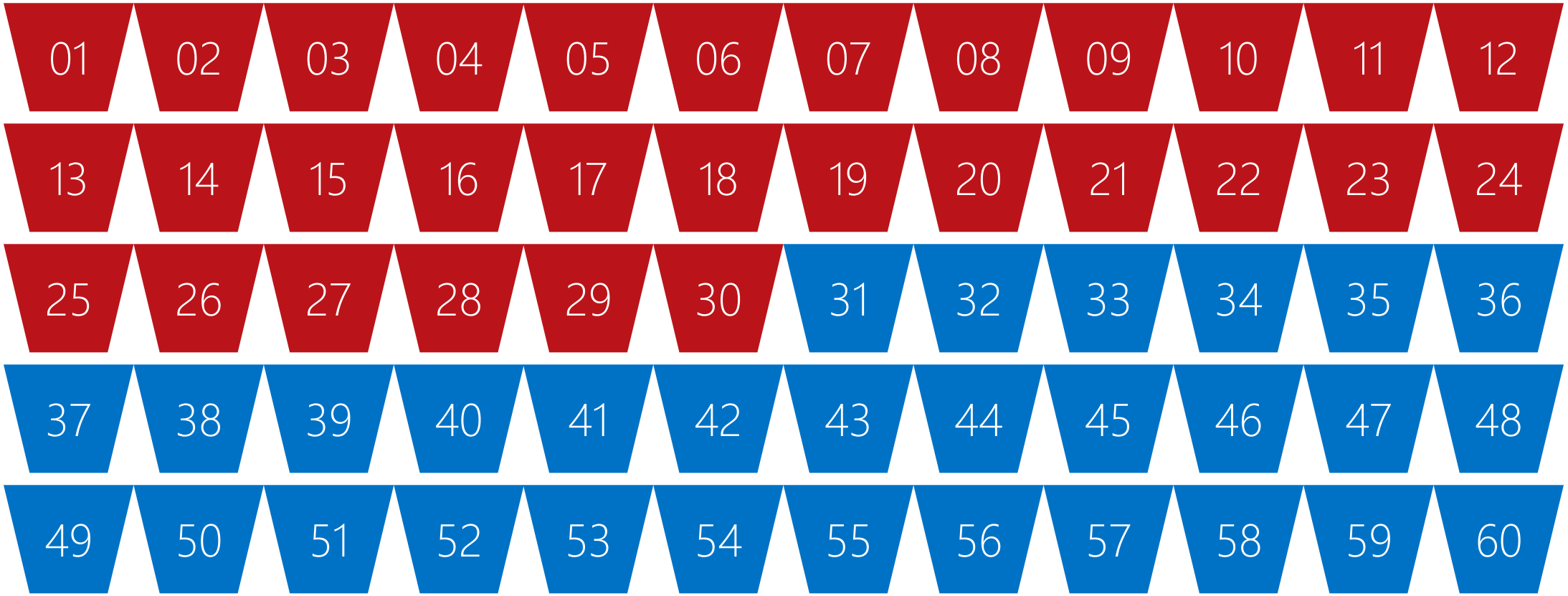
Mapping Compute in SQLDW

DW100



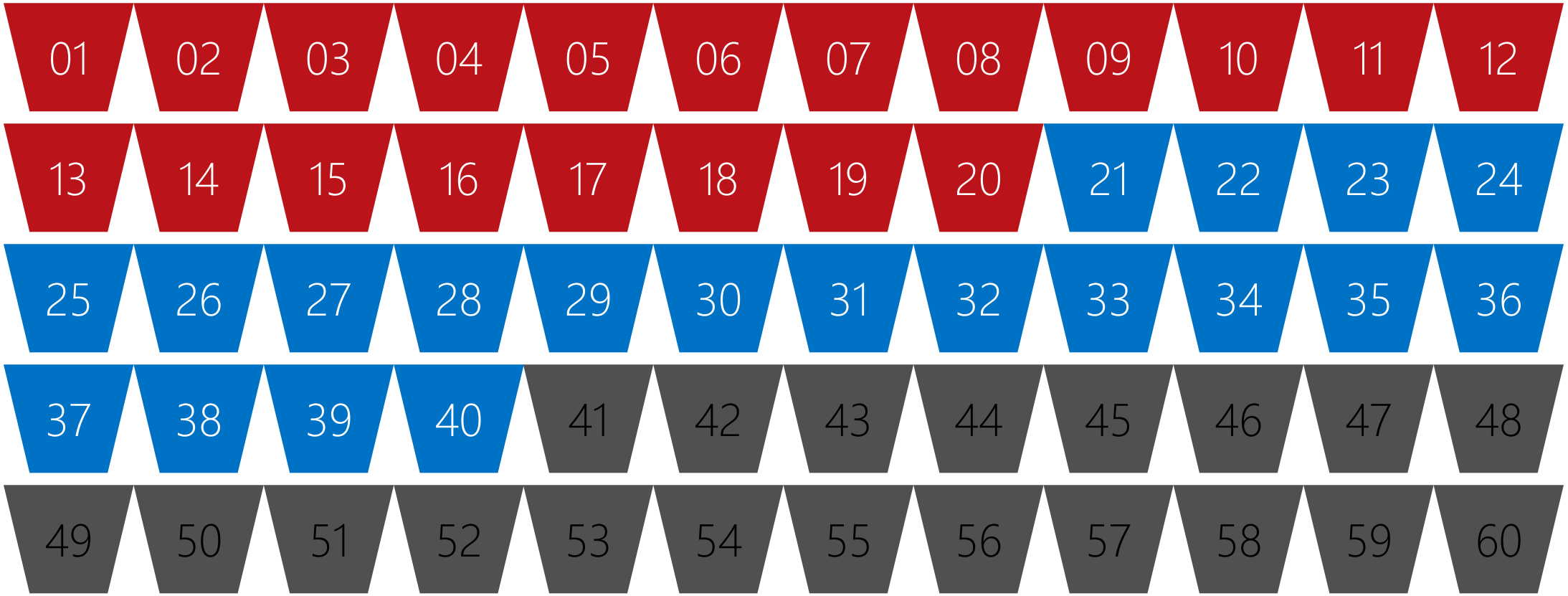
Mapping Compute in SQLDW

DW200



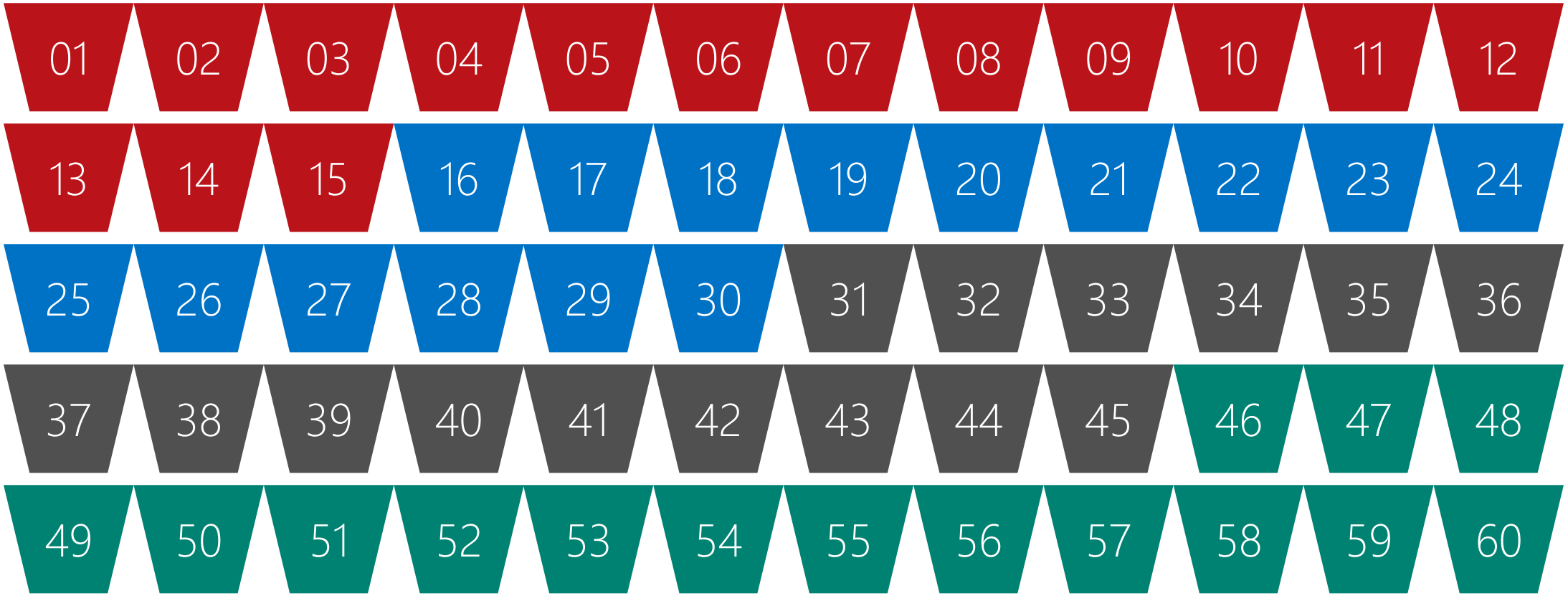
Mapping Compute in SQLDW

DW300



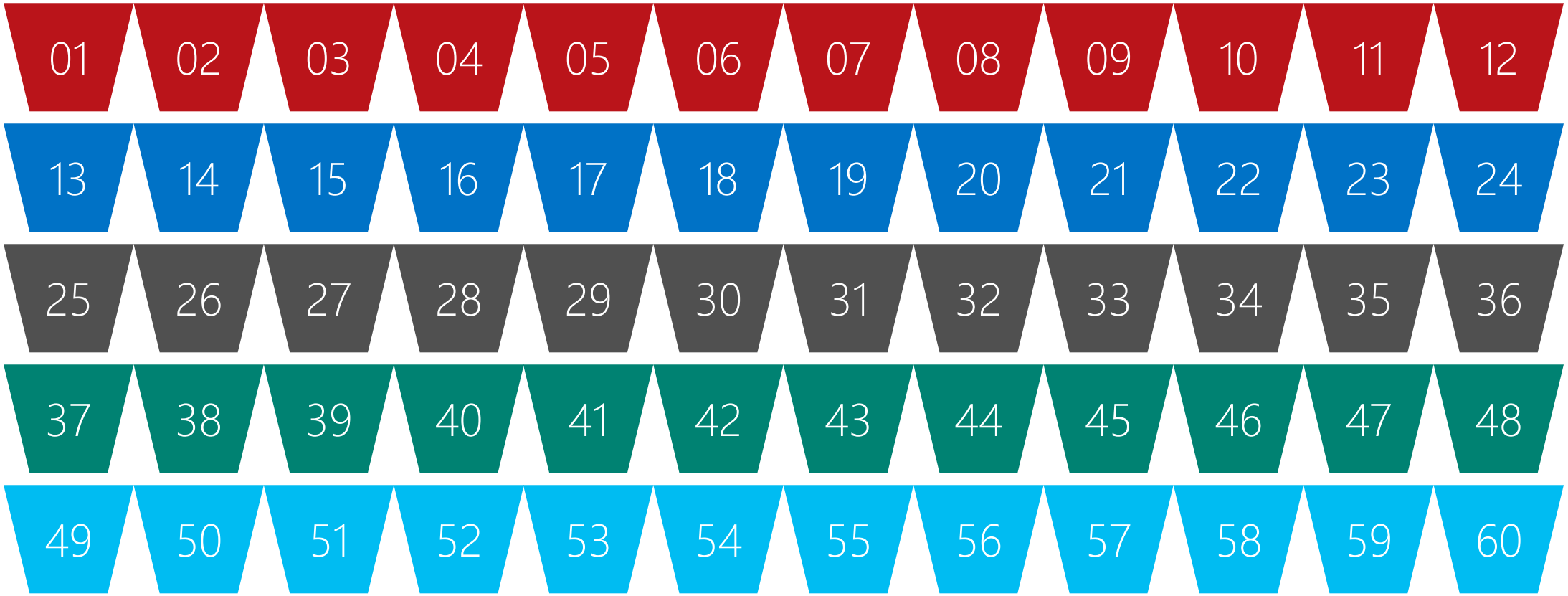
Mapping Compute in SQLDW

DW400



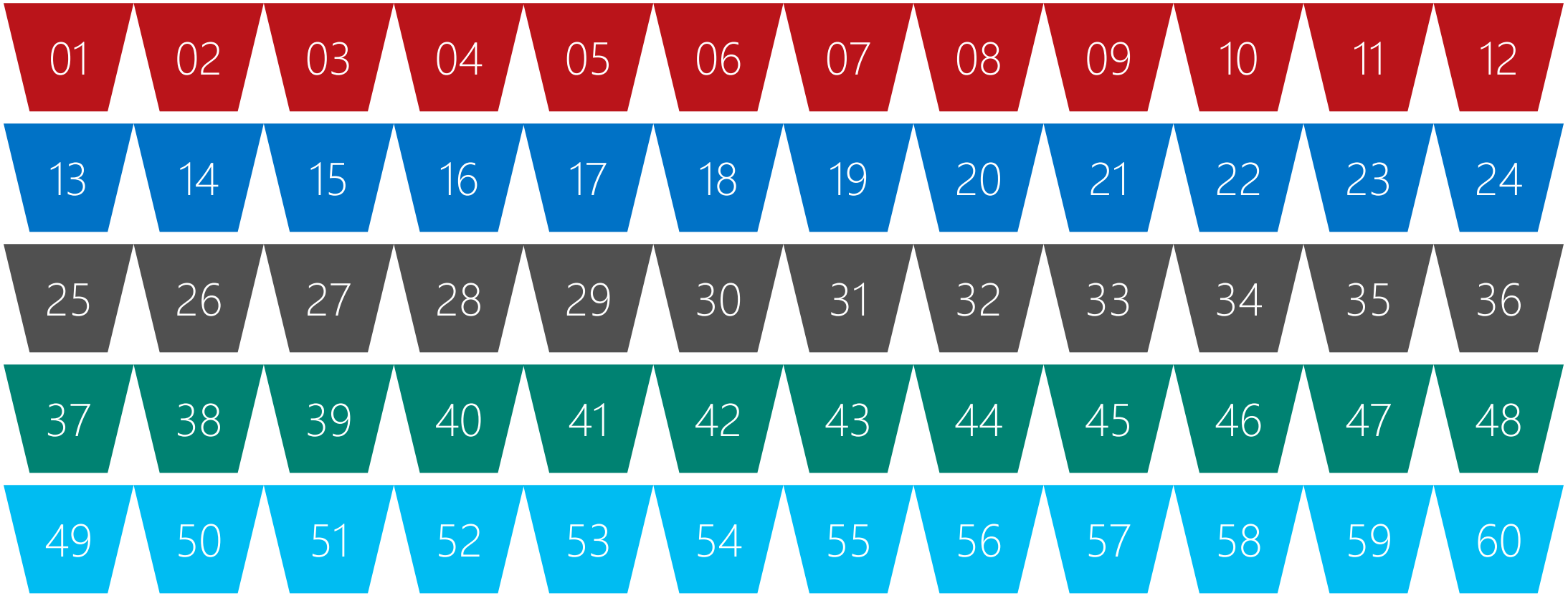
Mapping Compute in SQLDW

DW500



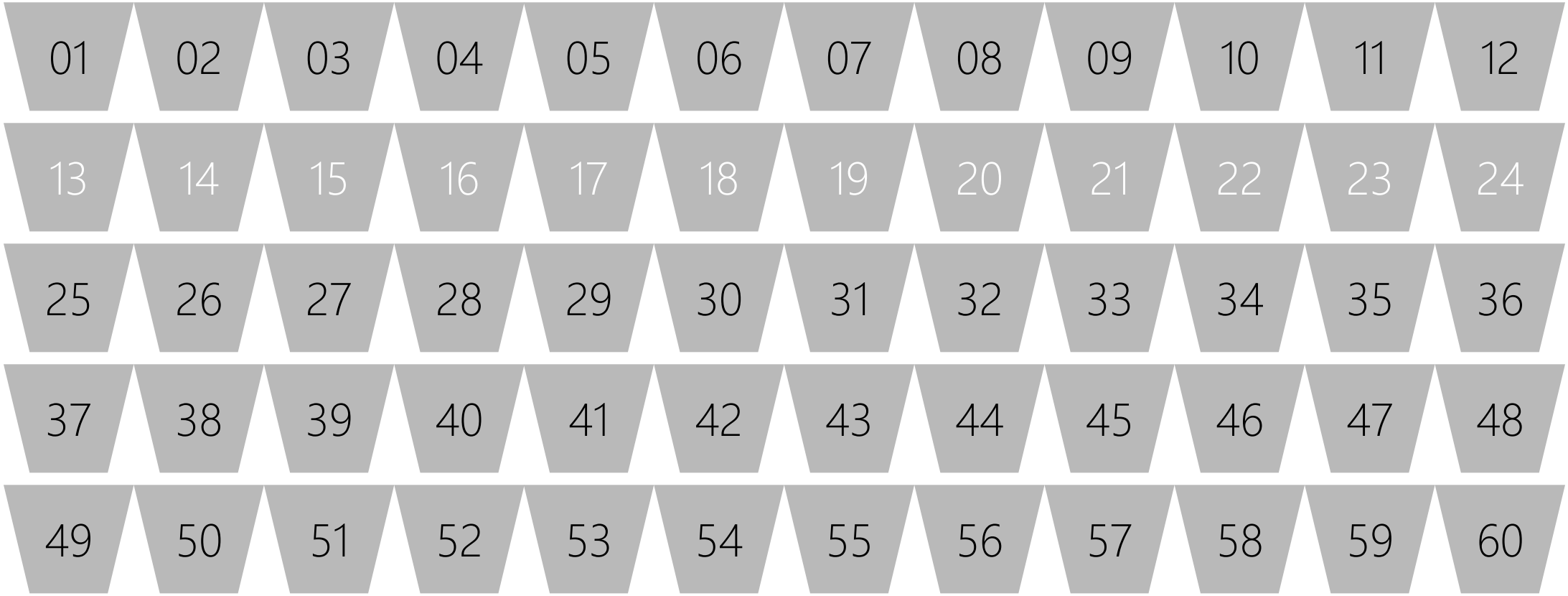
Pausing compute in SQLDW

DW500



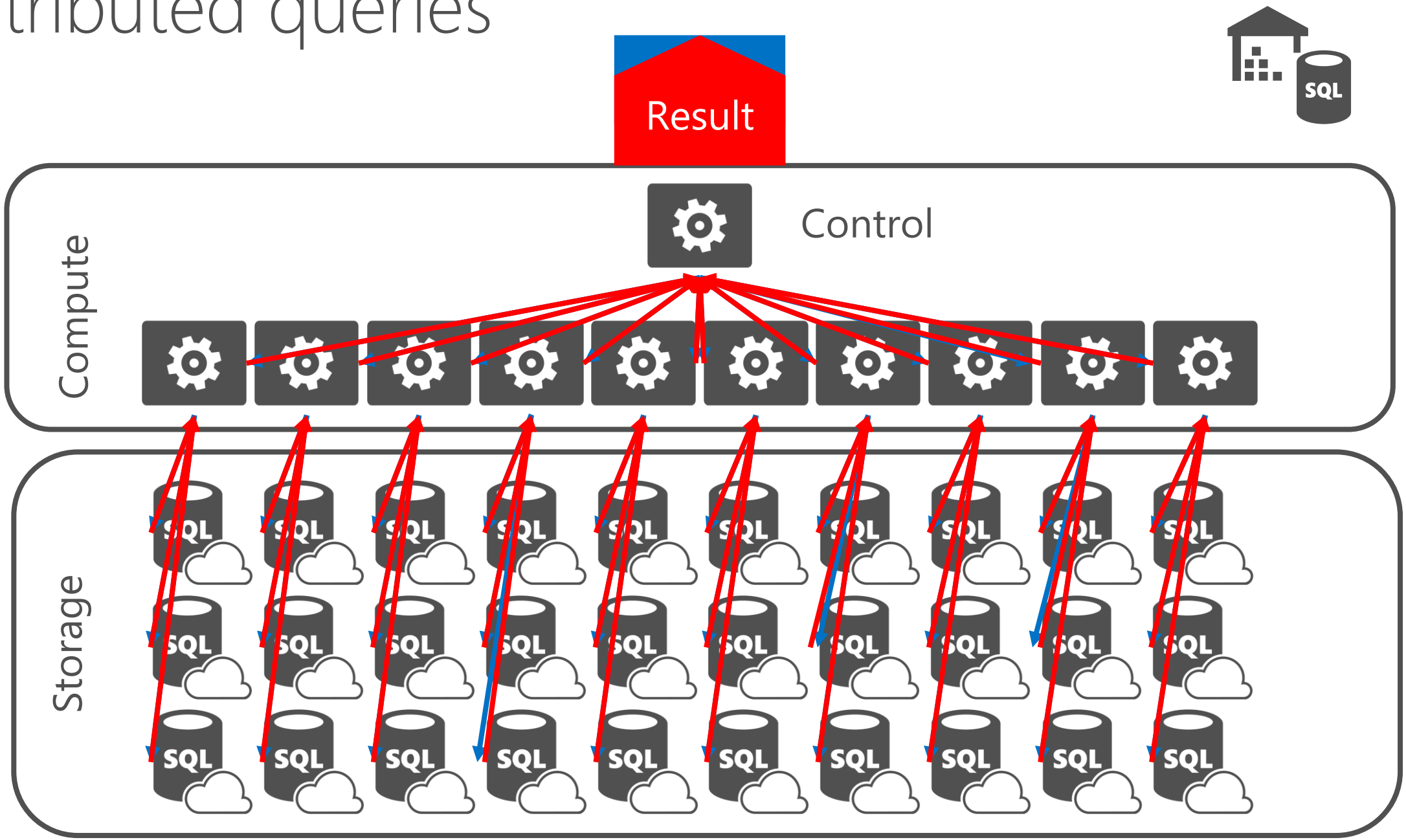
Resuming compute in SQLDW

DW500



Tables & Distribution

Distributed queries



Simple example

```
SELECT  COUNT_BIG(*)
FROM    dbo.[FactInternetSales]
;
```

```
SELECT  SUM(*)
FROM    dbo.[FactInternetSales]
;
```

Control

Compute

```
SELECT  COUNT_BIG(*)
FROM    dbo.[FactInternetSales]
;
```

```
SELECT  COUNT_BIG(*)
FROM    dbo.[FactInternetSales]
;
```

```
SELECT  COUNT_BIG(*)
FROM    dbo.[FactInternetSales]
;
```

```
SELECT  COUNT_BIG(*)
FROM    dbo.[FactInternetSales]
;
```

Creating tables

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]        int          NOT NULL
,   [DateKey]               datetime     NOT NULL
,   [StoreKey]              int          NOT NULL
,   [ProductKey]            int          NOT NULL
,   [PromotionKey]          int          NOT NULL
,   [CurrencyKey]           int          NOT NULL
,   [CustomerKey]           int          NOT NULL
,   [SalesOrderNumber]      nvarchar(20) NOT NULL
,   [SalesOrderLineNumber]  int              NULL
,   [SalesQuantity]         int          NOT NULL
,   [SalesAmount]           money        NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = ROUND_ROBIN
)
;
```

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]        int          NOT NULL
,   [DateKey]               datetime     NOT NULL
,   [StoreKey]              int          NOT NULL
,   [ProductKey]            int          NOT NULL
,   [PromotionKey]          int          NOT NULL
,   [CurrencyKey]           int          NOT NULL
,   [CustomerKey]           int          NOT NULL
,   [SalesOrderNumber]      nvarchar(20) NOT NULL
,   [SalesOrderLineNumber]  int              NULL
,   [SalesQuantity]         int          NOT NULL
,   [SalesAmount]           money        NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([ProductKey])
)
;
```

