Microsoft

Key Concepts

Massively Parallel Processing with the Microsoft Analytics Platform System

Agenda

- APS in a nutshell
- Appliance vs. Reference Architecture
- Scale up vs. Scale out
- Elements of MPP Database Systems
- Skew
- Parallelism
- Agnostic in the Enterprise

APS in a Nutshell

The Engine for Warehousing & Analytics

- Built for scale out warehousing and analytics
- Deep, native integration with Hadoop
- High performance data loading platform
- Concurrent data warehouse workload pattern
  - Load
  - Query

Much more than simply SQL Server

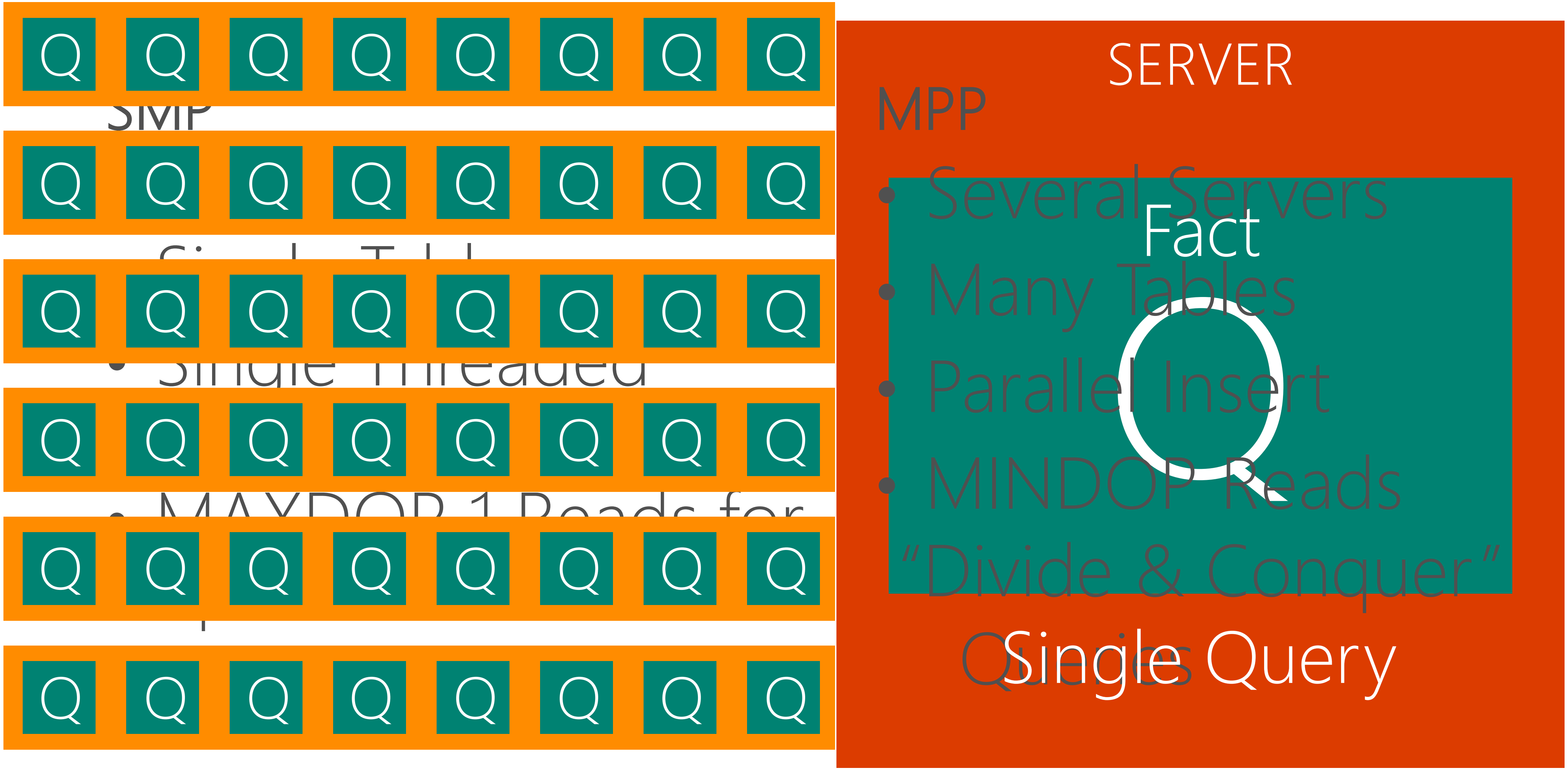
SQL Server plus

- Hardware
- Additional PDW specific software
- Hadoop integration
- Built-in development of engineering Best Practices
- Integrated systems management

New features come <u>first</u> to APS

- Updateable column store
- Agnostic Hadoop integration via PolyBase
- Cardinality estimation
- Cost-based distributed SQL Query (DSQL) engine
- Hub and spoke architecture support
- Analytical functions (e.g. LAG and LEAD)
- Incremental *functional* releases each year

It's parallel all the time, every time



Appliances

What is an appliance?

Think about buying a kettle…

- Performs a specific function – boils water
- Tuned for that function – doesn’t heat soup

Appliance buying process

Research your kettle

- Performance
- Aesthetics
- Compatibility

Placing an order

- Wait for delivery

Post-delivery

- Unbox
- Plug it in
- Activate warranty

Use it!

Remember:
only boil water, don't heat soup!

What happens if…

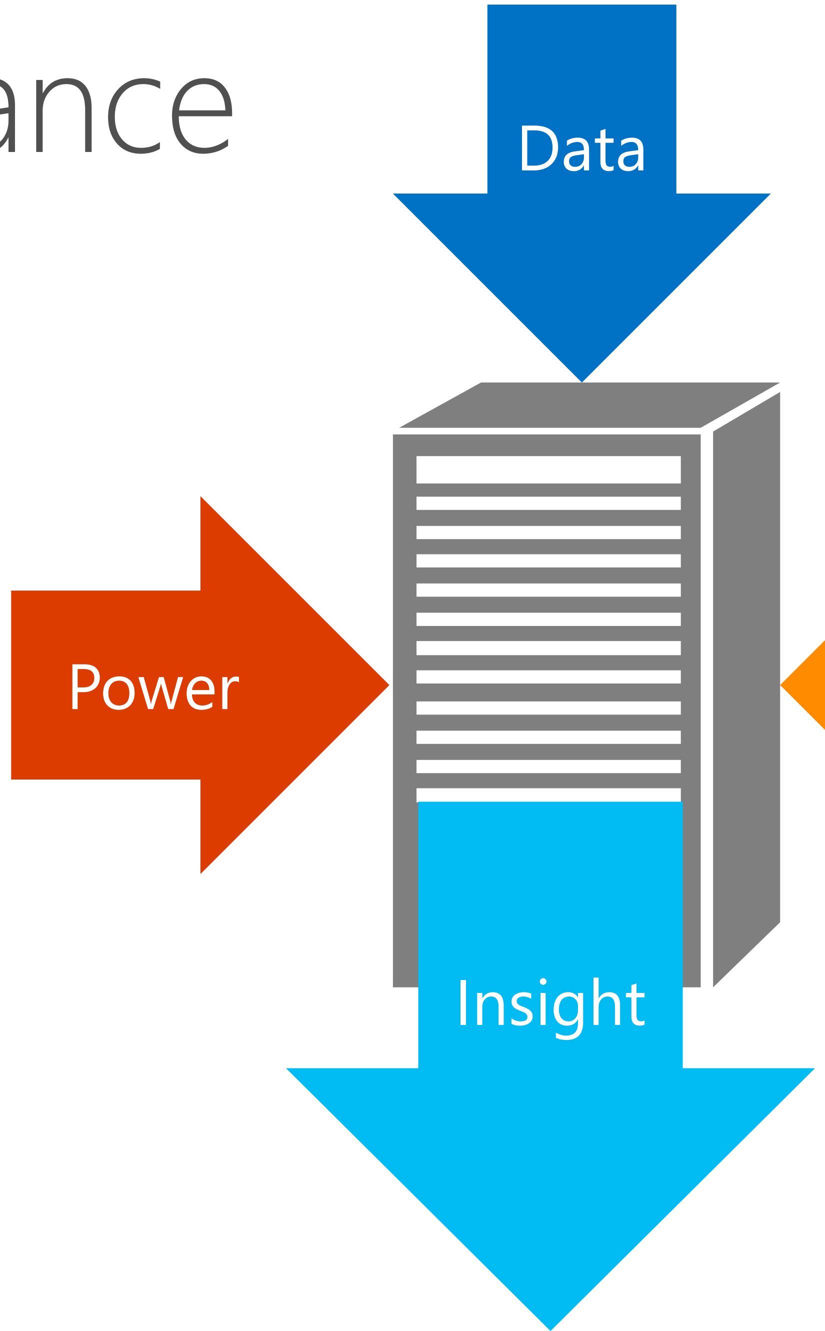
I want to boil water faster