ROUND ROBIN DISTRIBUTION

01	02	03	04	05	06	07	08	09	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60

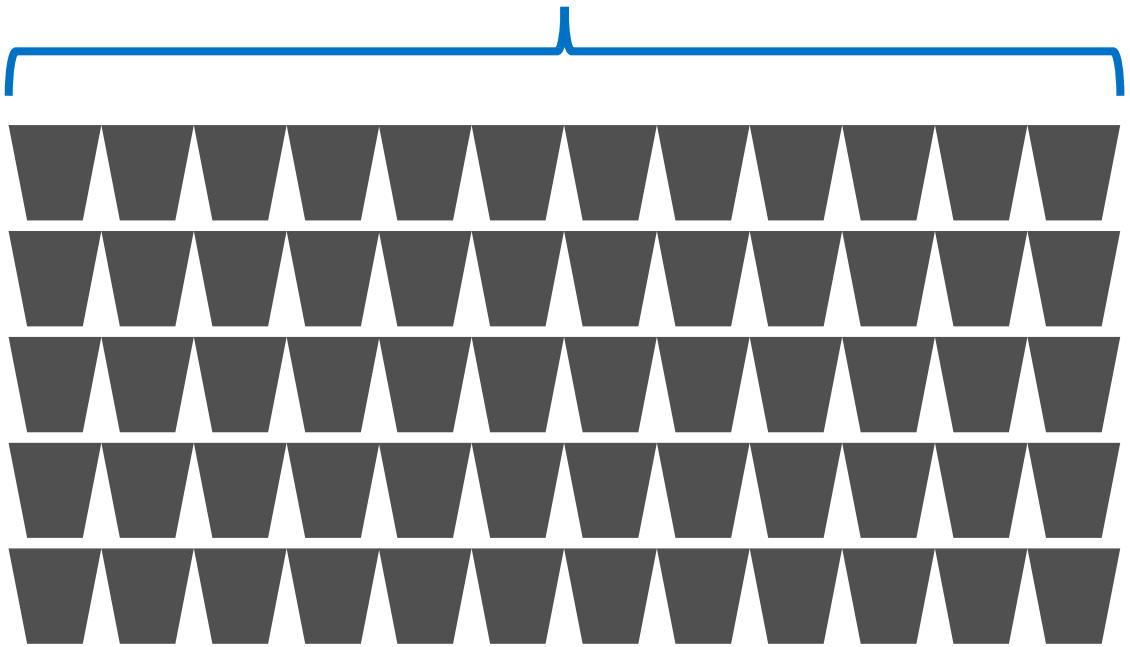
HASH distribution

HASH (03)

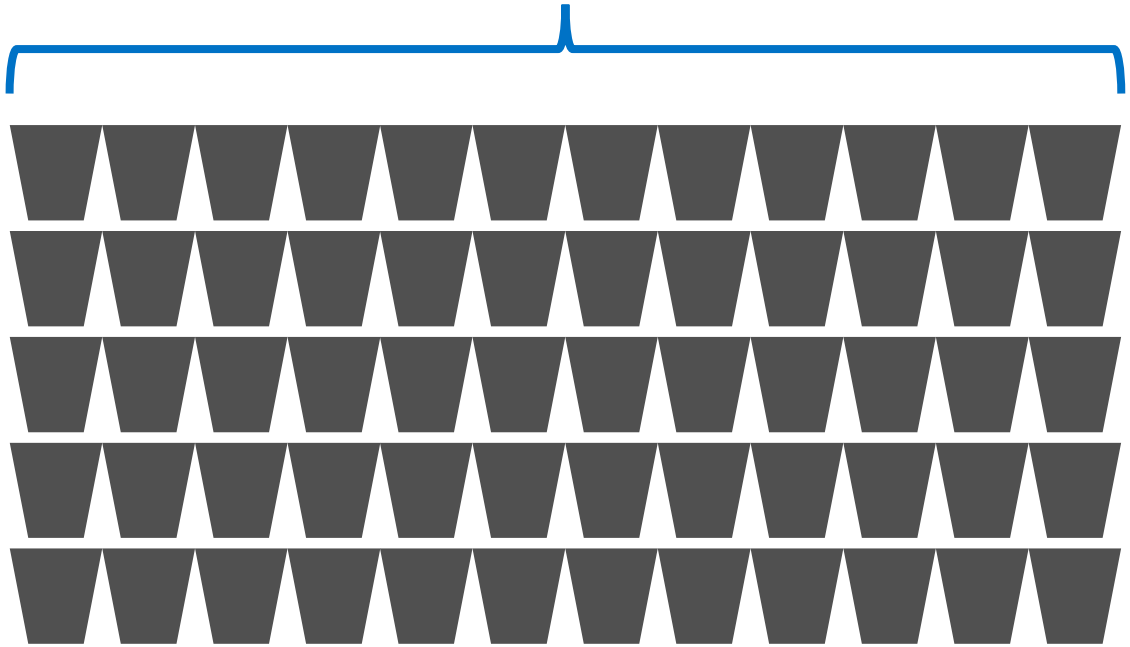
01	02	03	04	05	06	07	08	09	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60

Joining HASH tables

Store_Sales HASH([ProductKey])



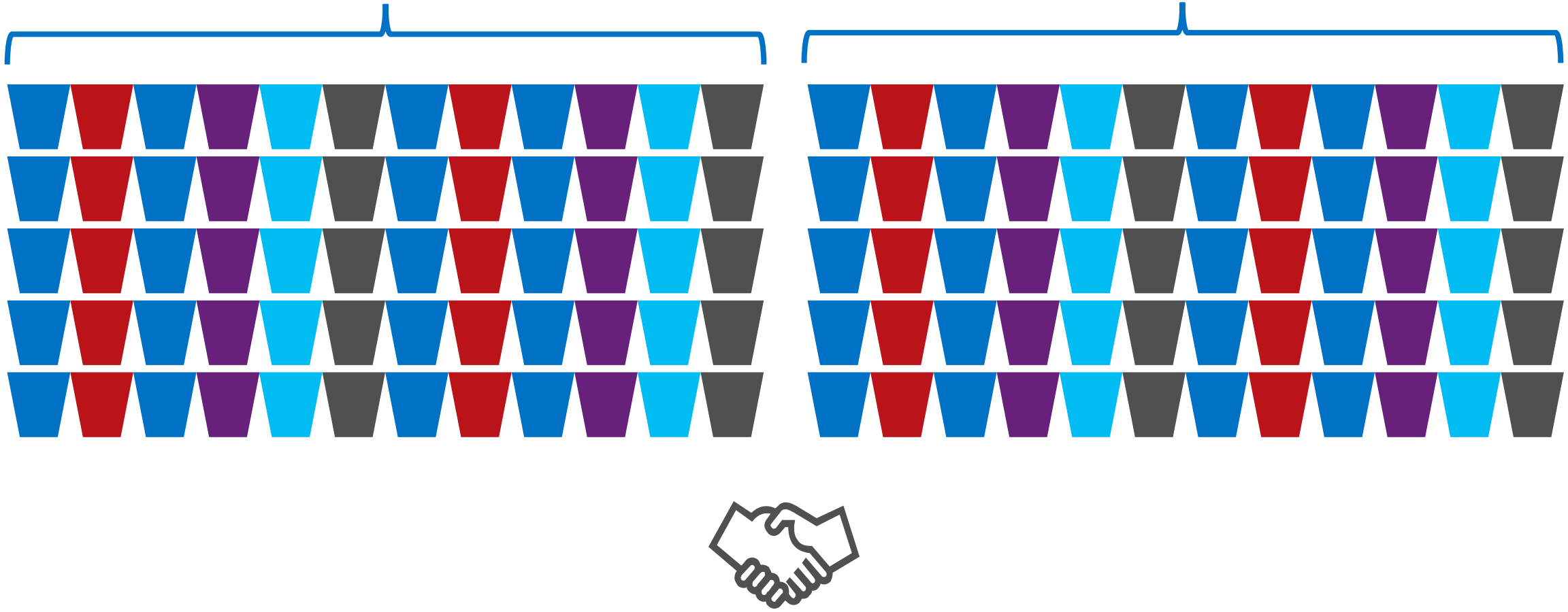
Web_Sales HASH([ProductKey])



Joining HASH tables

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

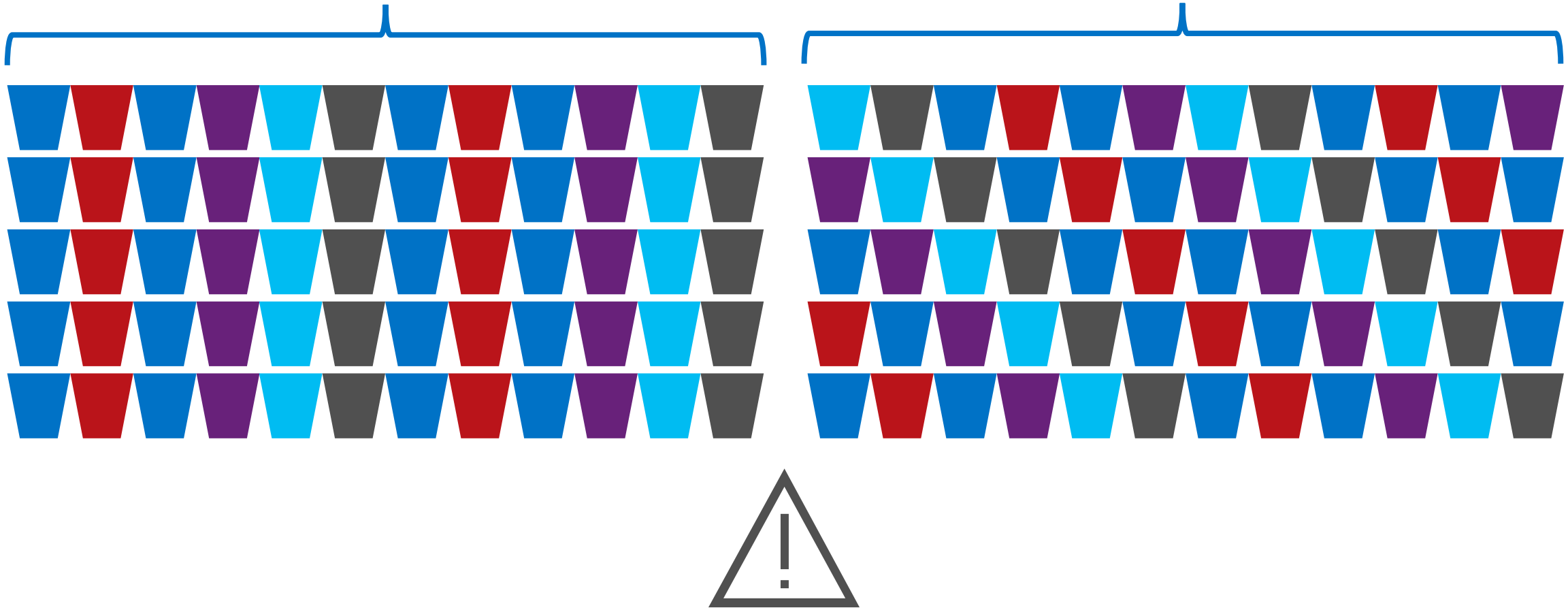
Web_Sales HASH([ProductKey])
[ProductKey] **INT** NULL



Joining HASH tables

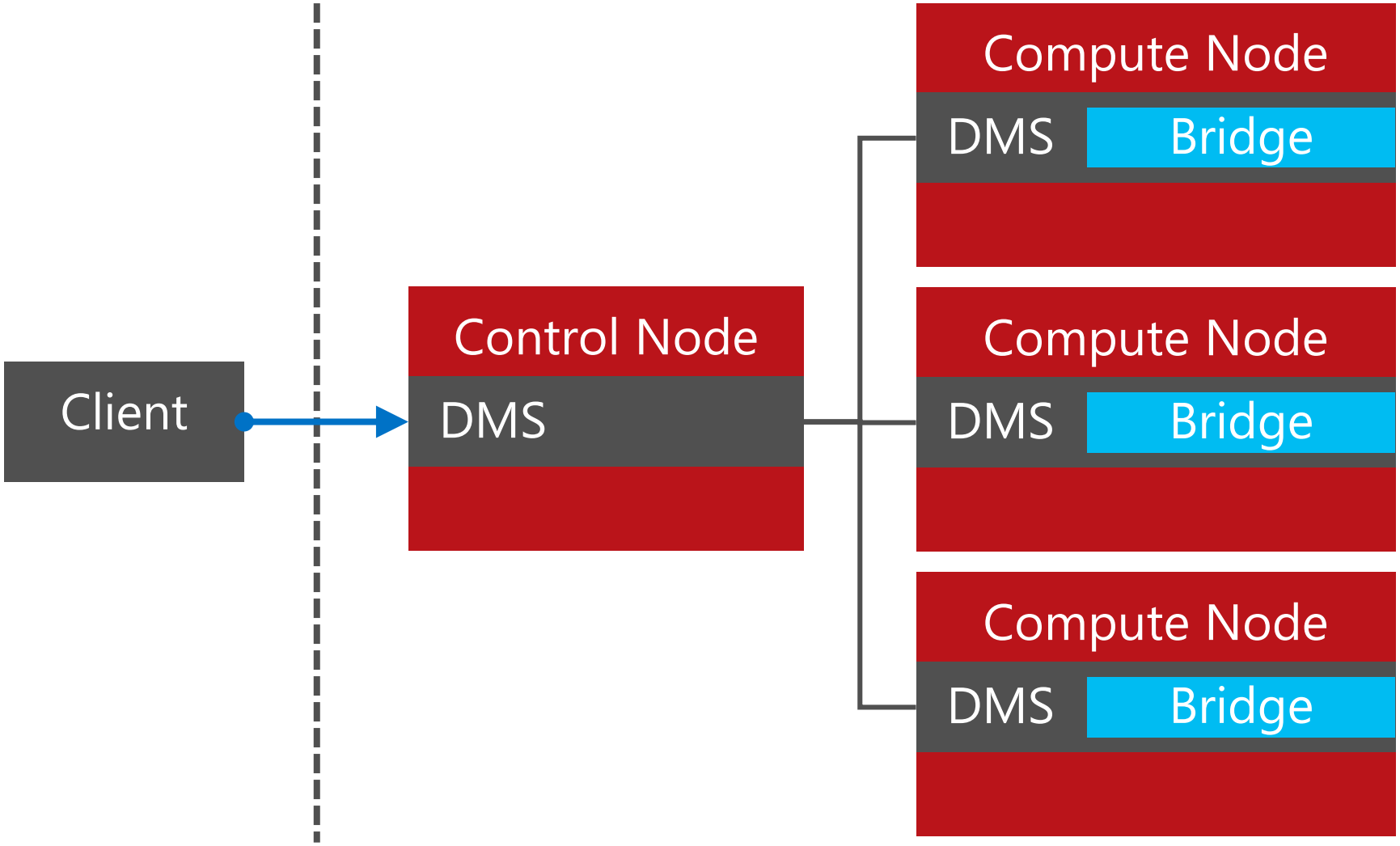
Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

Web_Sales HASH([ProductKey])
[ProductKey] **BIGINT** NULL

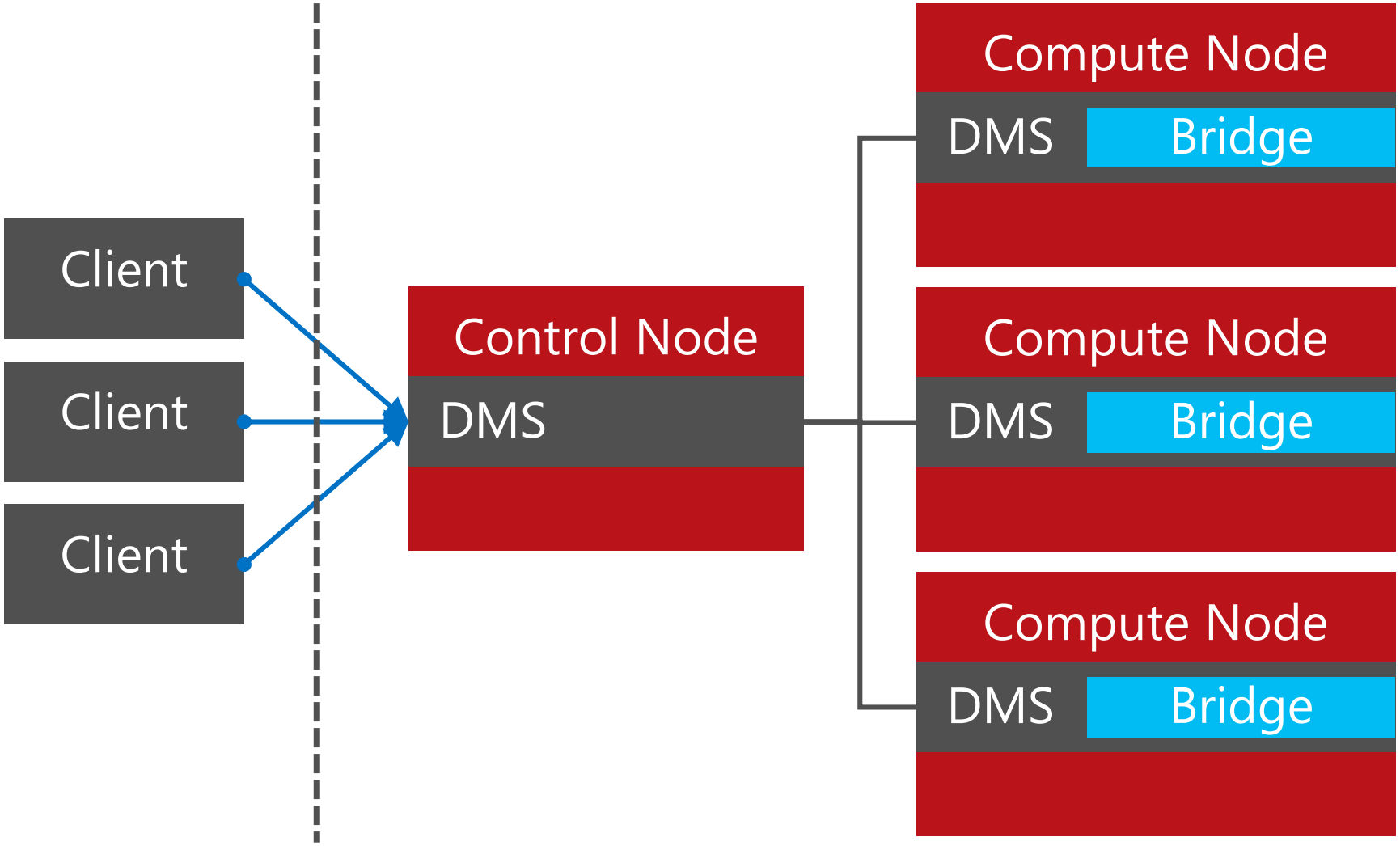


Client connectivity

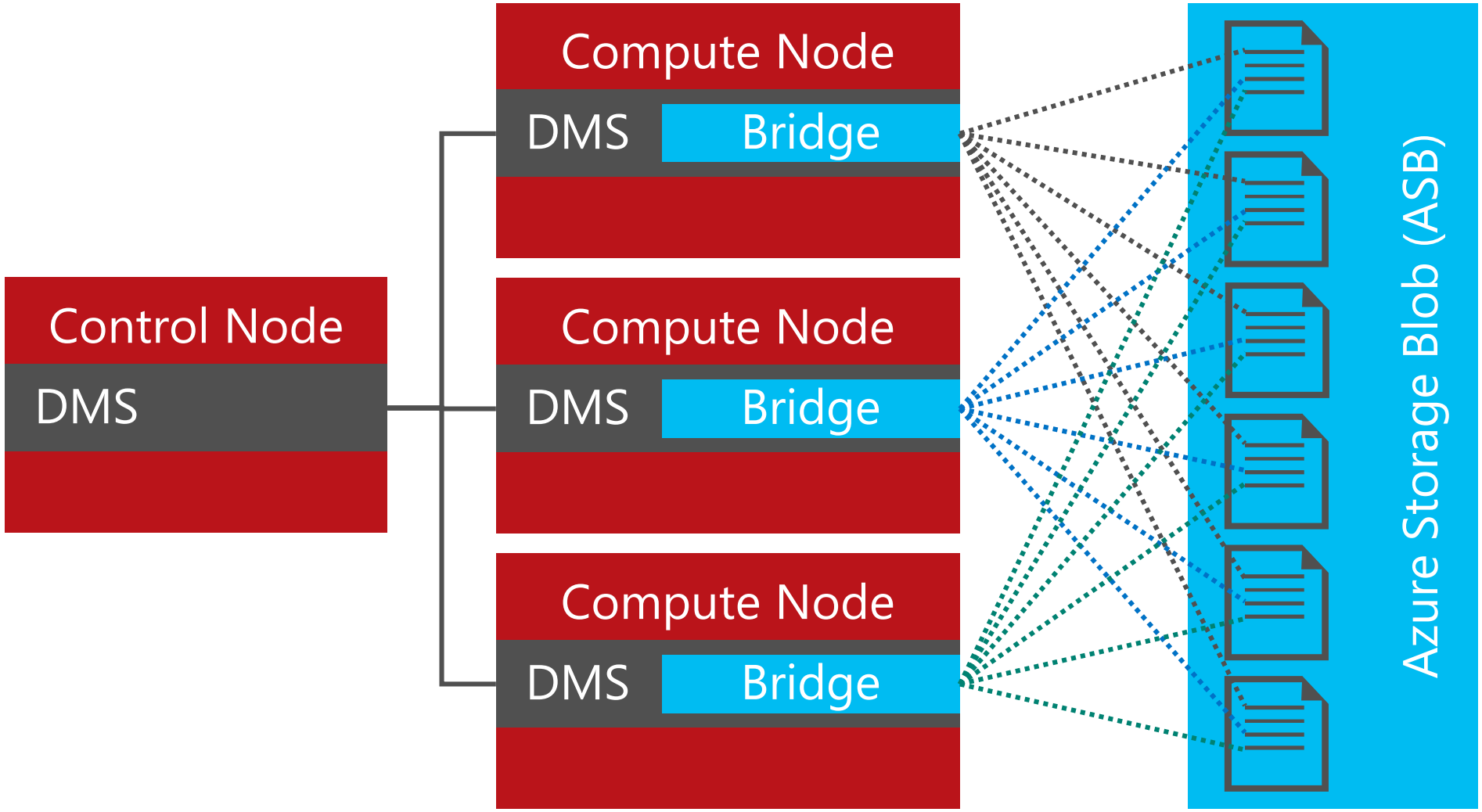
Single Gated Client



Single Gated Client Parallelised



Parallel Loading with PolyBase



Connectivity options

Windows or Linux

- ODBC
- OLEDB
- JDBC
- ADO.NET
- PHP

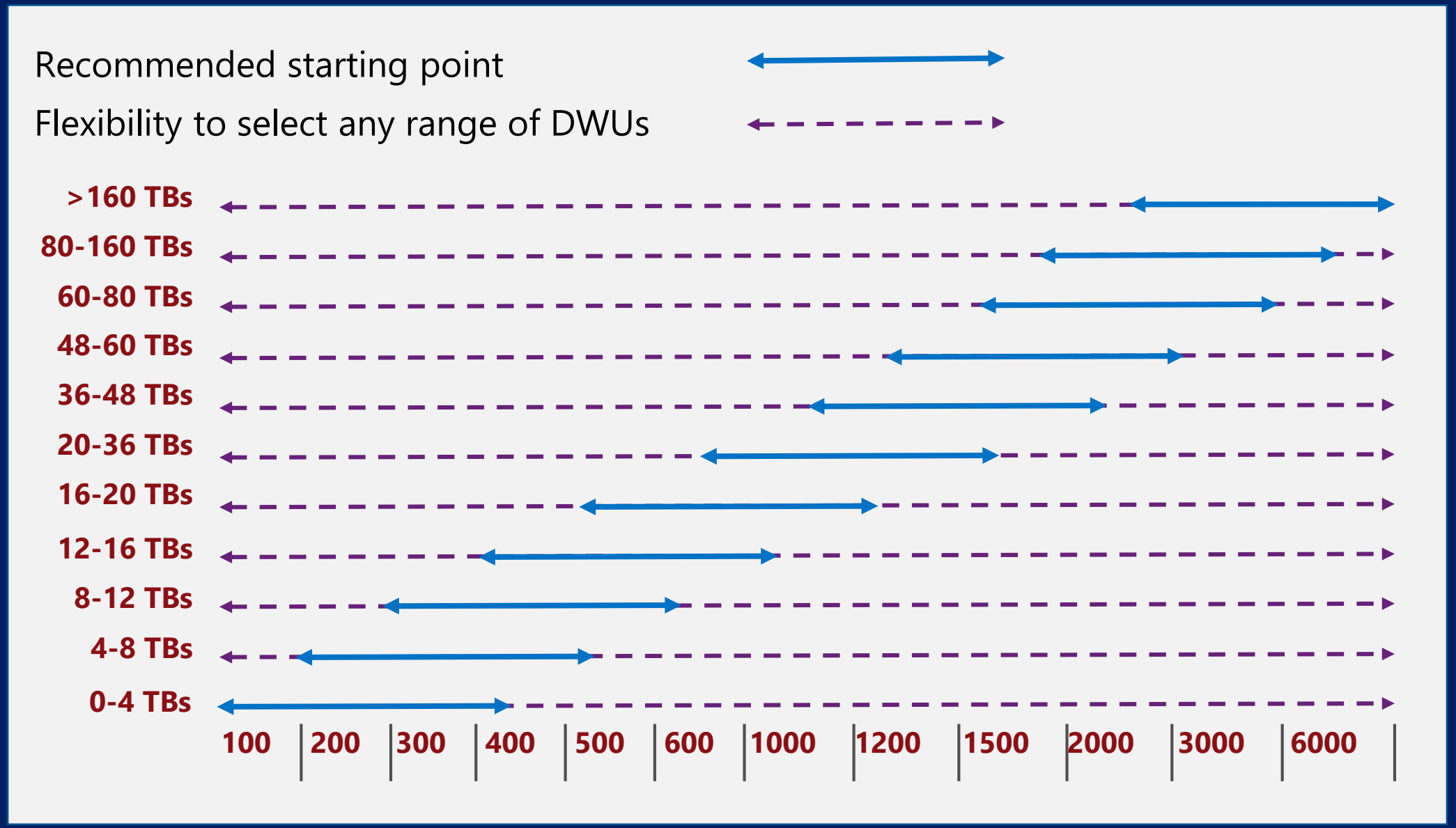
Sizing & Storage tiers

Sizing factors

Database capacity
Tempdb
Concurrency & Memory
Load
Transaction size
Memory management

DWU

Starting point: Sizing by capacity



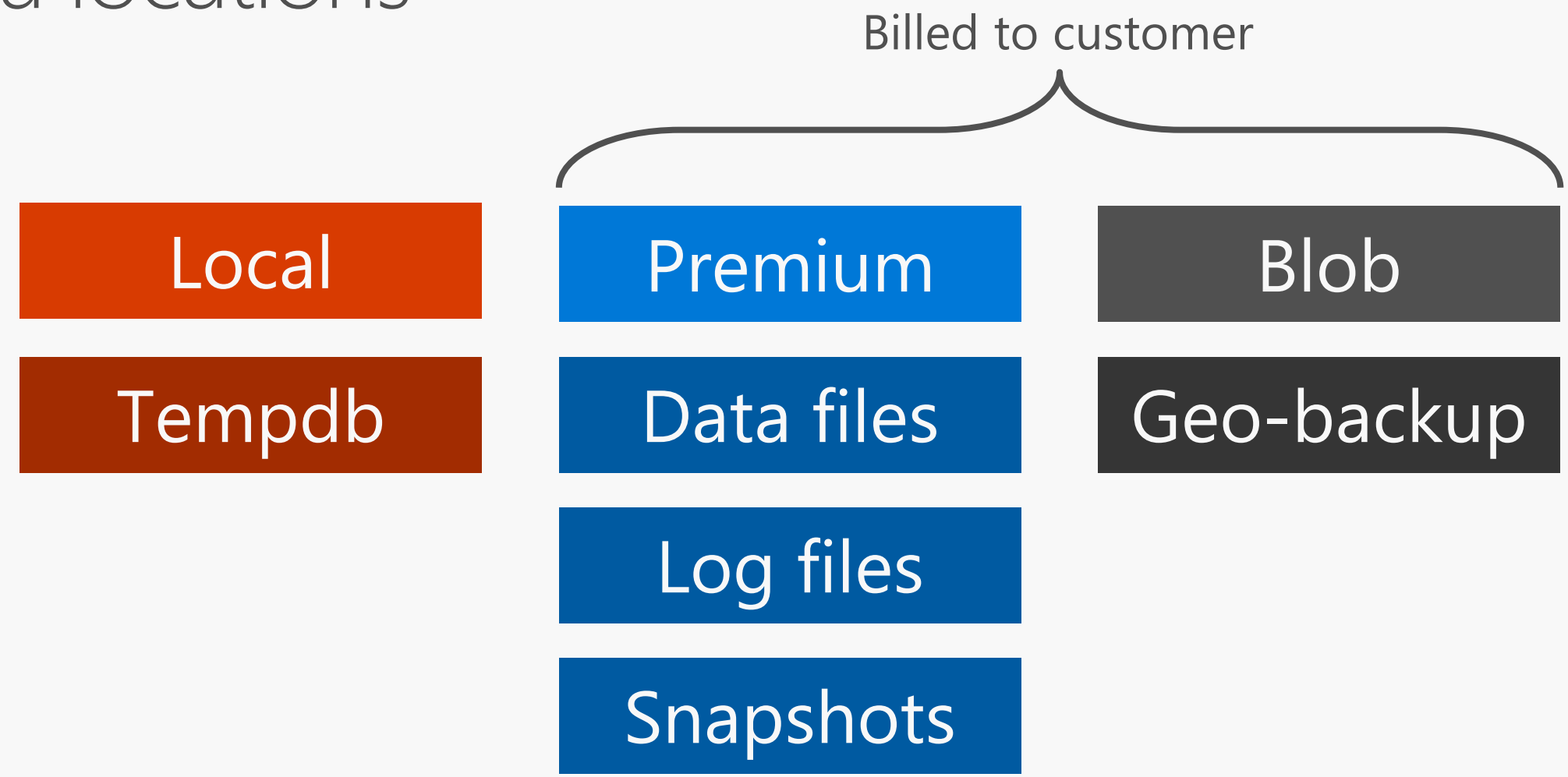
Storage tiers

Local storage

Premium storage (remote)

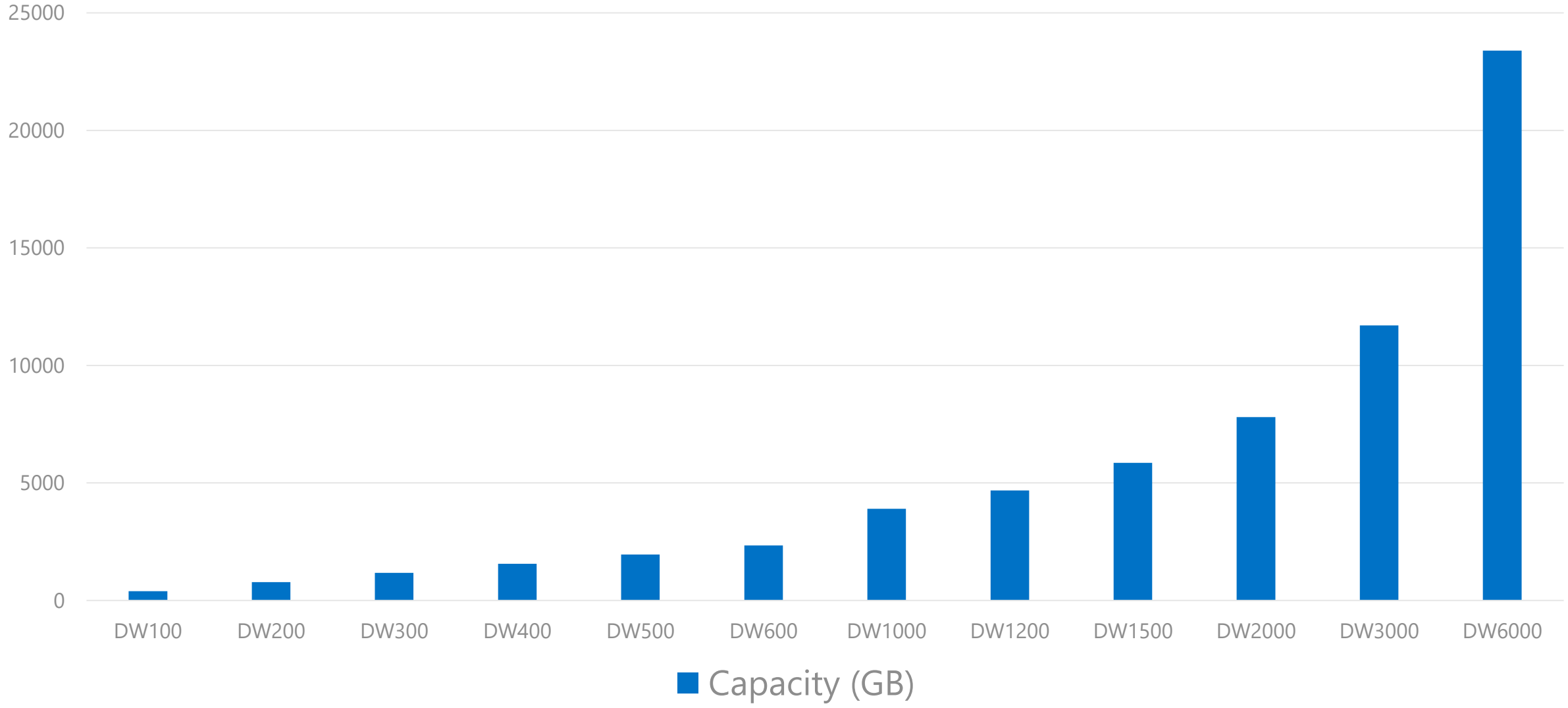
Blob storage (remote and geo redundant)

Data locations

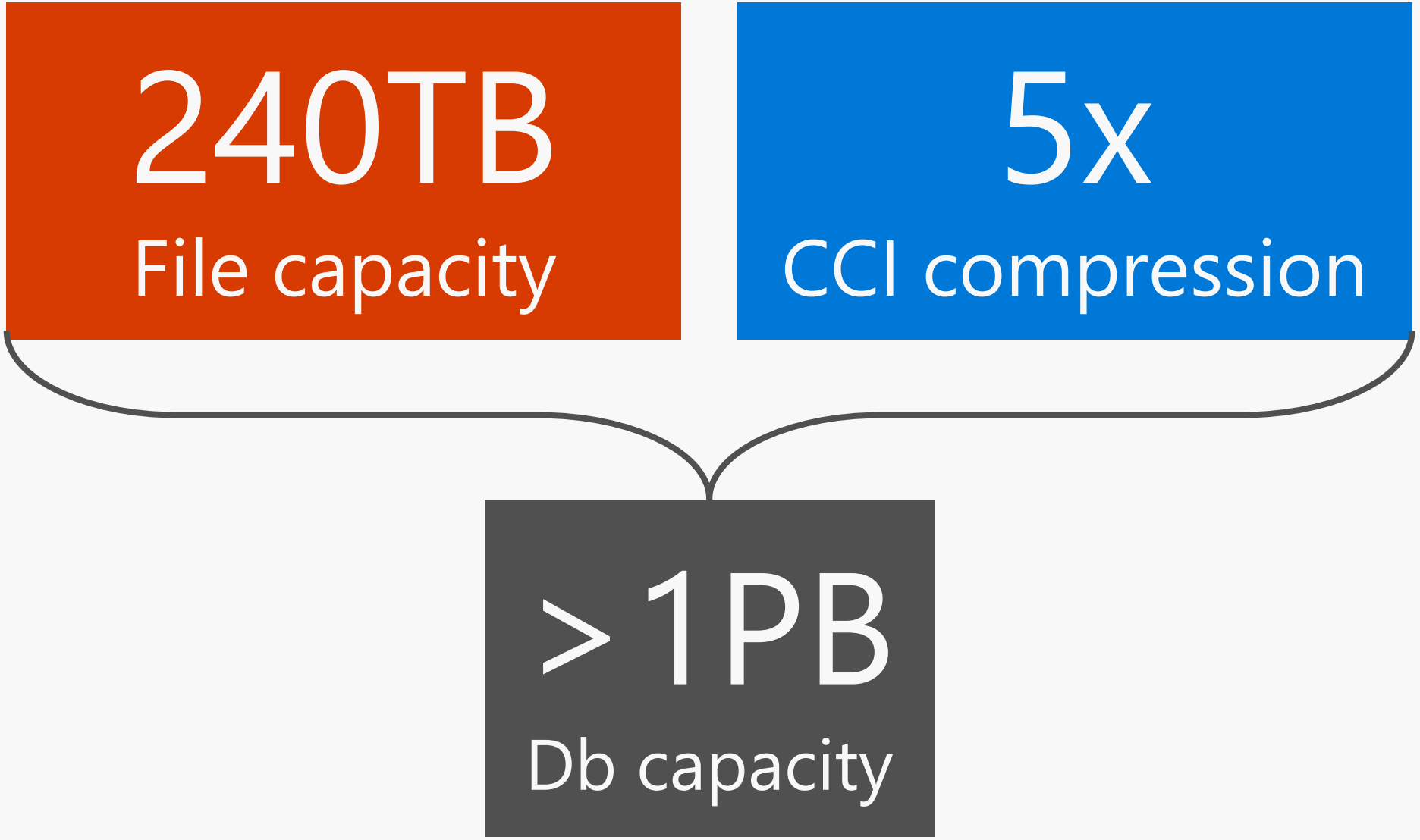


Local Storage: Tempdb sizing

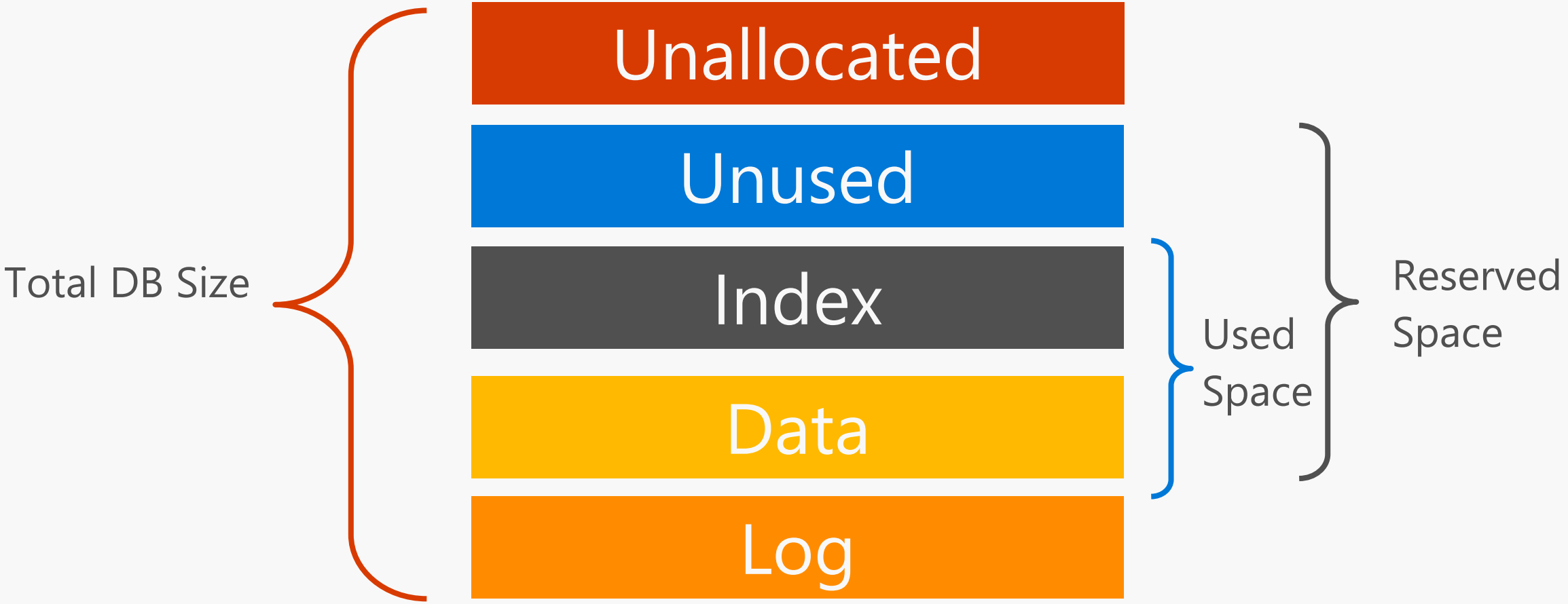
~399GB
per DW100



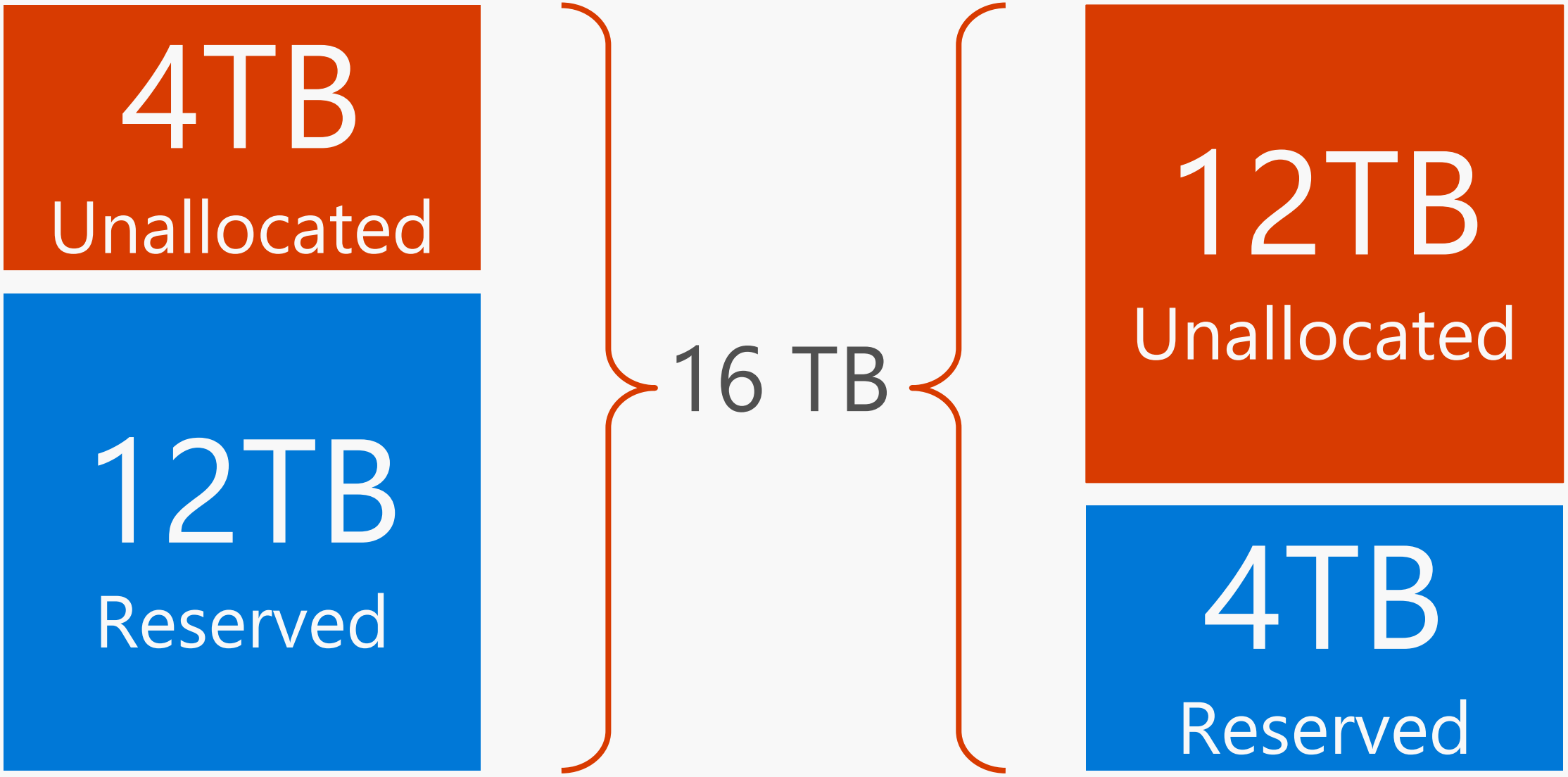
Premium Storage: Capacity limits



Premium Storage: Database Size



Premium Storage: Unallocated Space



Premium Storage: Snapshots

Frequency	Retention
4 hours	7 days

RPO : 8 Hours

Blob Storage: Geo-redundant backups

Frequency and Retention

1
Geo-backup

24hr
RPO

Essentials

Resource group jrjpartnerrg	Server name jrjpartner.database.windows.net
Status Online	Connection strings Show database connection strings
Location West US	Performance tier DataWarehouse (300 DWUs)
Subscription name ElascticScaleDev_657854	Geo-backup policy Enabled
Subscription ID	

Geo-backup policy

Save Discard Feedback

Geo-backup copies one of the automatic snapshots each day to RA-GRS storage. This can be used in an event of a disaster to recover your data warehouse to a new region.