- Do: buy a better kettle
- Don’t: reach for the screwdriver…

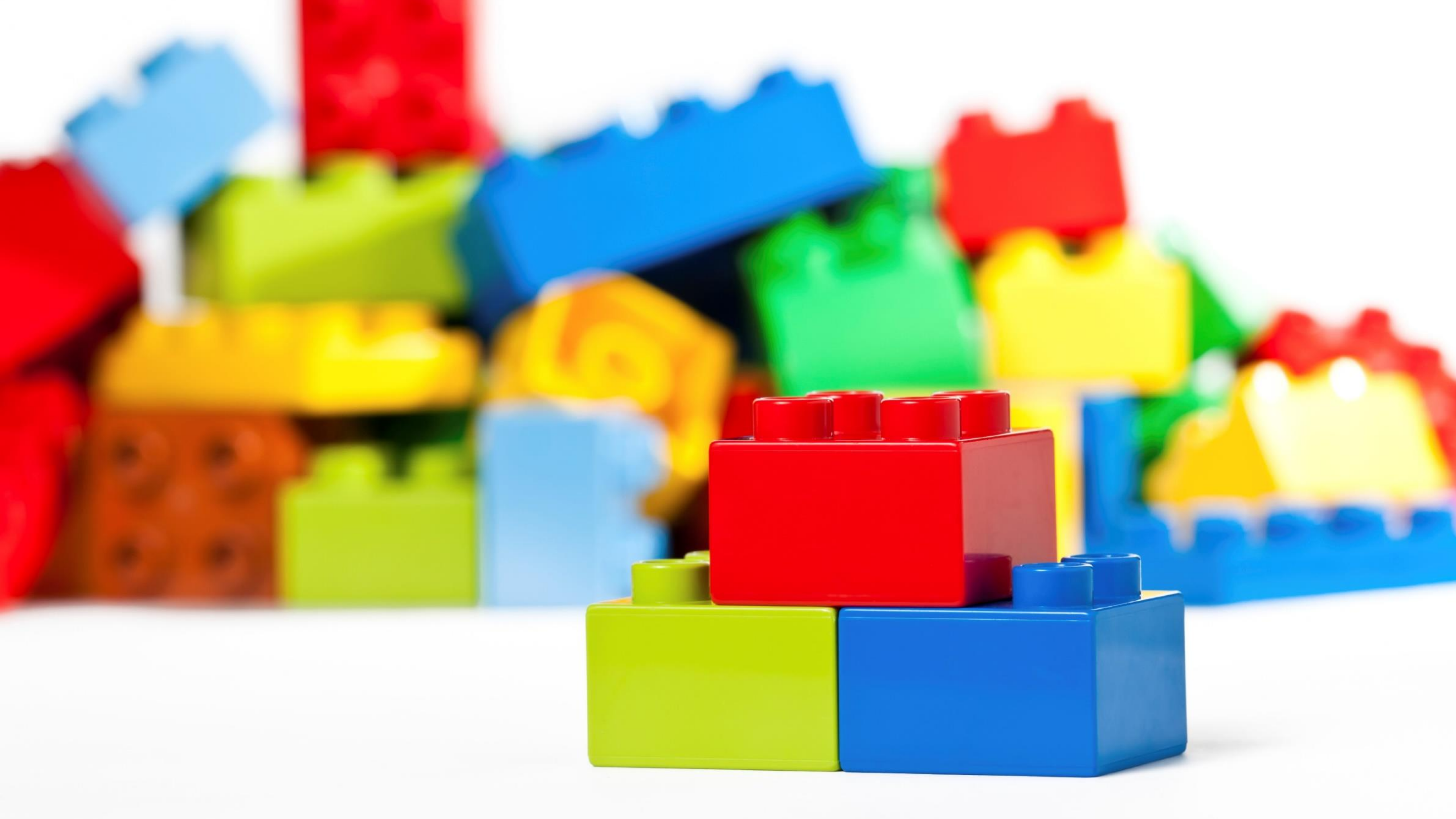
Kettle breaks

- Activate warranty

APS - Appliance



Reference Architectures



What is a Reference Architecture?

Think about building with Lego

- You get the bricks
- You get the instructions
- You do the build
- If you follow the instructions you get the product you purchased
- If you don't…you get what you built

Symmetric Multi-Processing

SMP - Definition

- Two or more multi-processors (CPUs or cores)
- Connected to a single shared memory
- Full access to all I/O devices
- Shared main/motherboard
- Known as “Scaling Up”

SMP - Examples

Microsoft® SQL Server® 2012
Fast Track Data Warehouse

Shared resources

- Good for joins
- Bad for bottlenecks

Massively Parallel Processing

What is MPP?

- A divide and conquer strategy