Learn more

Geo-backup policy

Enabled Disabled

Most recent Geo-backup time

2016-07-25T06:25:39Z

Capping storage capacity

```
CREATE DATABASE MyDB COLLATE
SQL_Latin1_General_CP1_CI_AS
(
    EDITION              = 'DataWarehouse'
,   SERVICE_OBJECTIVE    = 'DW400'
,   MAXSIZE              = 10240 GB
);

ALTER DATABASE MyDB
MODIFY (MAXSIZE = 245760 GB);
```

Storage sizing summary

Database size:

sp_spaceused

Table sizing:

DMVs

Snapshot size:

Total storage size (portal) – database size (sp_spaceused)

Free space (unallocated):

sp_spaceused

Table sizing 1/2

```
WITH base AS
(
SELECT   sm.[name]                                                          AS  [schema_name]
,        tb.[name]                                                          AS  [table_name]
,        nt.[distribution_id]                                               AS  [distribution_id]
,        nt.[pdw_node_id]                                                   AS  [node_id]
,        nps.[partition_number]                                             AS  [partition_nmbr]
,        nps.[reserved_page_count]                                          AS  [reserved_space_page_count]
,        nps.[reserved_page_count] - nps.[used_page_count]                  AS  [unused_space_page_count]
,        nps.[in_row_data_page_count]
+        nps.[row_overflow_used_page_count] + nps.[lob_used_page_count]     AS  [data_space_page_count]
,        nps.[reserved_page_count]
- (nps.[reserved_page_count] - nps.[used_page_count])
- ([in_row_data_page_count]+[row_overflow_used_page_count]+[lob_used_page_count]) AS  [index_space_page_count]
,nps.[row_count]                                                            AS  [row_count]
from sys.schemas sm
join sys.tables  tb                          ON  sm.[schema_id]       = tb.[schema_id]
join sys.pdw_table_mappings tm               ON  tb.[object_id]       = tm.[object_id]
join sys.pdw_nodes_tables nt                 ON  tm.[physical_name]   = nt.[name]
join sys.dm_pdw_nodes pn                     ON  nt.[pdw_node_id]     = pn.[pdw_node_id]
join sys.dm_pdw_nodes_db_partition_stats nps ON  nt.[object_id]       = nps.[object_id]
                                             AND nt.[pdw_node_id]     = nps.[pdw_node_id]
                                             AND nt.[distribution_id] = nps.[distribution_id]
)
```

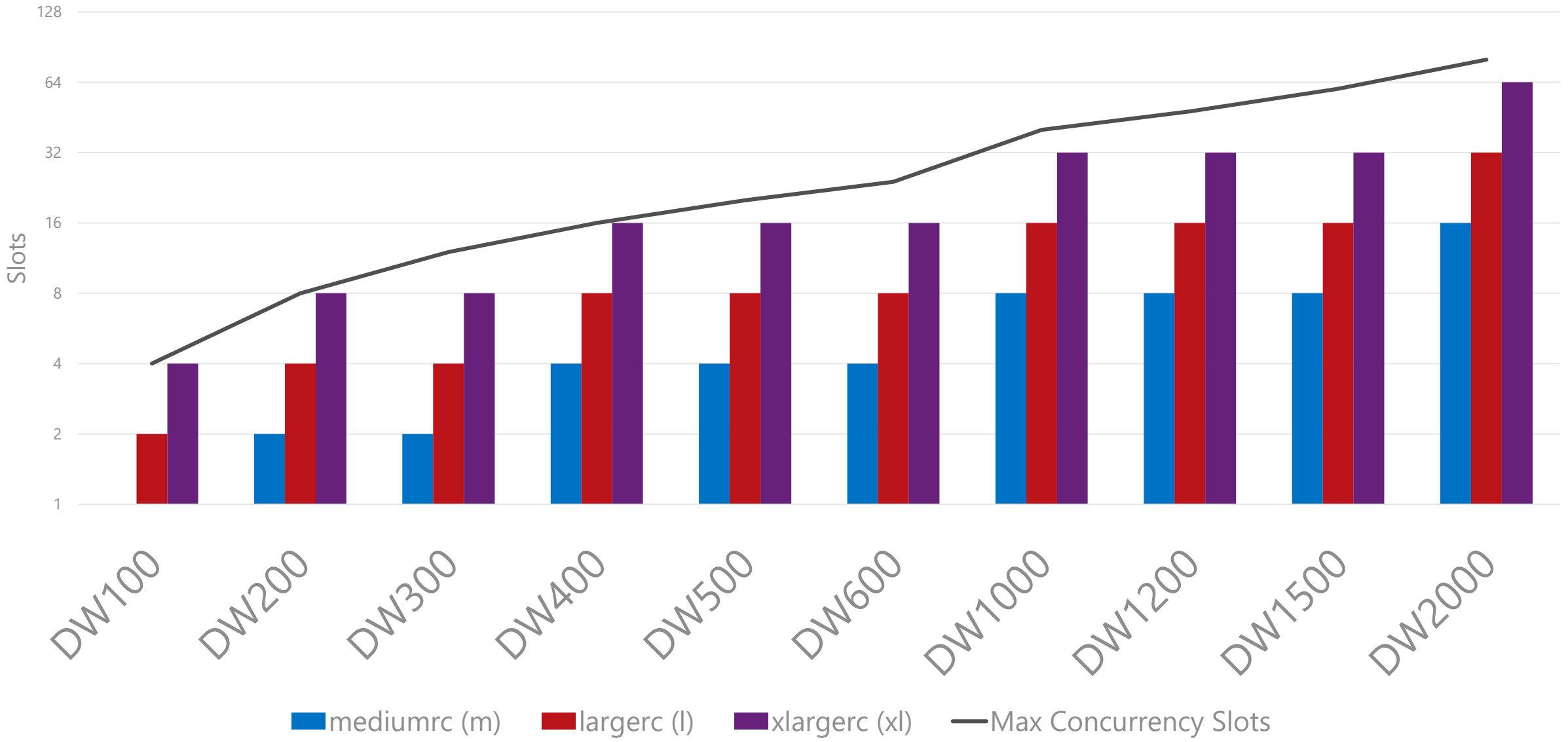
Table sizing 2/2

```
, size AS
(
SELECT
     [schema_name]
,    [table_name]
,    [row_count]
,    [distribution_id]
,    [node_id]
,    [partition_nmbr]
,    ([reserved_space_page_count] * 8.0)/1048576    AS [reserved_space_GB]
,    ([unused_space_page_count]   * 8.0)/1048576    AS [unused_space_GB]
,    ([data_space_page_count]     * 8.0)/1048576    AS [data_space_GB]
,    ([index_space_page_count]    * 8.0)/1048576    AS [index_space_GB]
FROM [base]
)
SELECT  *
FROM    [size]
WHERE   [schema_name] = 'cso'
AND     [table_name]  = 'FactOnlineSales'
;
```

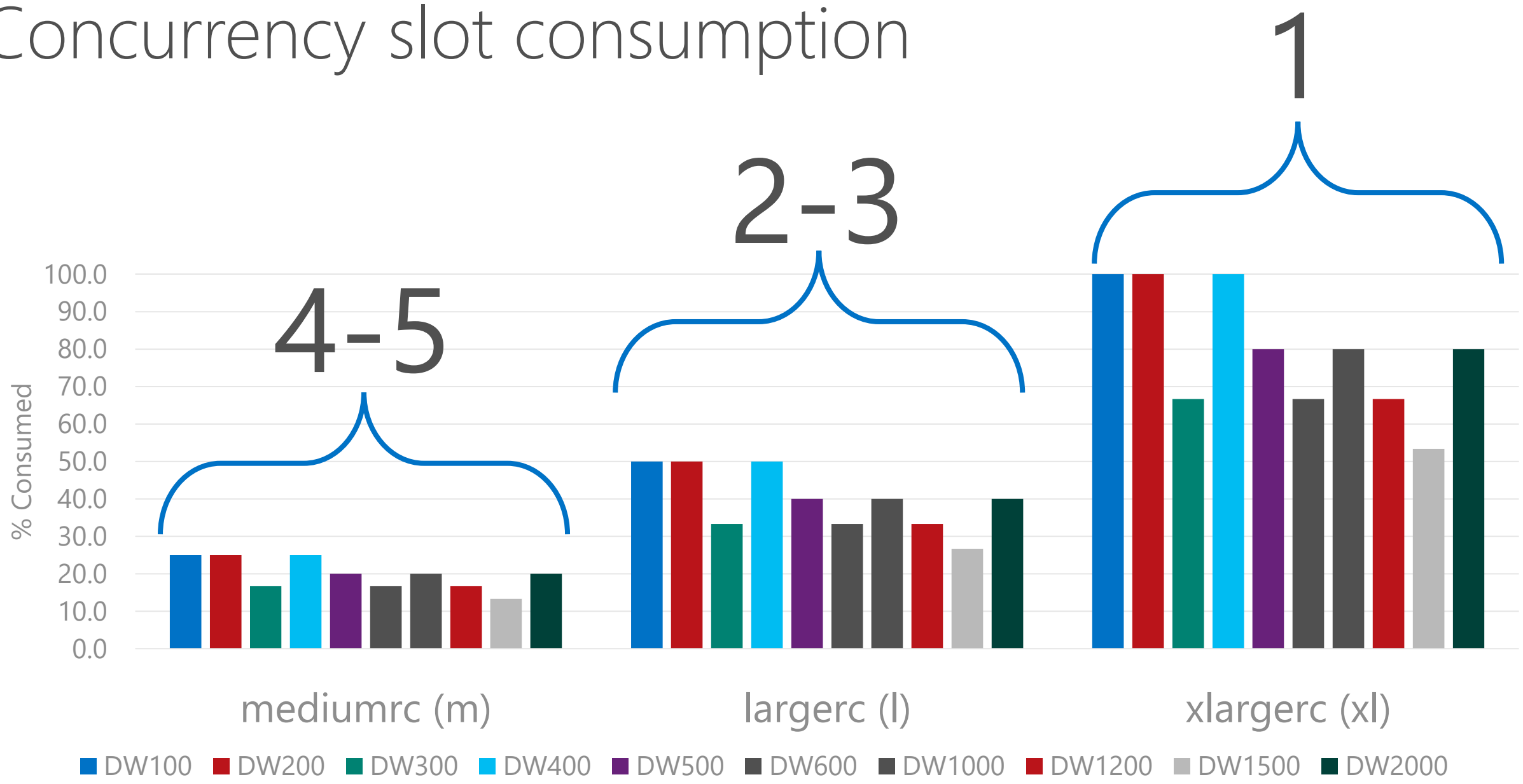
Concurrency

Concurrency: queries

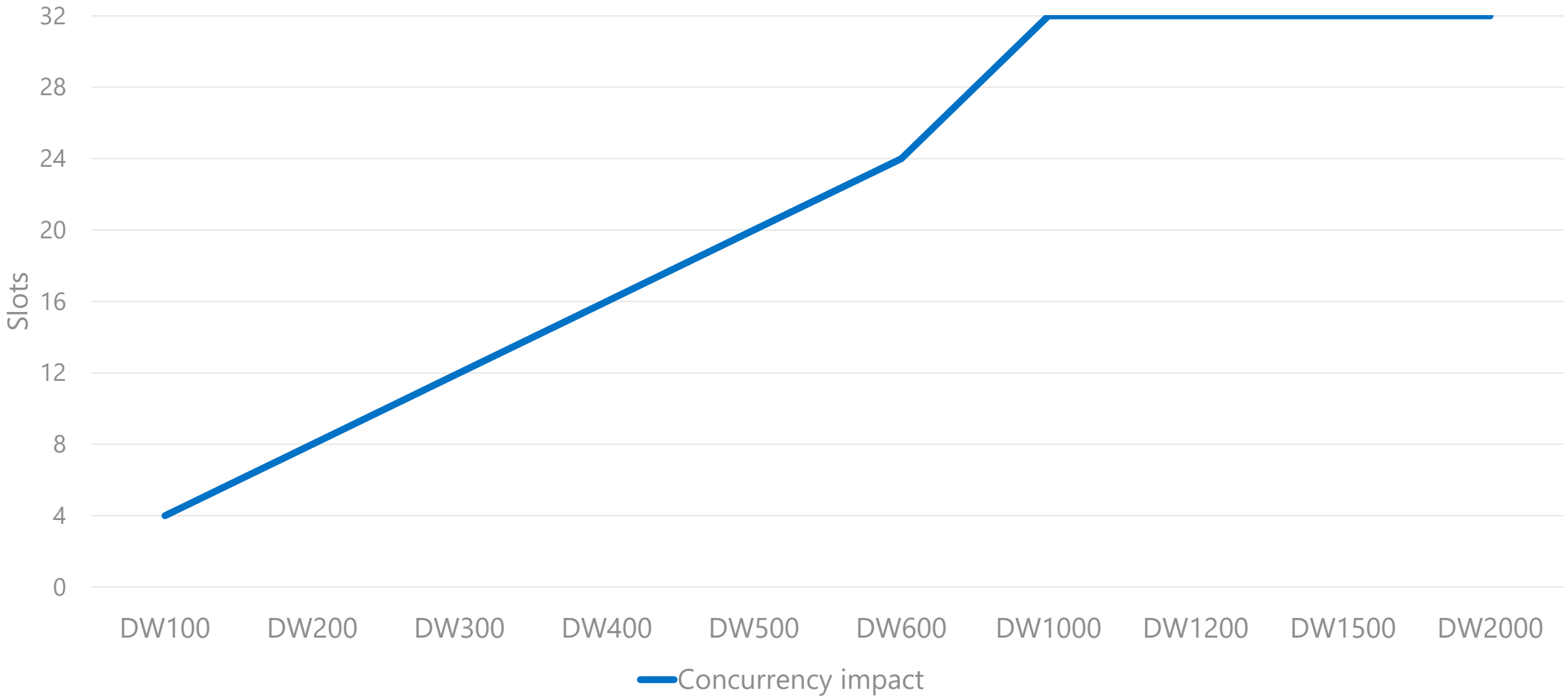
Concurrency: slots



Concurrency slot consumption



Concurrency impact: queries & slots



Loading

Sizing for the data load

Delimited text guidance.

- Evenly split the data into multiple files.
- One file per reader.
- Delimited text is the fastest.

Compressed text limits concurrent access to text files

Split data across files
OR
Use different file format

DWU	Readers	Writers
DW100	8	60
DW200	16	60
DW300	24	60
DW400	32	60
DW500	40	60
DW600	48	60
DW1000	60	60

Loading delimited text

A compressed text file cannot be read in parallel

Splitting data across multiple files maximises load performance

Data loading

Exception

Target Table = CI or NCI

Load user is defaultrc

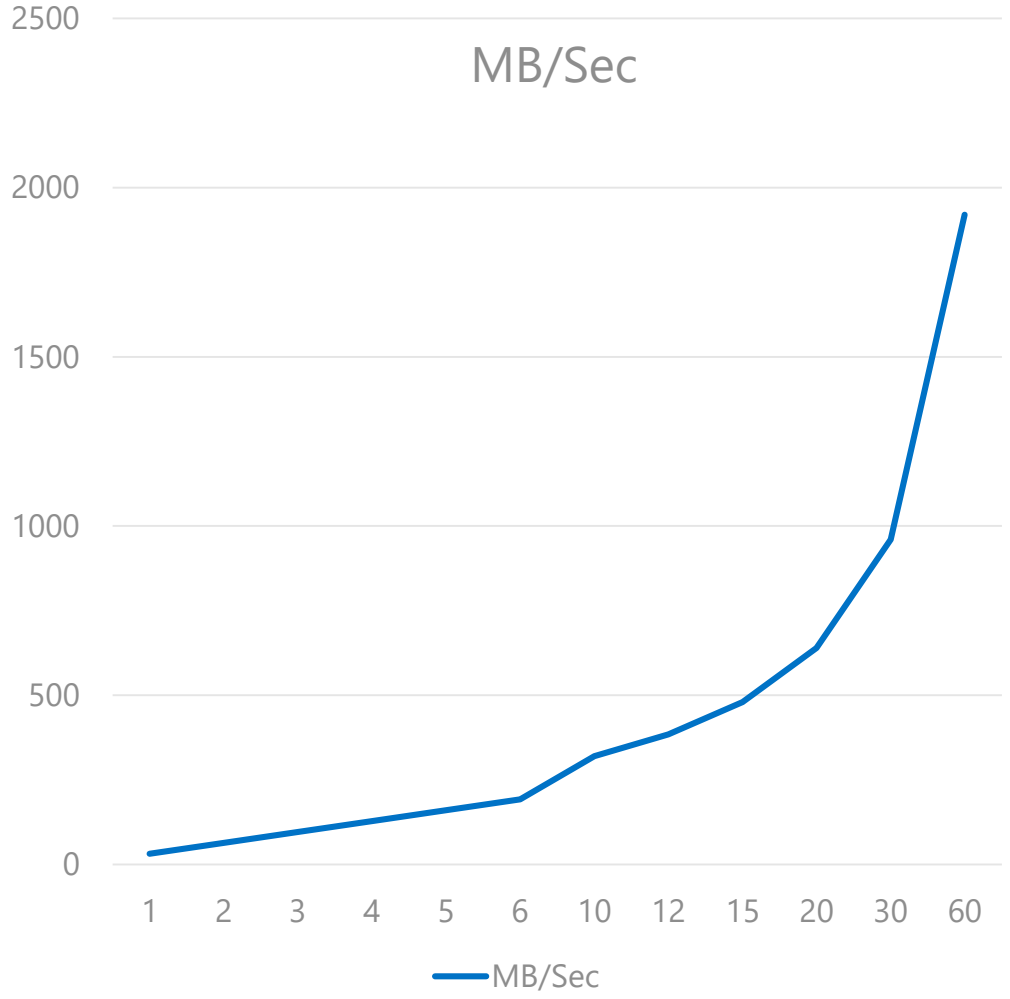
60 Writers

TAKE AWAY

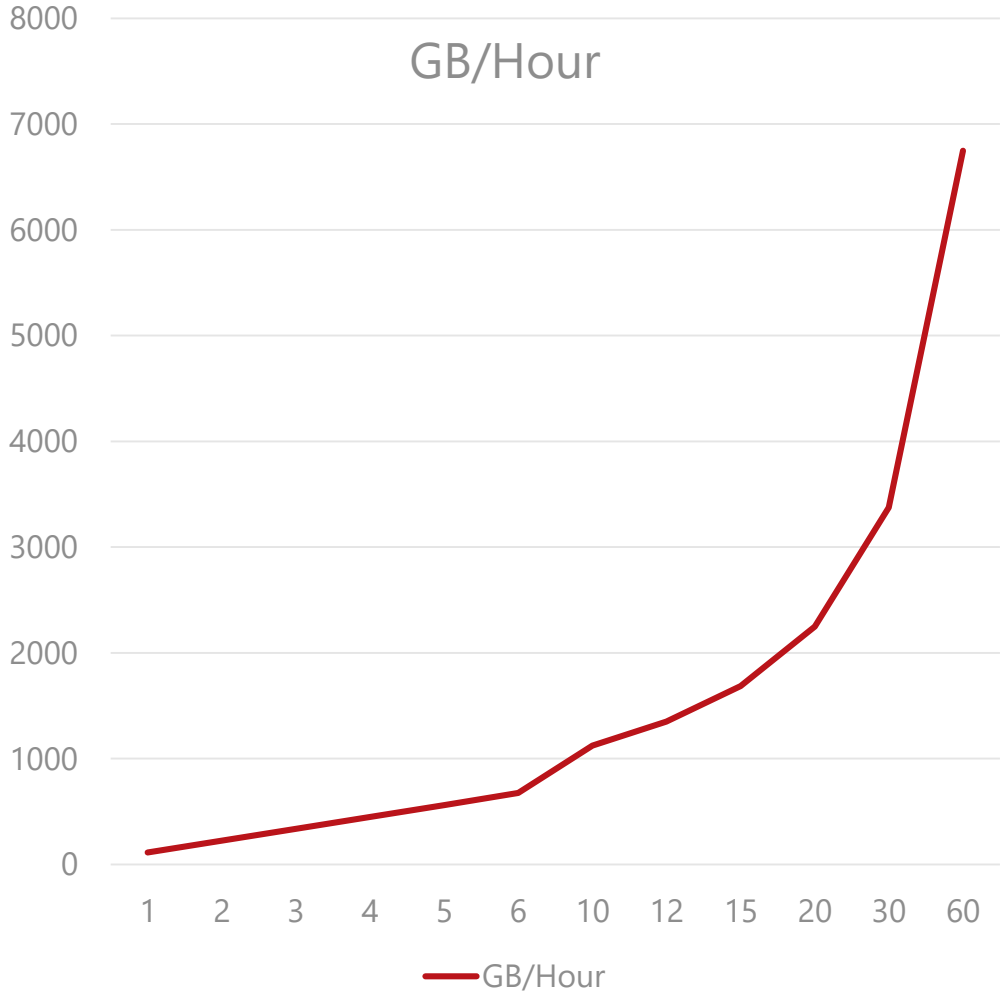
Use mediumrc+ for high DWU loads

DWU	Max External Readers	Max Writers
DW100	8	60
DW200	16	60
DW300	24	60
DW400	32	60
DW500	40	60
DW600	48	60
DW1000	80	80
DW1200	96	96
DW1500	120	120
DW2000	160	160
DW3000	240	240
DW6000	480	480

Load scaling



30-32 MB/Sec/Node



110-115 GB/Hour/Node

DWU (Data Warehouse Unit)

Introducing DWU

DWU
DW100
DW200
DW300
DW400
DW500
DW600
DW1000
DW1200
DW1500
DW2000

CPU

RAM

I/O

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

T-SQL

```
CREATE DATABASE MyDB COLLATE
SQL_Latin1_General_CP1_CI_AS
(
    EDITION              = 'DataWarehouse'
,   SERVICE_OBJECTIVE    = 'DW400'
,   MAXSIZE              = 10240 GB
)
;
```

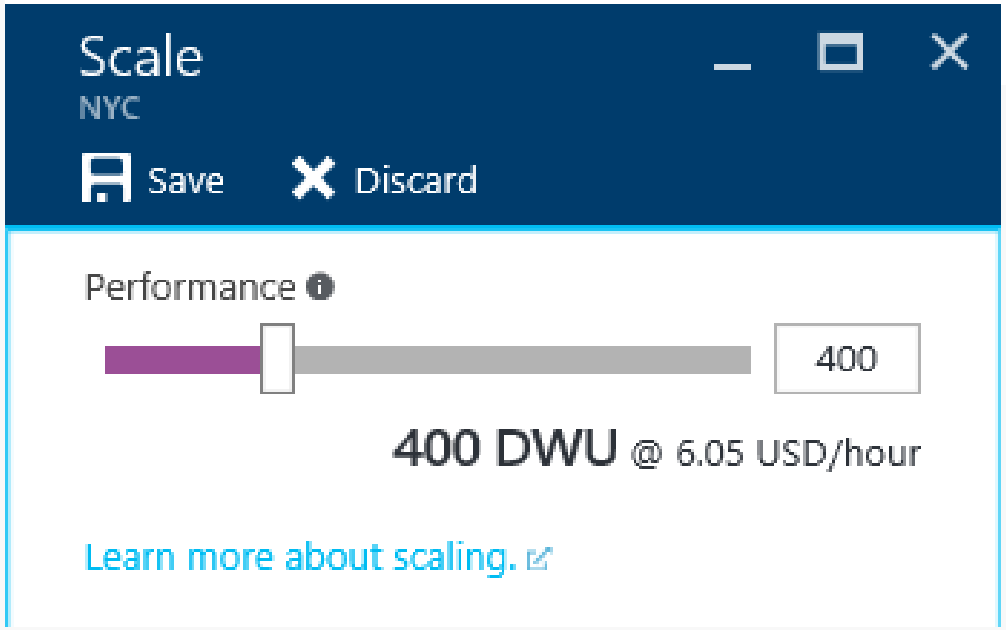
Sizing by storage capacity?