- Take one big problem & break it up
- Team approach “Many hands make light work”

Requires

- A method for scheduling tasks
- A communication plan to maximise efficiency
- A distribution method for exchange of goods

SCALE UP VS OUT

UP

- Diminishing returns
- Non-linear costs at scale
- Parallel execution hard
- Low-mid complexity
- High concurrency
- Shared everything

OUT

- Linear scale (6PB+)
- Incremental cost
- Parallel execution by default
- Complex queries
- Medium concurrency
- Shared nothing


RST
NMI
426609
ID

Elements of MPP Database

Concepts

- Logical layer
- Physical layer
- Distributed query engine
- Orchestration
- Data movement
- Lots of machines!

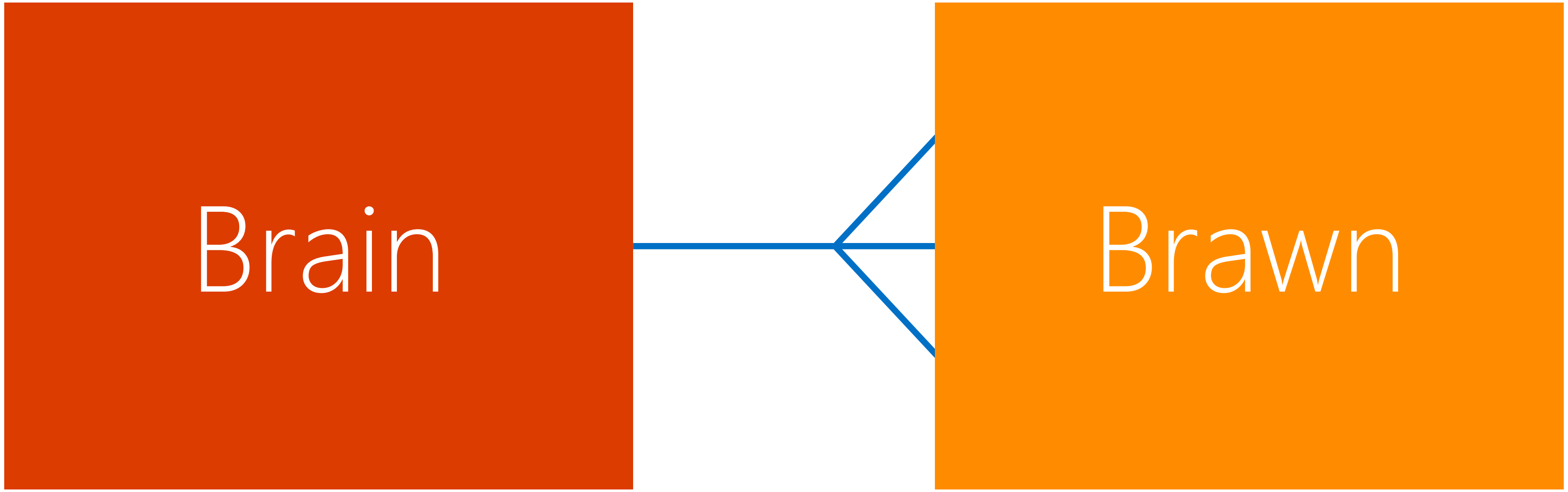
Logical Layer

- Holds application metadata
- Does not persist application data
- Receives intermediate results
- Performs final aggregation