1TB / DWU100 is good place to start

Introducing Data Warehouse Units

DWU
DW100
DW200
DW300
DW400
DW500
DW600
DW1000
DW1200
DW1500
DW2000
DW3000
DW6000

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

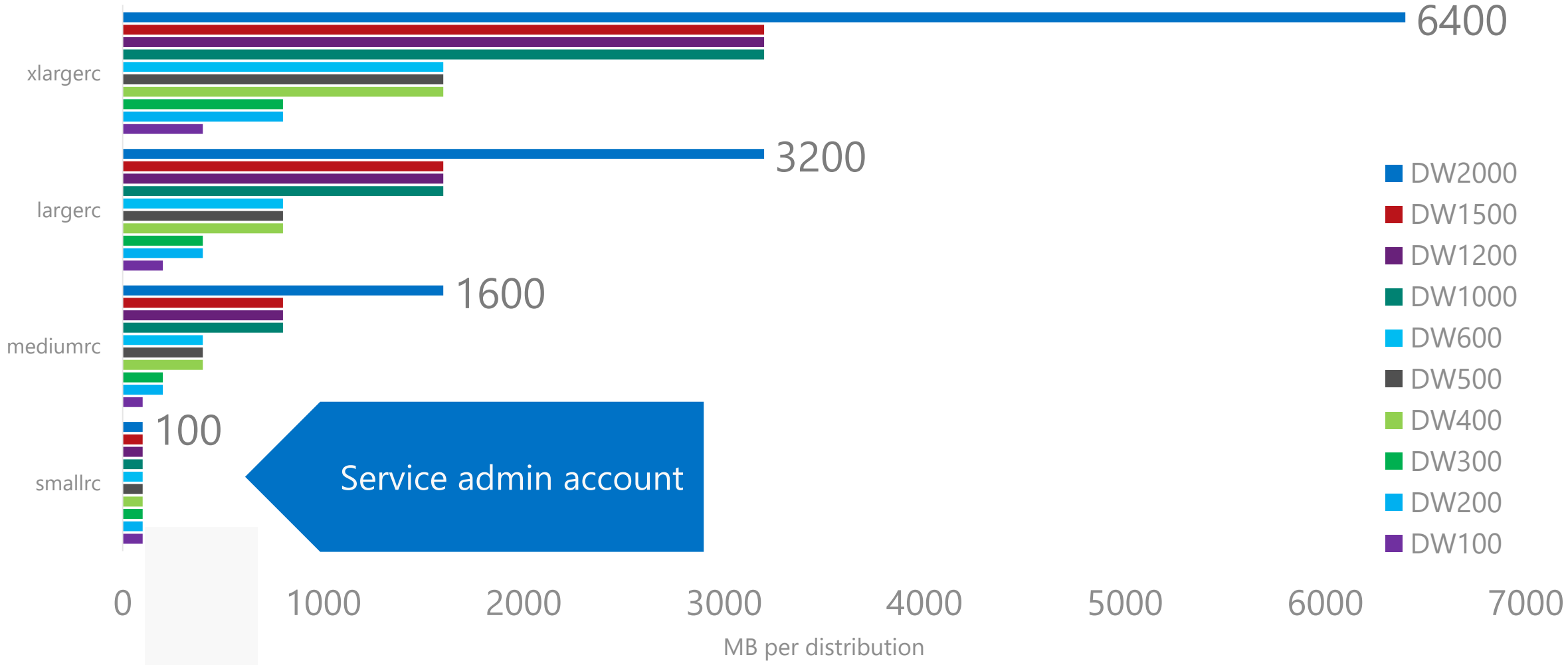


CPU

RAM

I/O

Memory Management (MB per distribution)



Memory grant sizing factors

Target rows in the rowgroup

Table Overhead

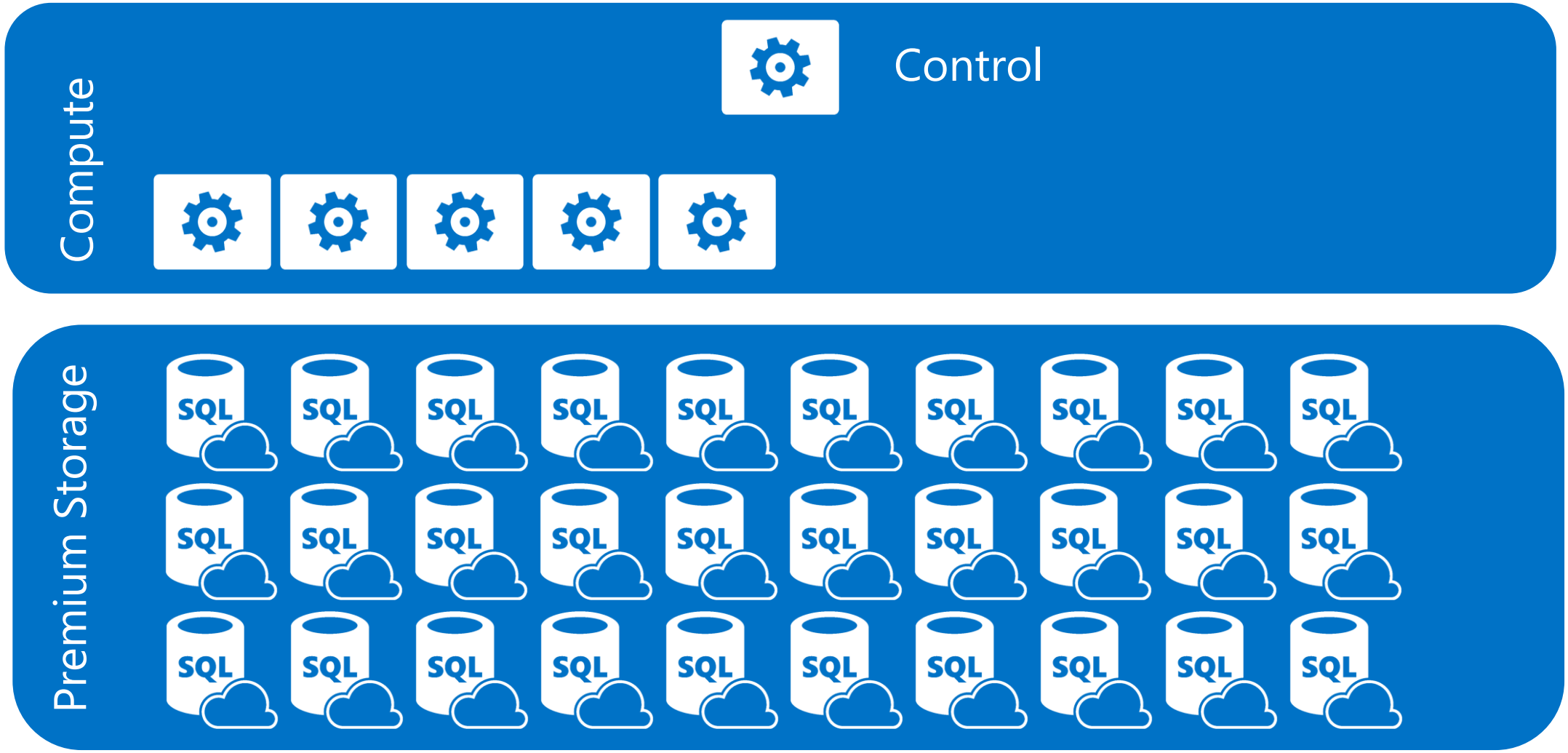
#columns

#short string character typed columns

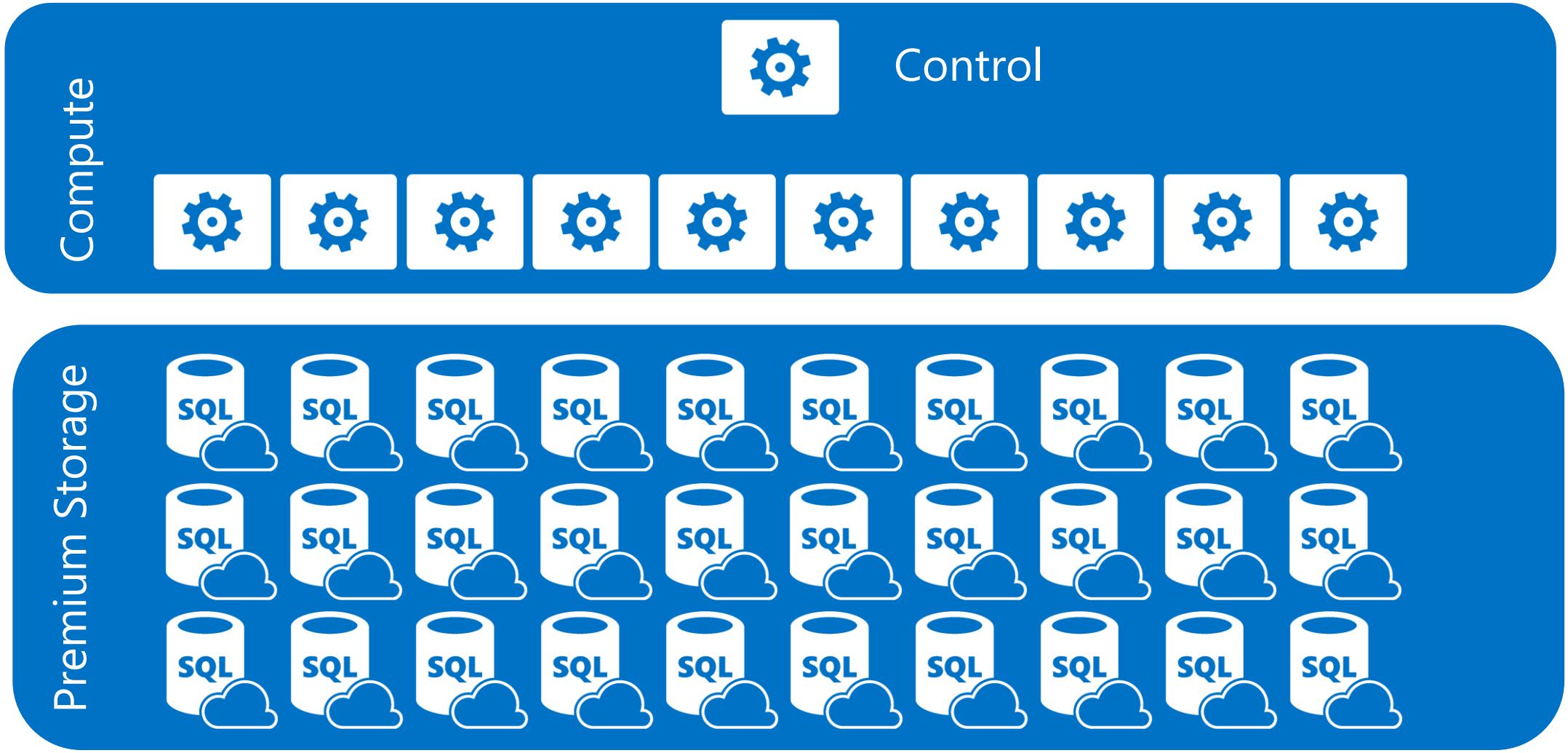
#long string character typed columns

Differentiating technical capabilities

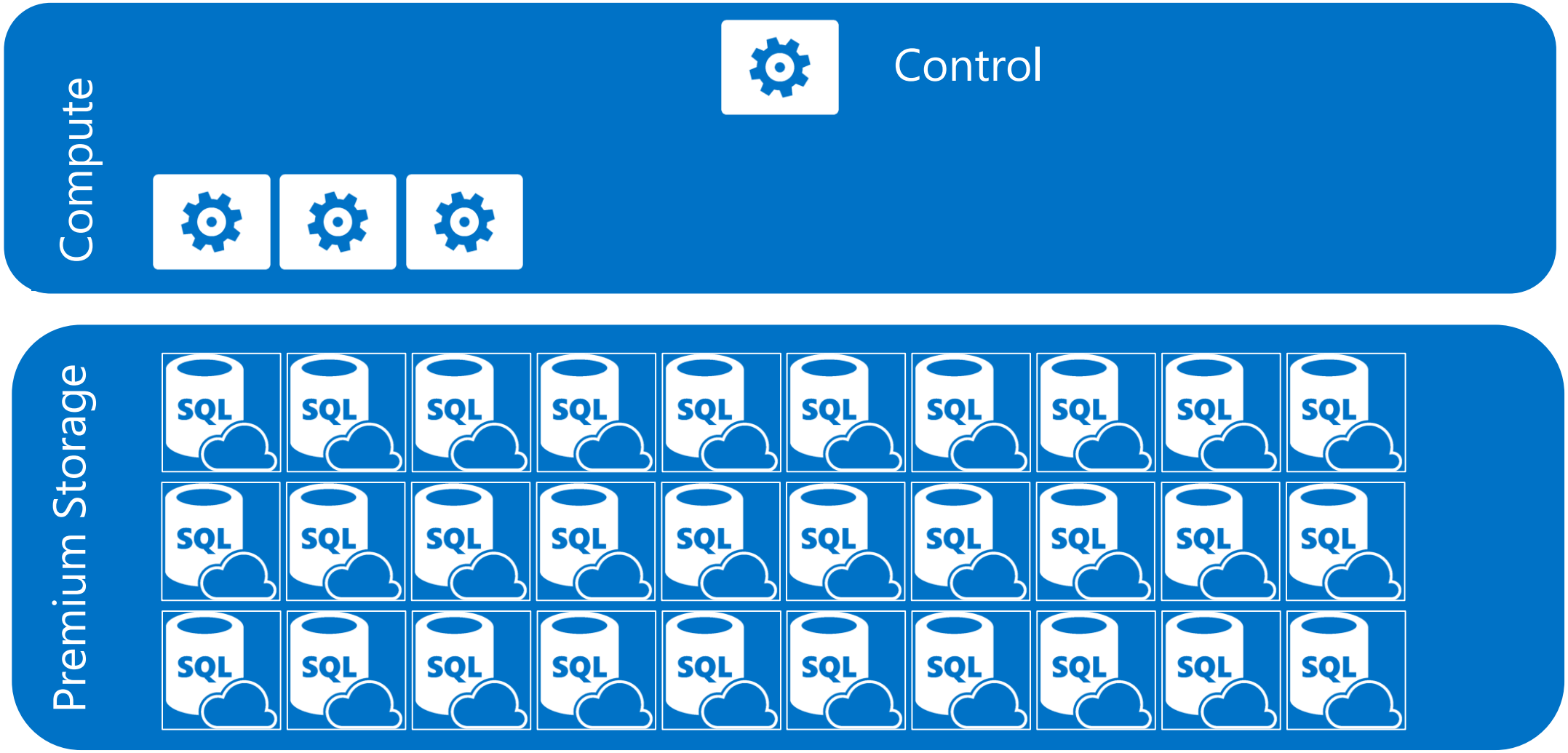
Separate compute from storage



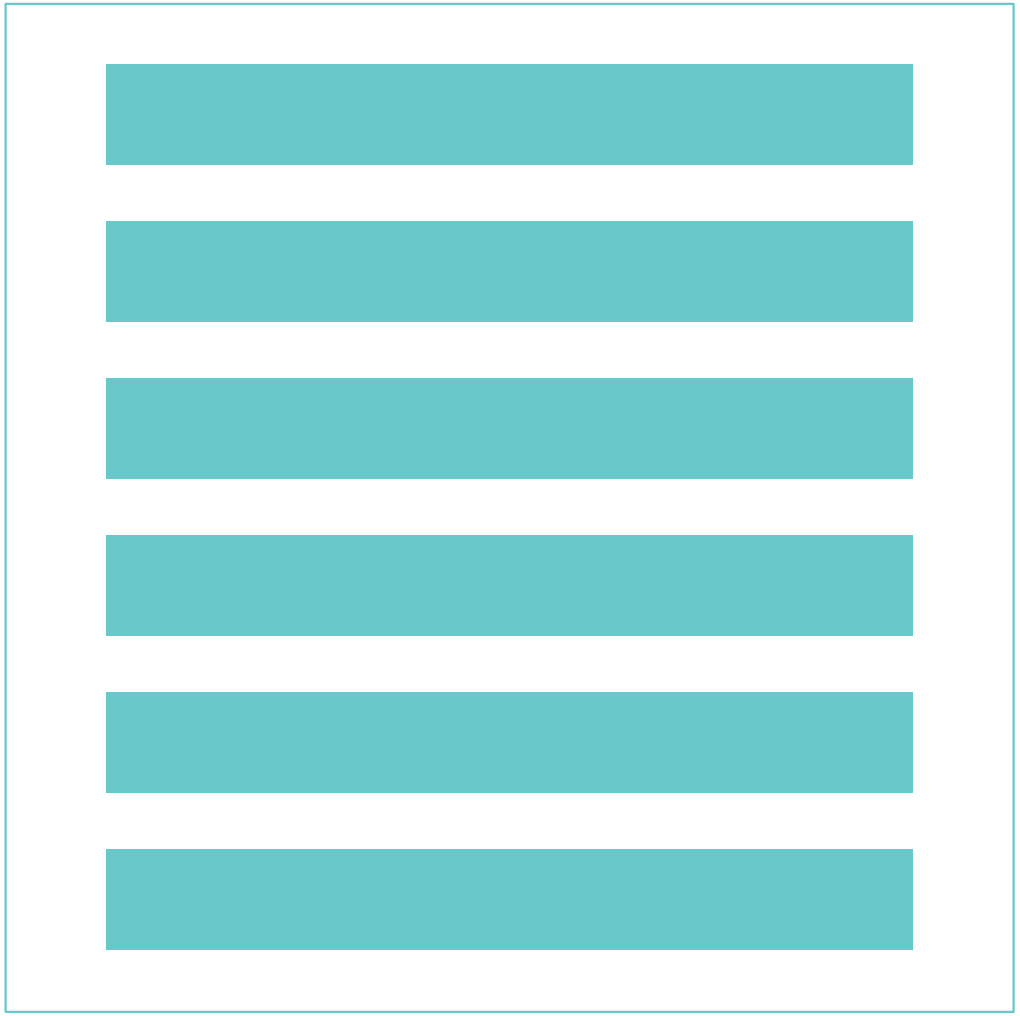
Independently scale compute

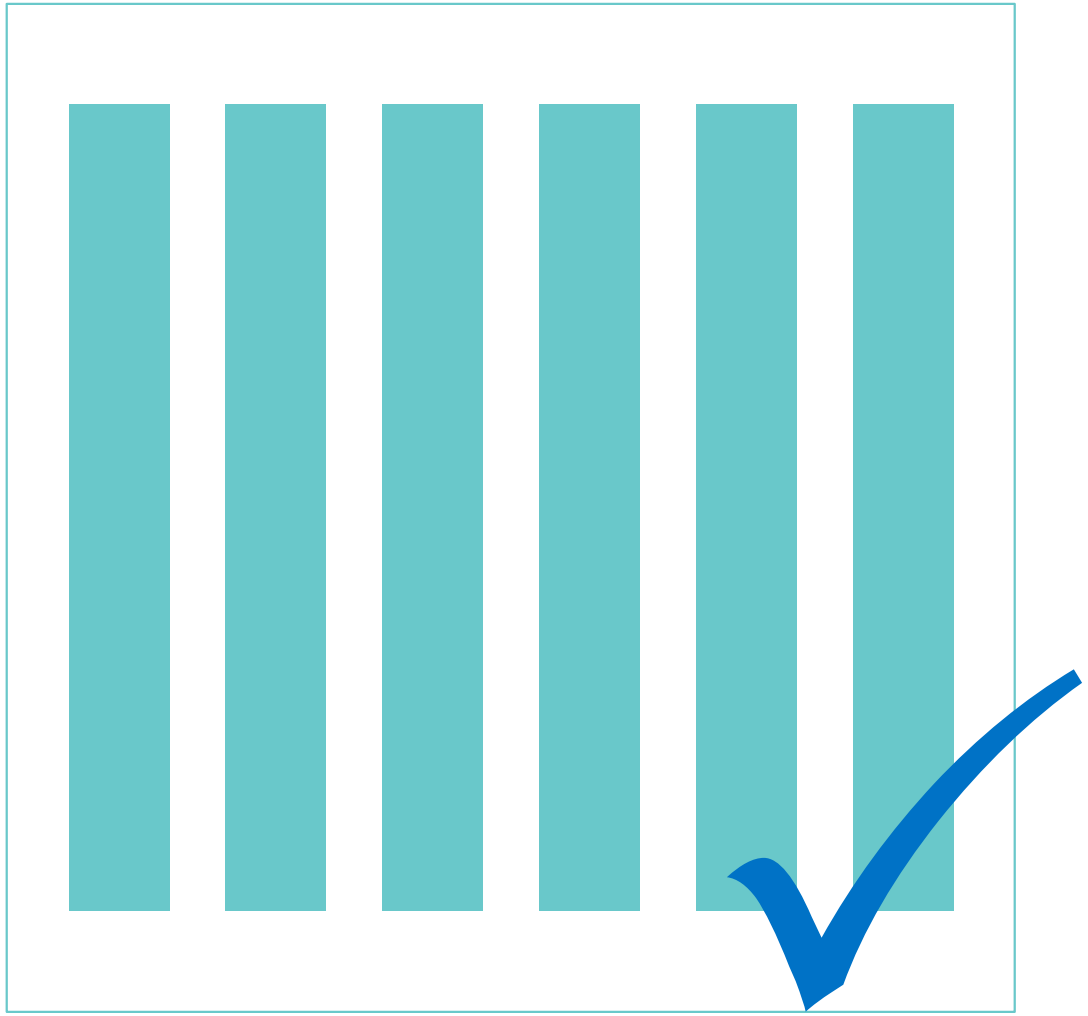


Pause and resume workload

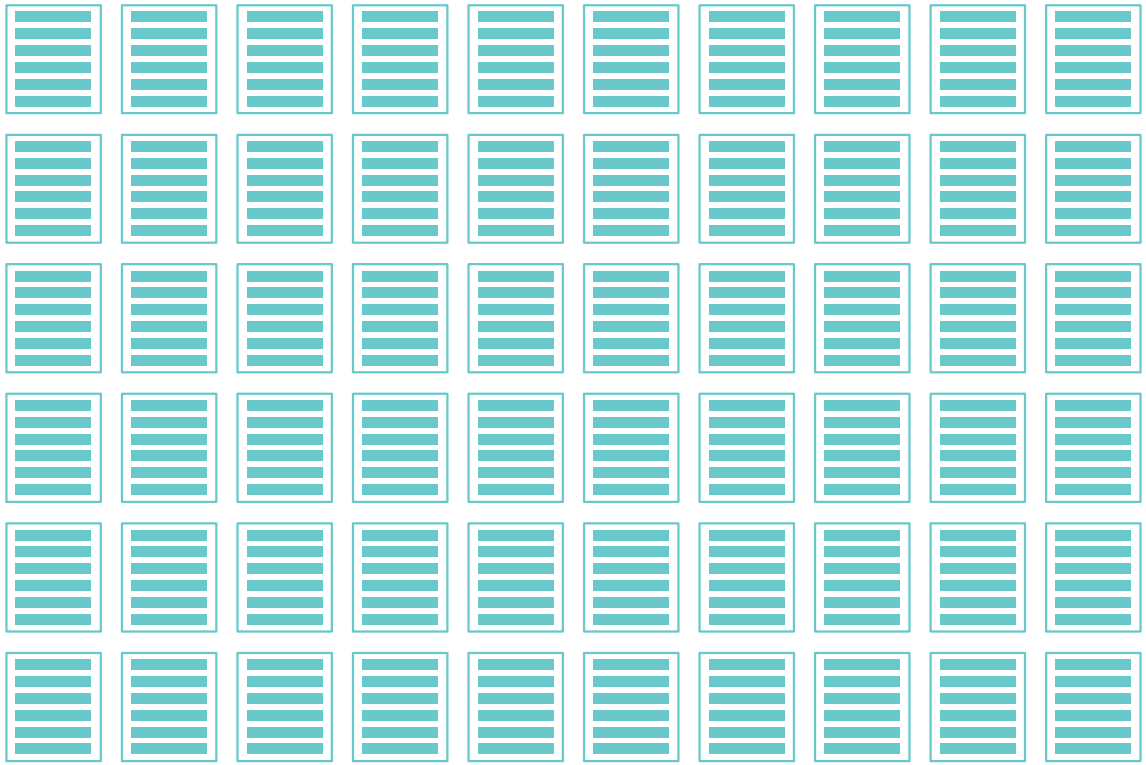


Supports Rowstore & Columnstore

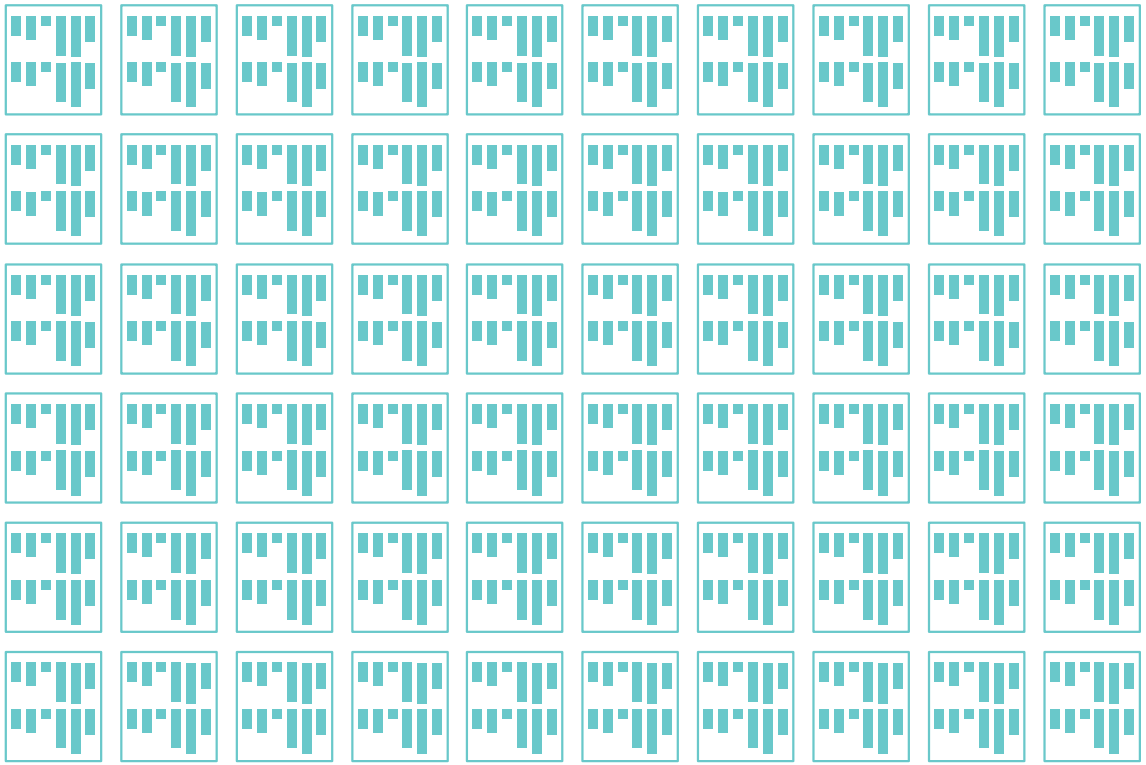




Scale out = Distributed tables

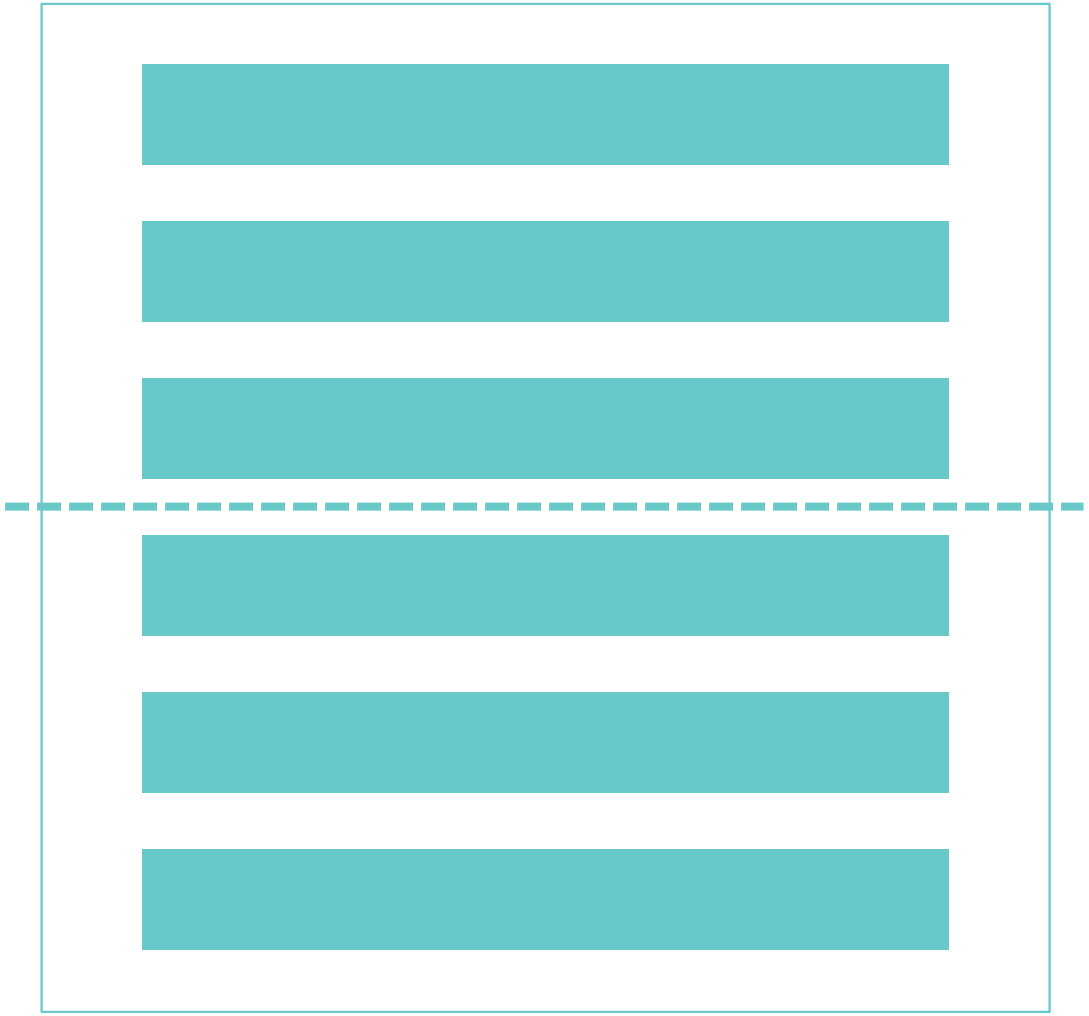


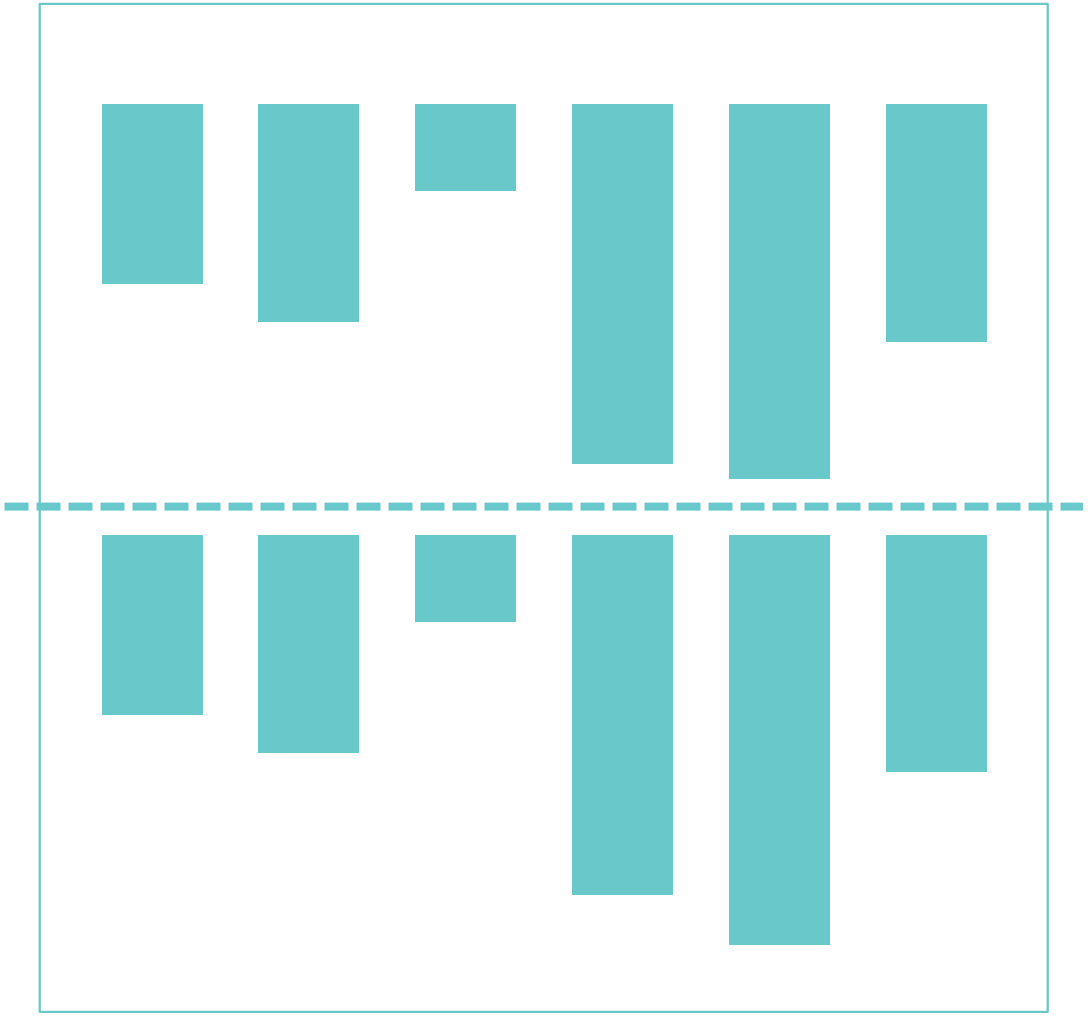
ROW STORE



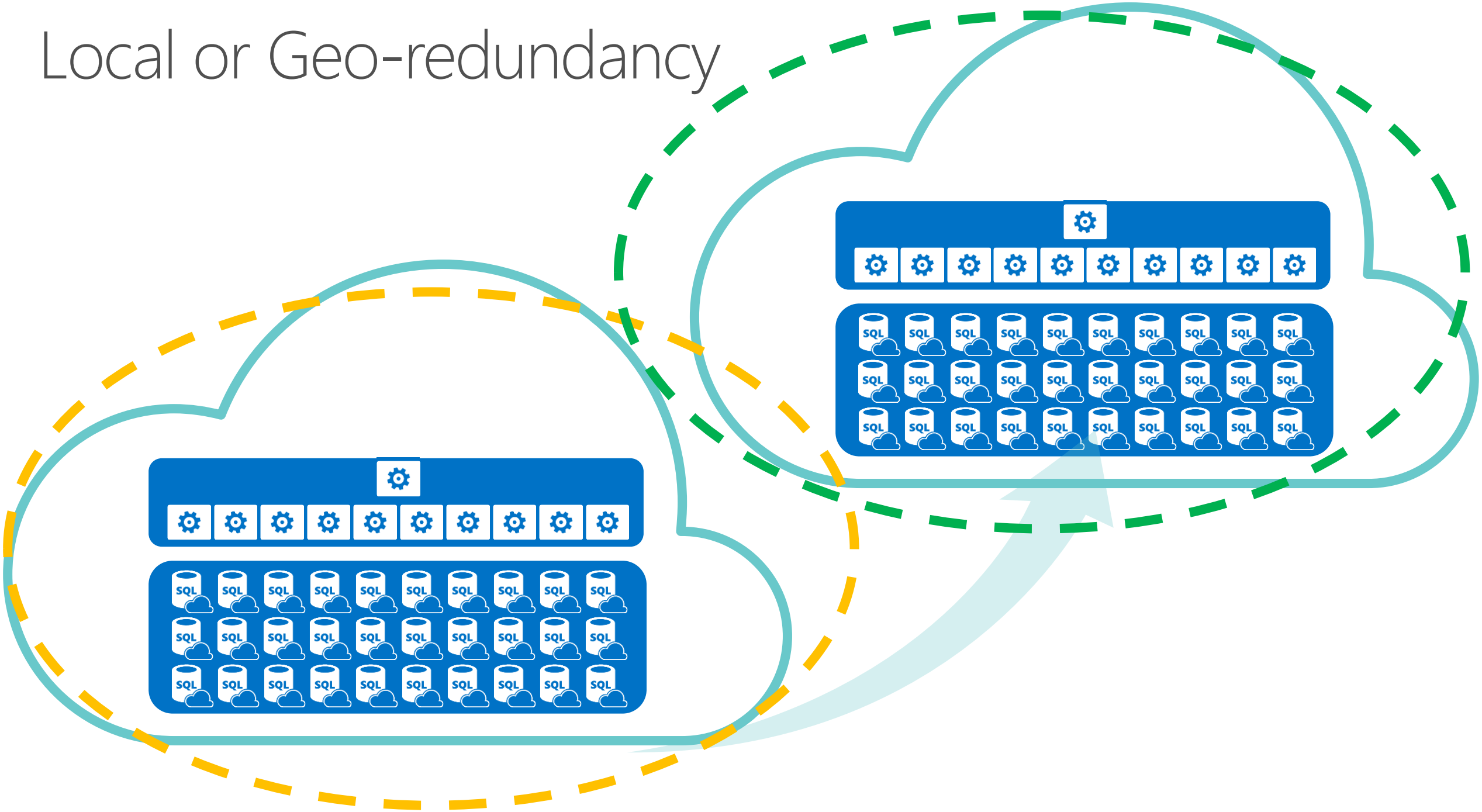
COLUMN STORE

Row & Column Store & Partition

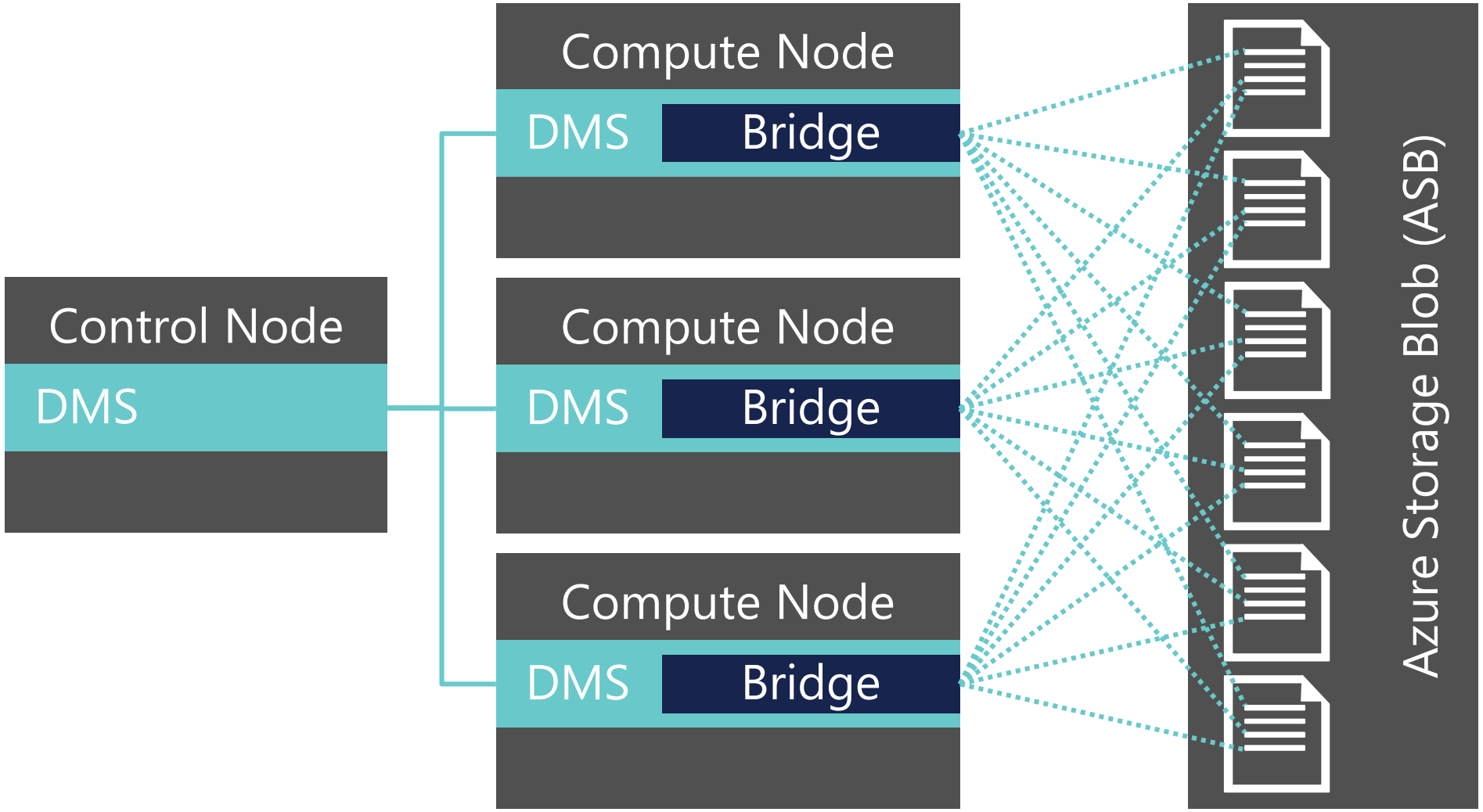




Local or Geo-redundancy



Parallel Loading with PolyBase



Azure SQLDW Take-aways

No CAPEX

Low OPEX

Provision in just 5 minutes

Scale in seconds

Fully parallel load

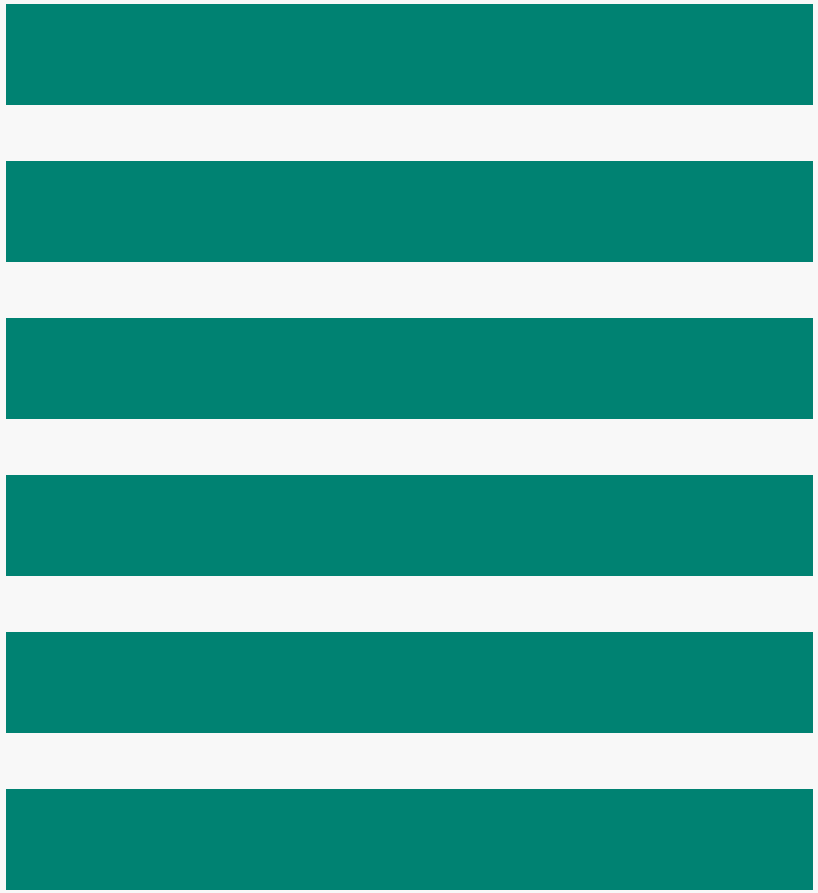
Fully managed platform

Time to insight measured in minutes

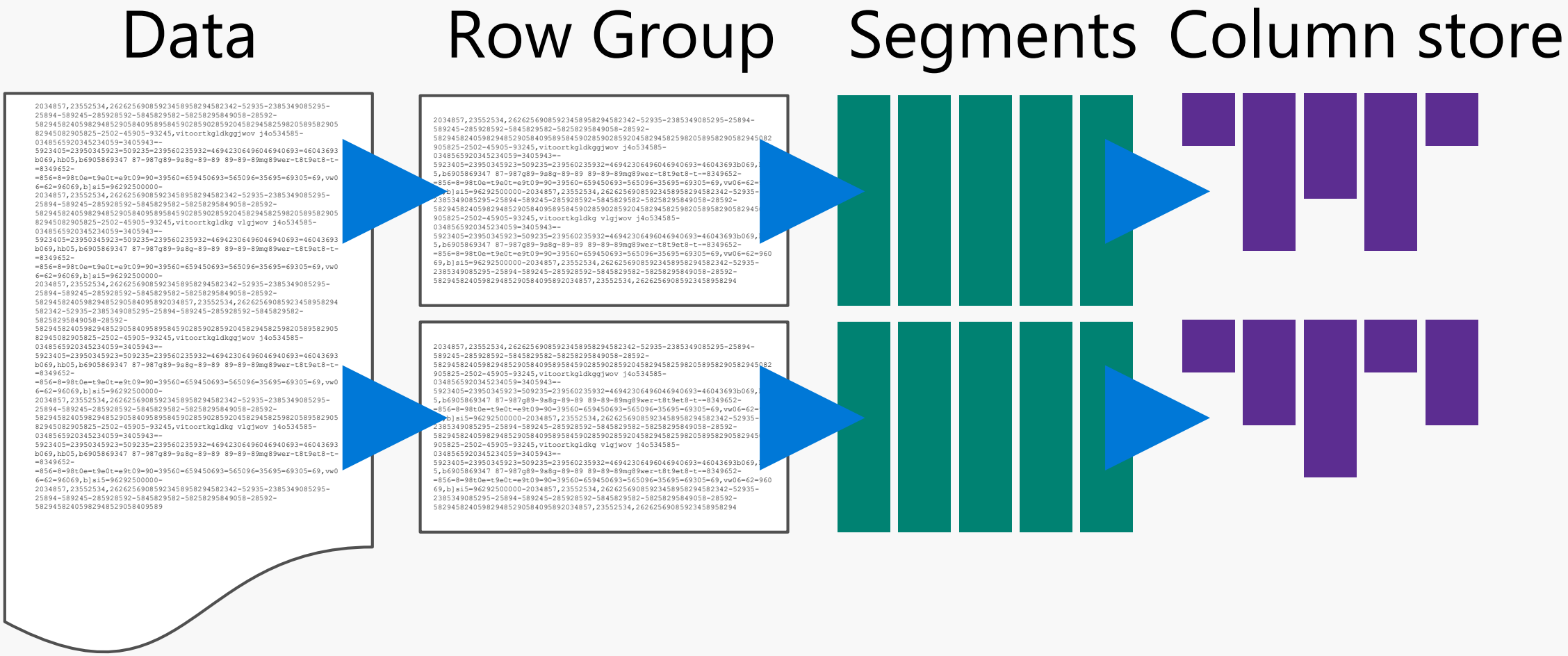
Available with a one month free trial

Developing with Azure SQL DW

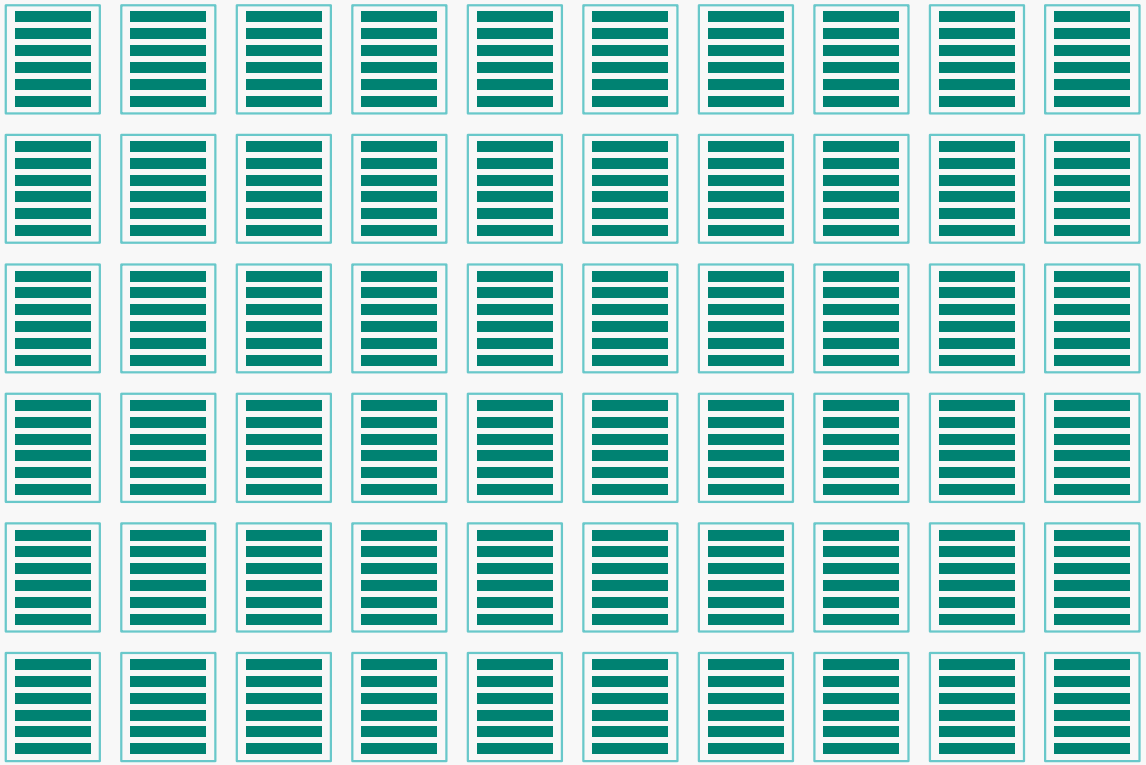
Row store & Column store



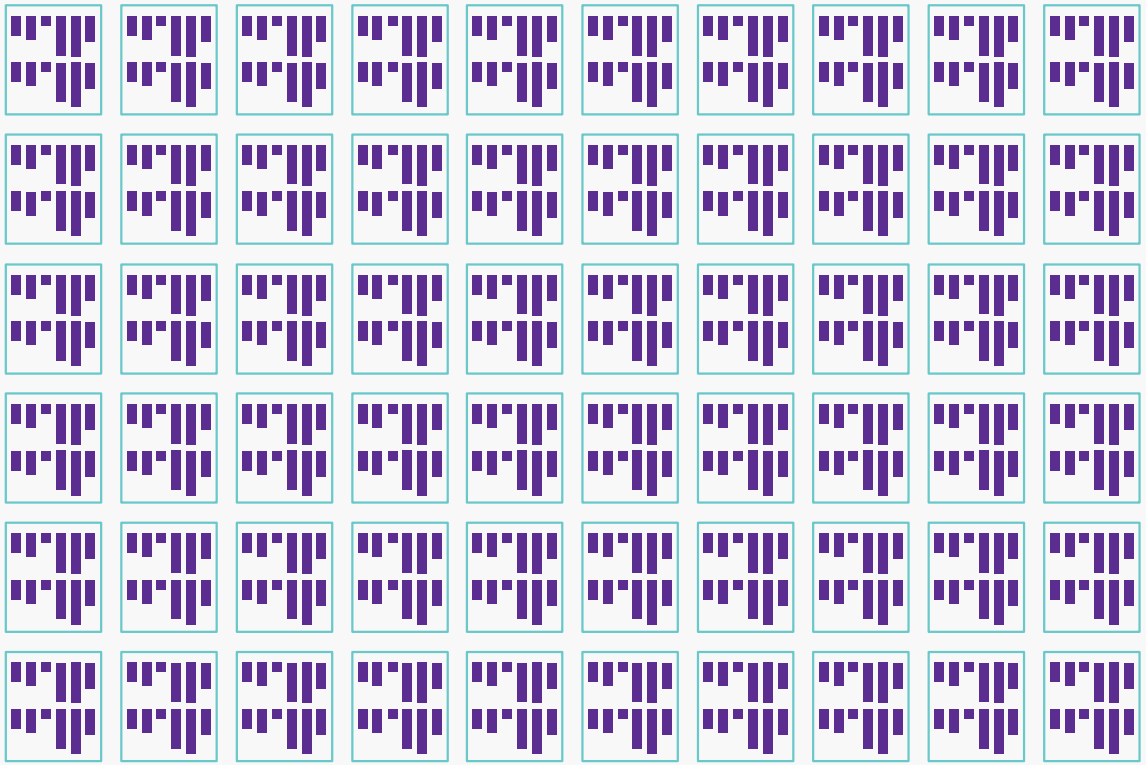
Column store taxonomy



Scale out = Distributed tables



ROW STORE



COLUMN STORE

Creating tables

```
CREATE TABLE [dbo].[DimStore]
(
    [StoreKey]          int             NOT NULL
,   [GeographyKey]      int             NOT NULL
,   [StoreName]         nvarchar(100)   NOT NULL
,   [StoreType]         nvarchar(15)        NULL
,   [StoreDescription]  nvarchar(300)   NOT NULL
,   [Status]            nvarchar(20)    NOT NULL
,   [OpenDate]          datetime        NOT NULL
,   [CloseDate]         datetime            NULL
,   [ETLLoadID]         int                 NULL
,   [LoadDate]          datetime            NULL
,   [UpdateDate]        datetime            NULL
)
WITH
(   CLUSTERED INDEX([StoreKey])
,   DISTRIBUTION = ROUND_ROBIN
)
;
```

Row

```
CREATE TABLE [dbo].[FactOnlineSales]
(
    [OnlineSalesKey]        int             NOT NULL
,   [DateKey]               datetime        NOT NULL
,   [StoreKey]              int             NOT NULL
,   [ProductKey]            int             NOT NULL
,   [PromotionKey]          int             NOT NULL
,   [CurrencyKey]           int             NOT NULL
,   [CustomerKey]           int             NOT NULL
,   [SalesOrderNumber]      nvarchar(20)    NOT NULL
,   [SalesOrderLineNumber]  int                 NULL
,   [SalesQuantity]         int             NOT NULL
,   [SalesAmount]           money           NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([ProductKey])
)
;
```

Column

Distribution

Guidance: Table Design

Row store: small tables & dimensions

<60 million rows

Round Robin most tables

Exception: shared hash distribution key is available in dimension

Column store: fact tables

Hash distribute table (ideally)

Column contains static values

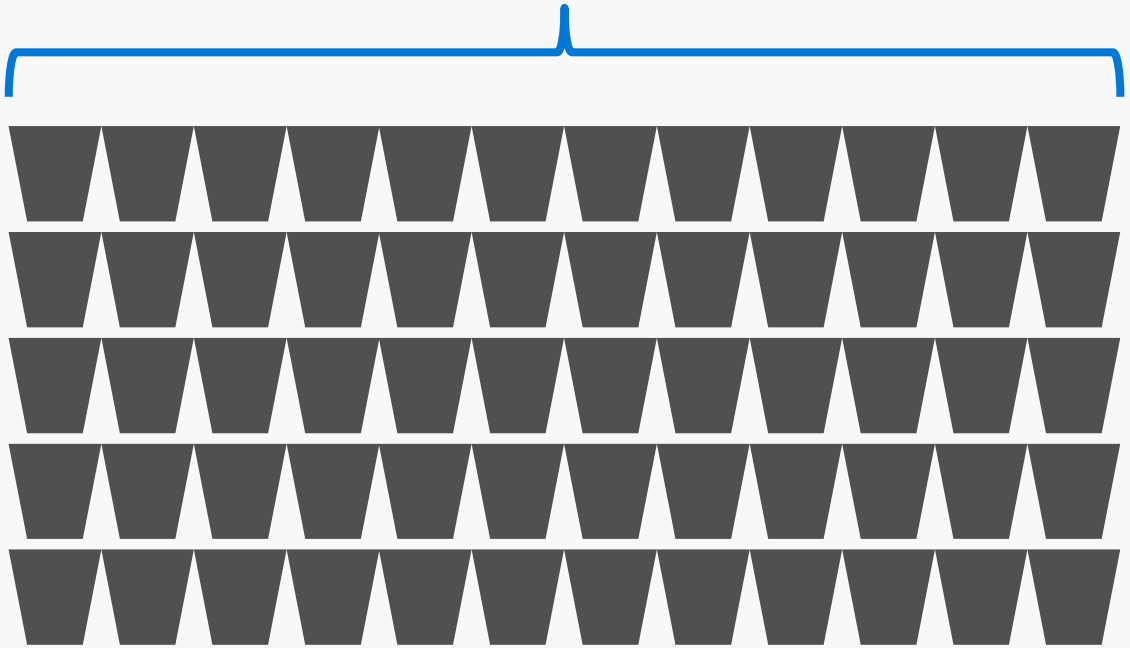
Column does not contain NULL values

Large number of distinct values (600+)

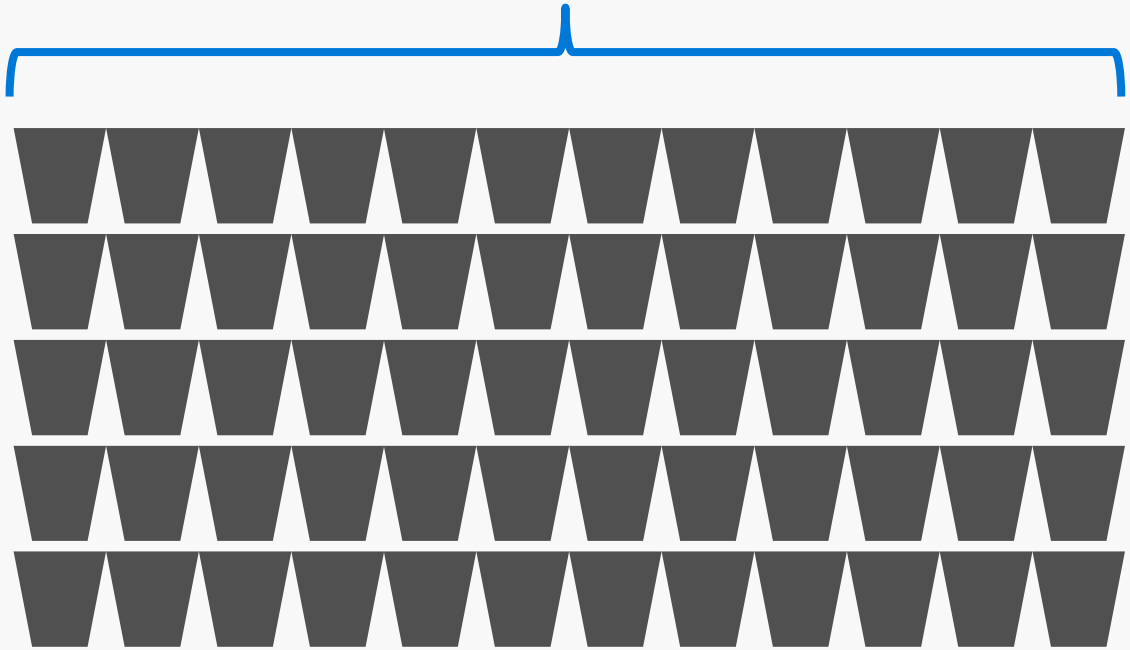
Even distribution of rows across the distinct values

Joining distributed tables: round robin to hash

DimProduct ROUND_ROBIN



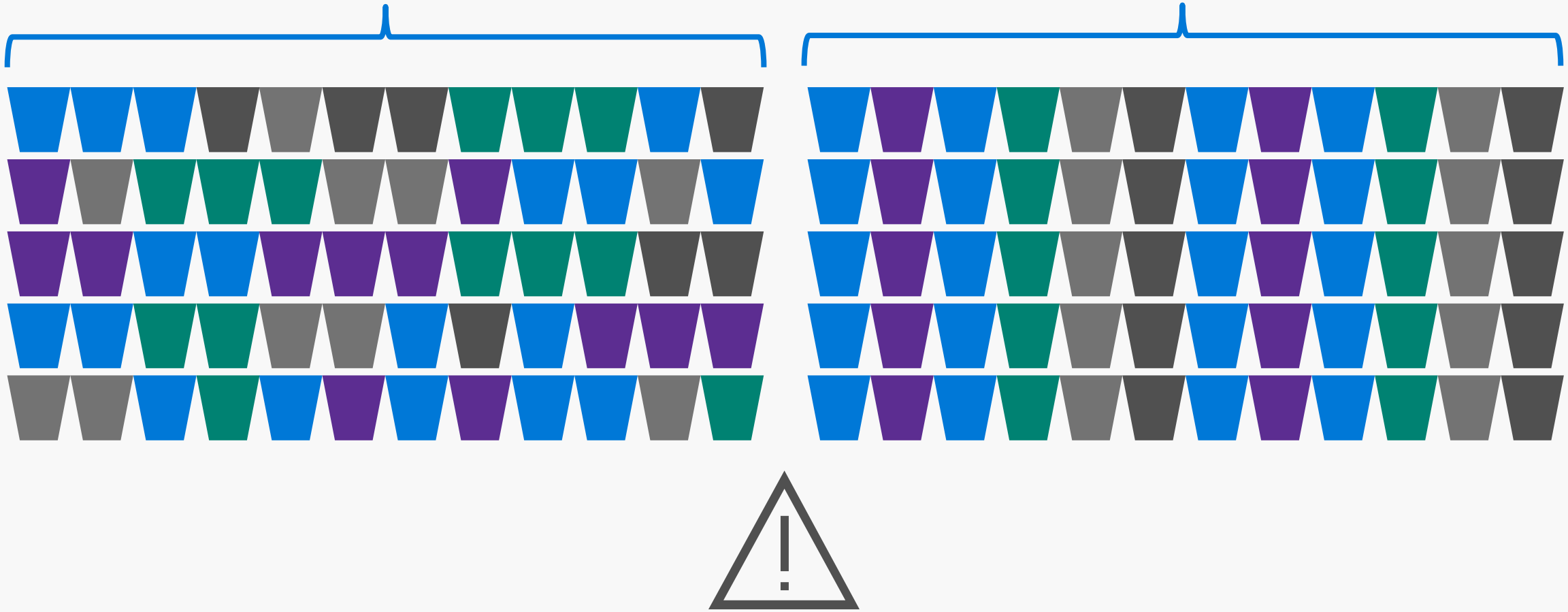
Store_Sales HASH([ProductKey])



Joining distributed tables: round robin to hash

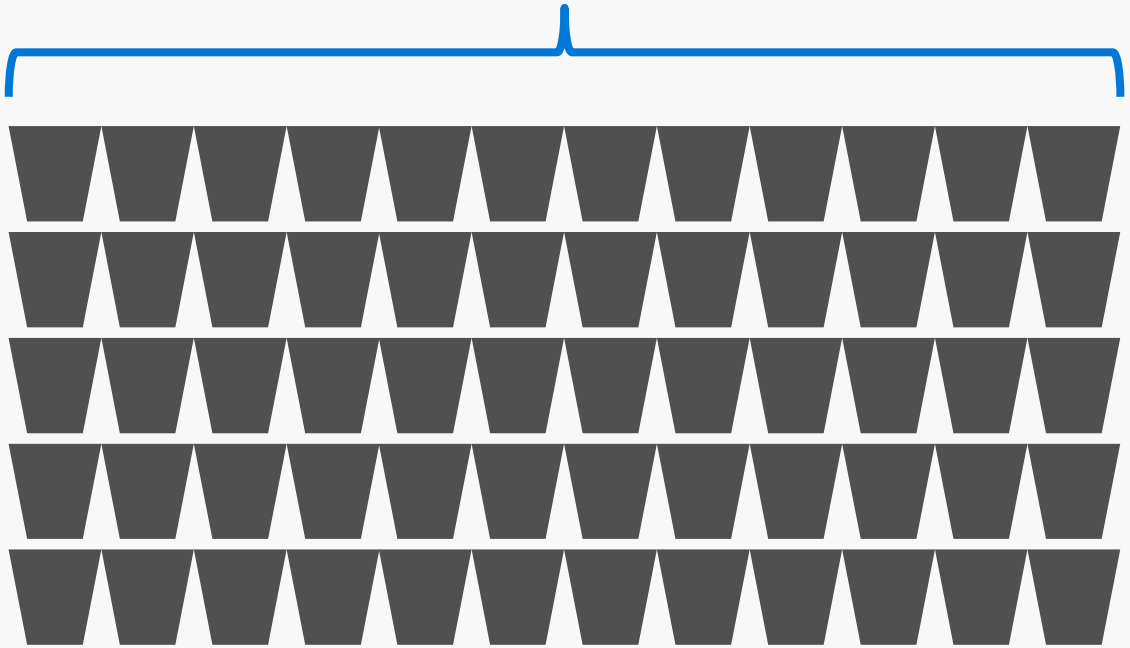
DimProduct ROUND_ROBIN
[ProductKey] **INT** NULL

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

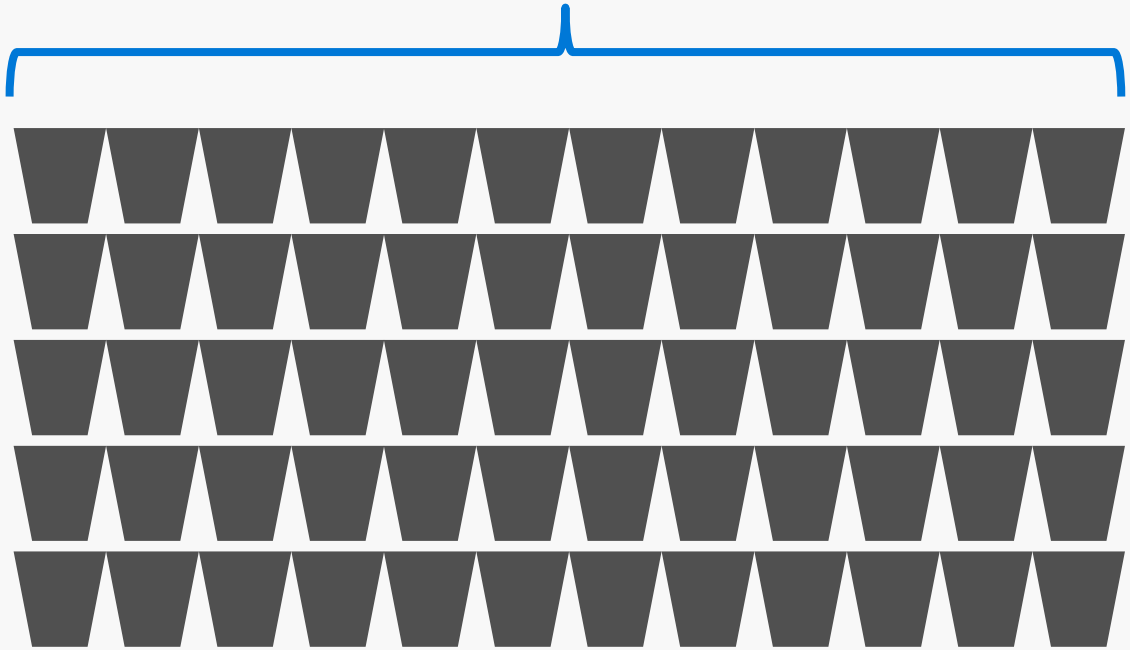


Joining distributed tables: hash to hash

Store_Sales HASH([ProductKey])



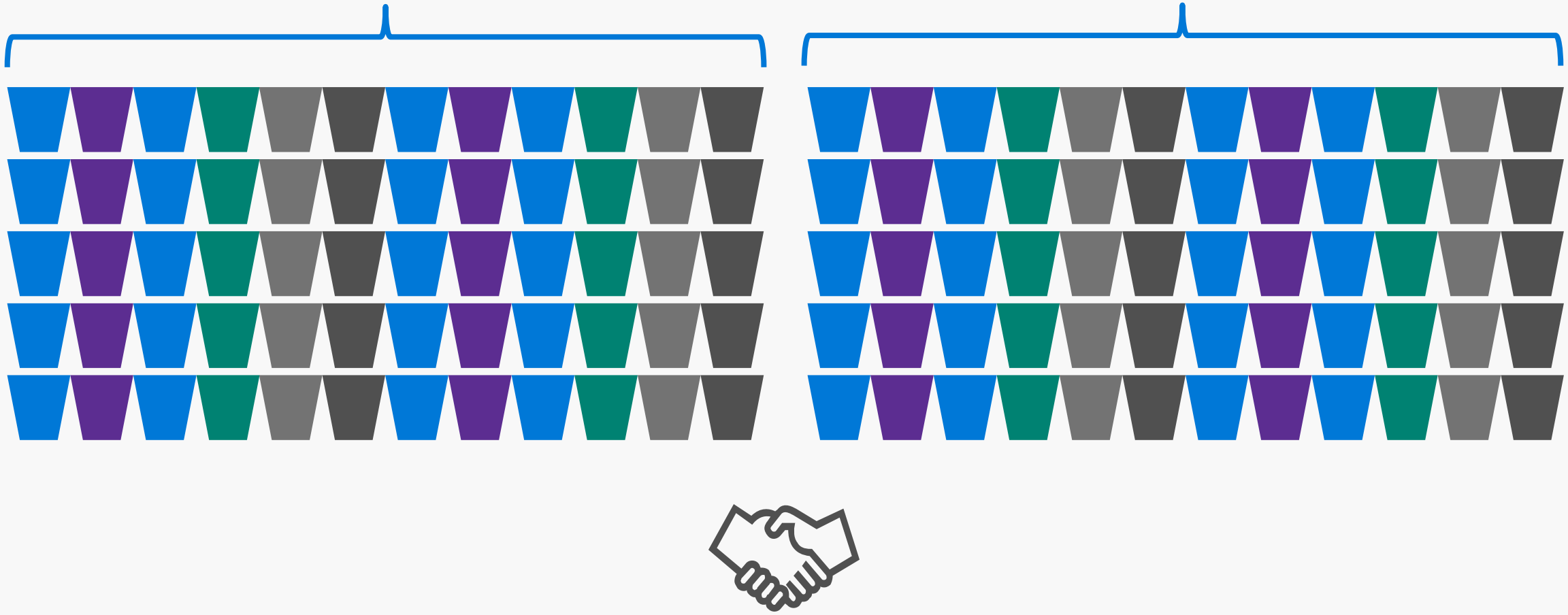
Web_Sales HASH([ProductKey])



Joining distributed tables: hash to hash

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

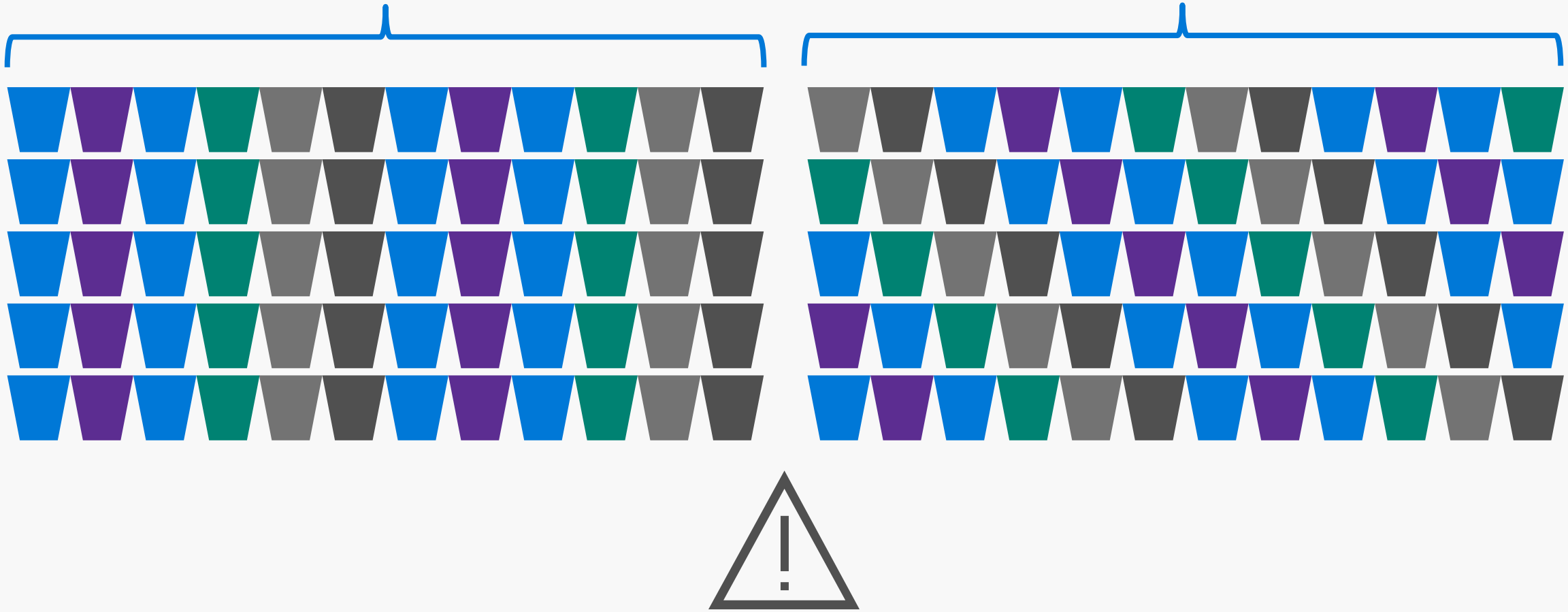
Web_Sales HASH([ProductKey])
[ProductKey] **INT** NULL



Joining distributed tables: incompatible hash

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

Web_Sales HASH([ProductKey])
[ProductKey] **BIGINT** NULL



Guidance: hash and joins

Hash distributed columns

Identify columns used frequently in joins and group by aggregations
Avoid columns used in the where clause

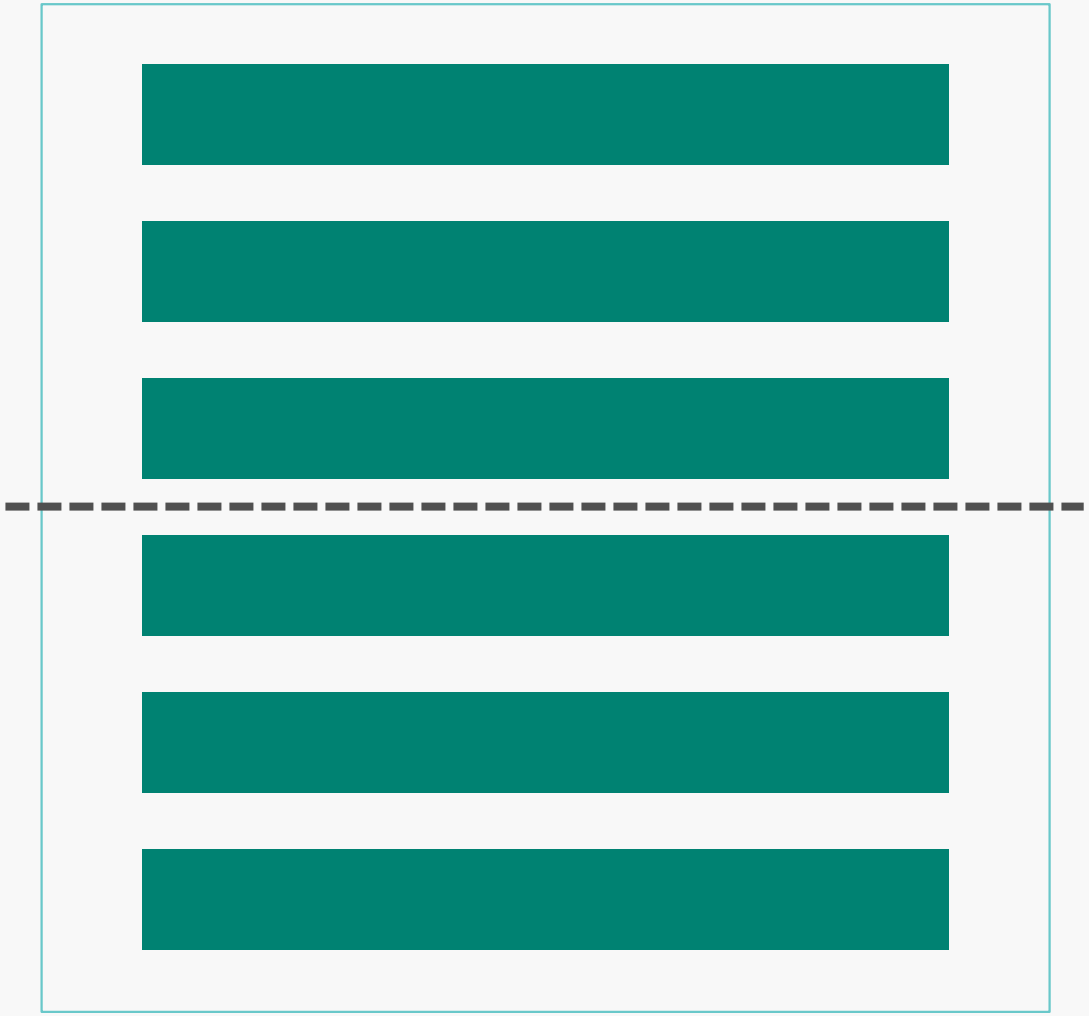
Join compatibility

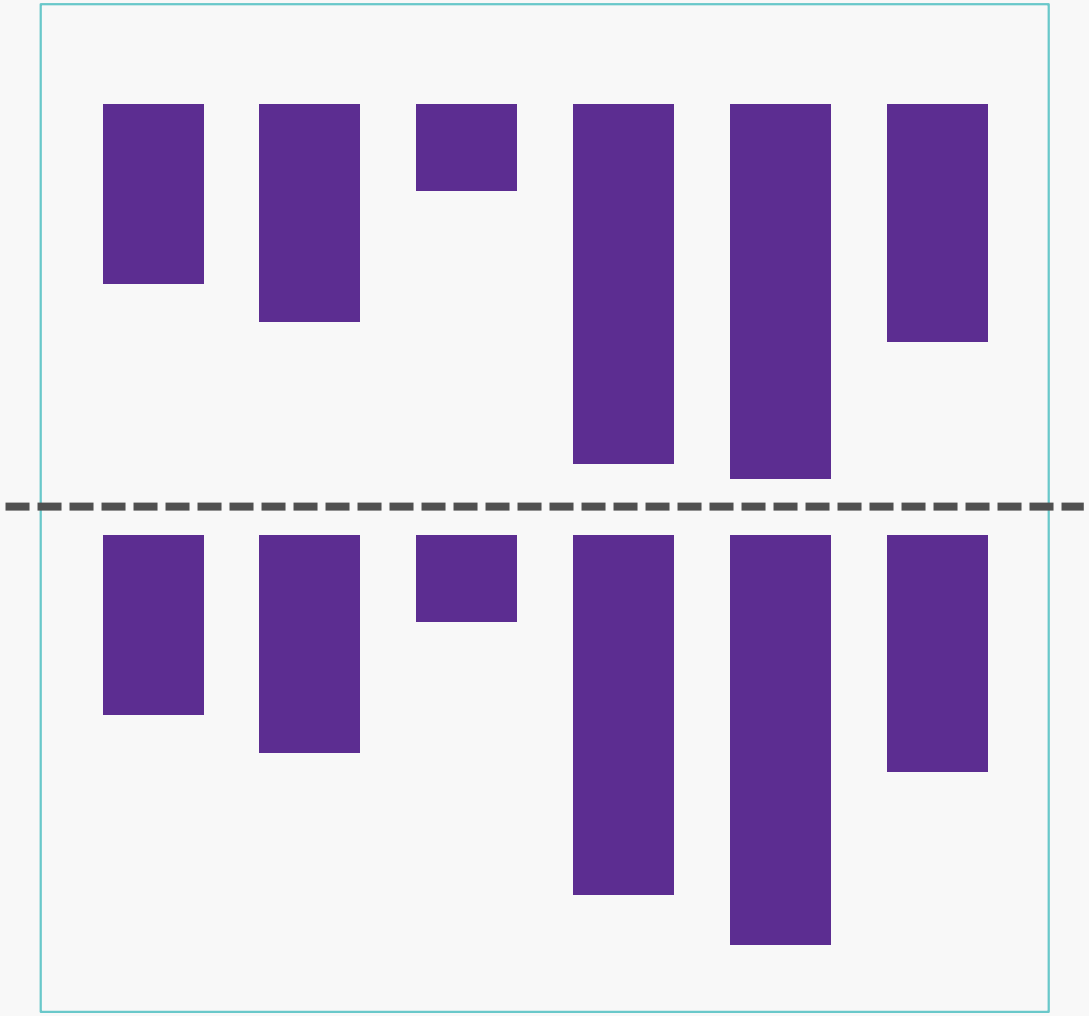
Shared distribution key
Must be an equi-join (=)
Distribution keys must have the same data type

Aggregation compatibility

Compatible = distribution key in group by clause
Incompatible = no group by clause or distribution key not present

Row & Column Store & Partition





Guidance: Partitioning

Don’t over partition

Partitioning granularity likely to differ to SQL Server
Data is already spread across 60 distributions
Columnstore index row groups up to 1,048,576 rows

Partition for data management

Sliding window development
Targeted index rebuilds

Pricing

Compute pricing: DWU

$900

Per DWU100

Per Month

Storage sizing: Premium storage

$122.88

TiB / Month

Data files

Log files

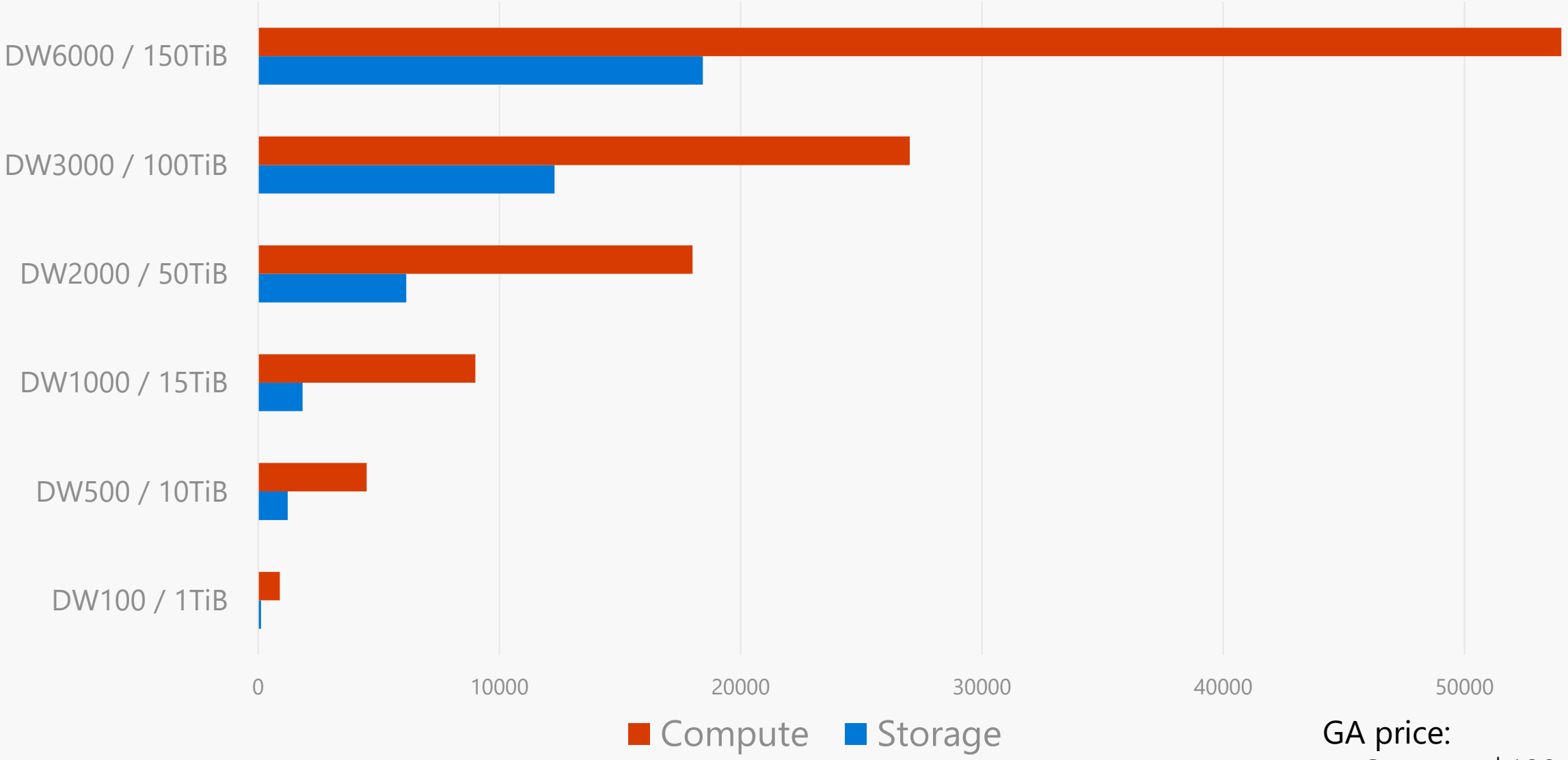
Snapshots

Storage pricing: Geo-redundant backups

$0.12

GB / Month

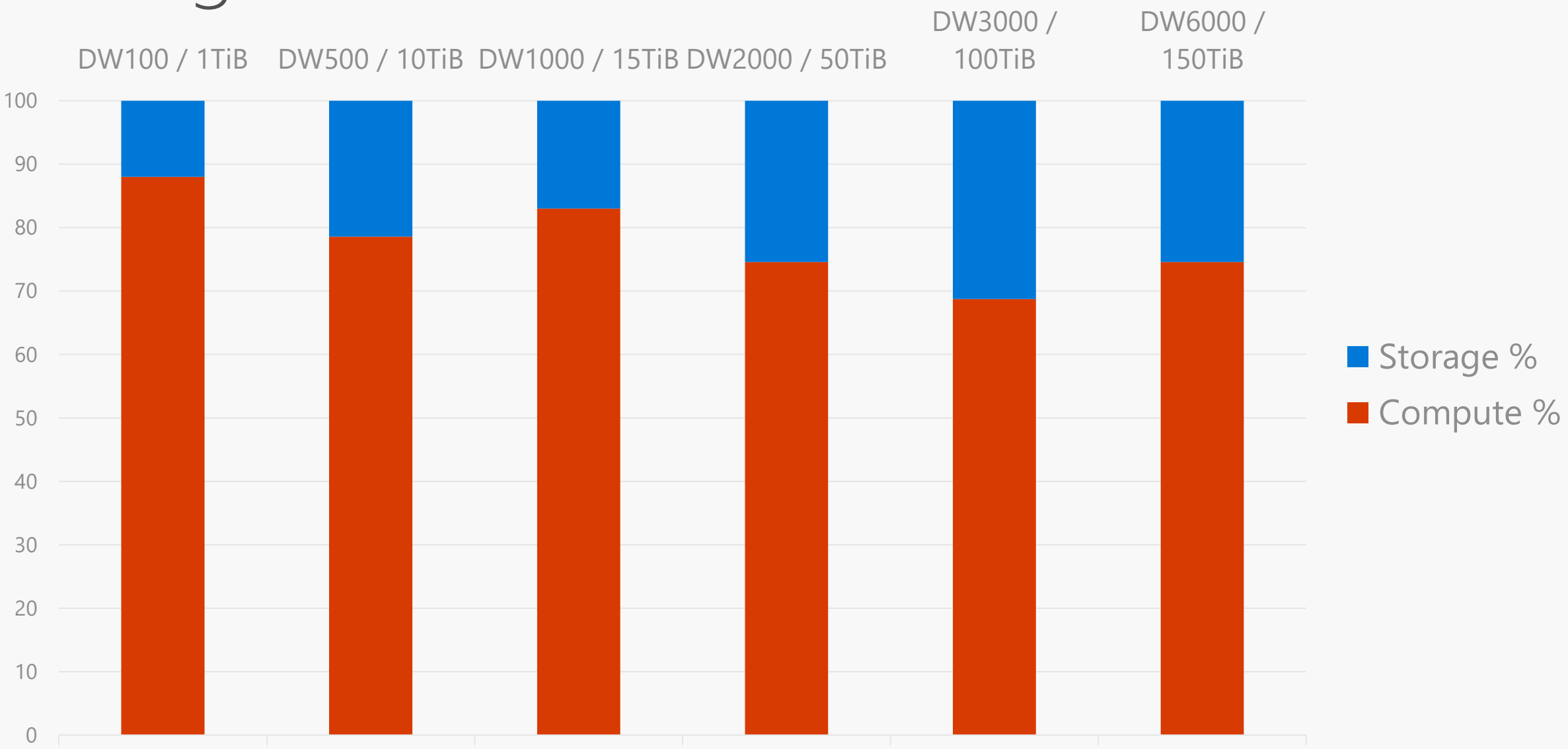
Pricing scenarios



GA price:

- Storage: $122.88
- DWU: $1.21

Pricing scenarios



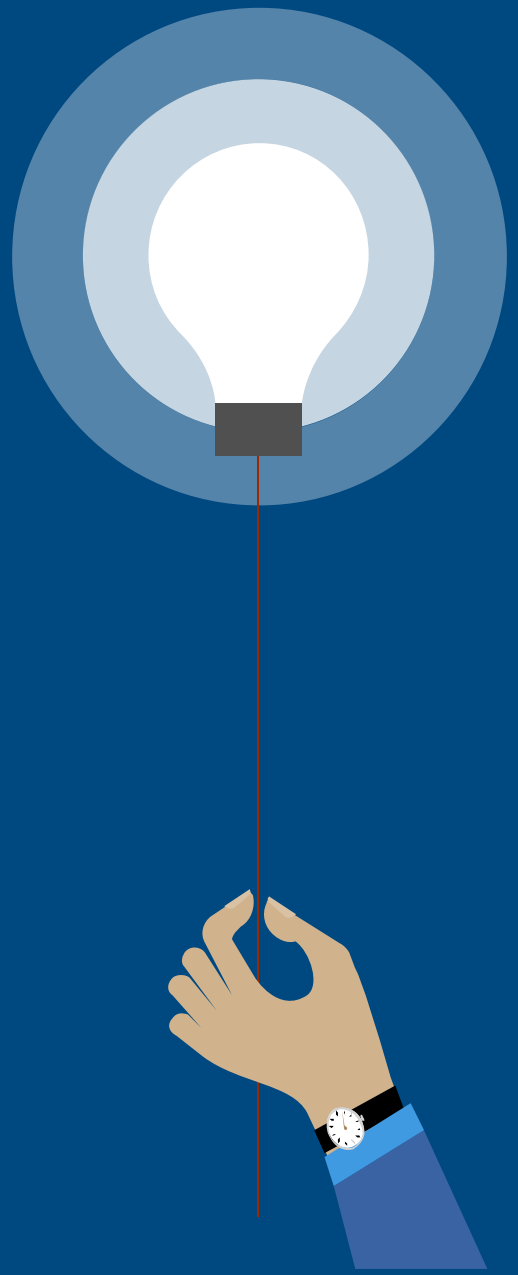
Resources \ CTA

Free trial

- One month free trial
- Up to 1000 DWU
- No restriction on usage during trial

Trial ends June 30th 2017

http://azure.com/sqldwfreetrial



Online (external\public) resources

ACOM

- http://aka.ms/sql-dw-docs

Feedback

- http://aka.ms/sql-dw-feedback

MSDN

- http://aka.ms/msdn-t-sql

Stack overflow

- http://stackoverflow.com/questions/tagged/azure-sqldw

Video: A developers guide to Azure SQL Data Warehouse

- http://aka.ms/build-sql-dw-developers-guide

Video: Building Analytics for the Modern Business

- http://aka.ms/build-sql-dw-analytics-modern-biz

Video: Fraud analysis using SQL DW & SQL Server 2016 Reporting Services

- http://aka.ms/video-sql-dw-fraud-analysis

Demos, labs and training

Ramp up

Unlocking Epilepsy with Azure SQL Data Warehouse

- https://microsoft.sharepoint.com/sites/infopedia/pages/layouts/kcdoc.aspx?k=G01KC-1-17003

Fraud detection with Azure SQL Data Warehouse & SQL Server 2016 Reporting Services

- https://microsoft.sharepoint.com/sites/infopedia/pages/layouts/kcdoc.aspx?k=G01KC-1-17031

Data science and data warehousing with Azure SQL Data Warehouse and Cortana Intelligence Suite

- https://gallery.cortanaintelligence.com/Solution/Data-Warehousing-and-Data-Science-with-SQL-Data-Warehouse-and-Spark-2

MPP Data Warehouse training site

- https://microsoft.sharepoint.com/sites/Infopedia_G01KC/Pages/Custom/MPP-Data-Warehouse-Training.aspx