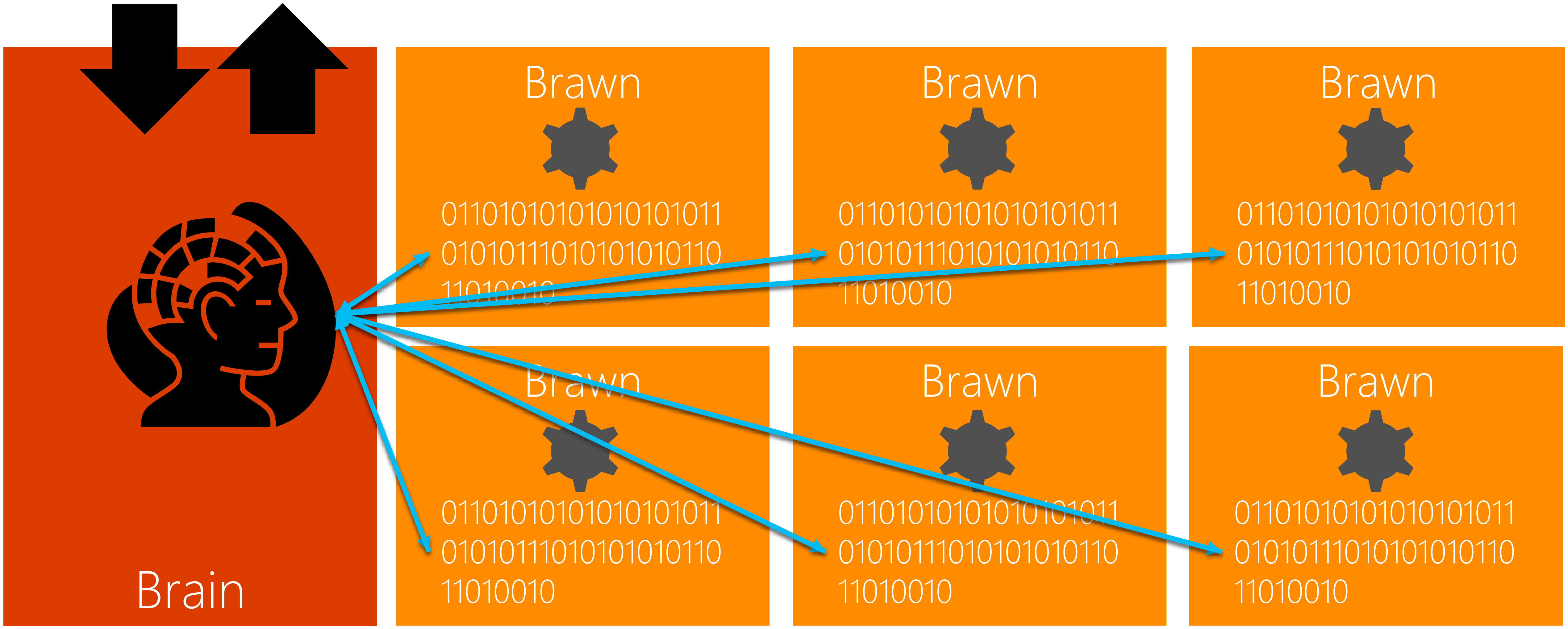
Physical Layer

- Persists application data
- Performs queries as instructed

Logical vs Physical



Brain vs. Brawn



Brain

- Accepts requests from user
- Interprets requests for scale out
- Optimises requests
- Orchestrates actions / steps
- Final computation
- Returns result

> - Runs SQL Server
> - Stores metadata
> - Performs final computation

One brain

Requires distributed query engine

Brawn

Performs the heavy lift

- Accepts requests from brain

Is a highly tuned SMP System

- Optimises requests from brain

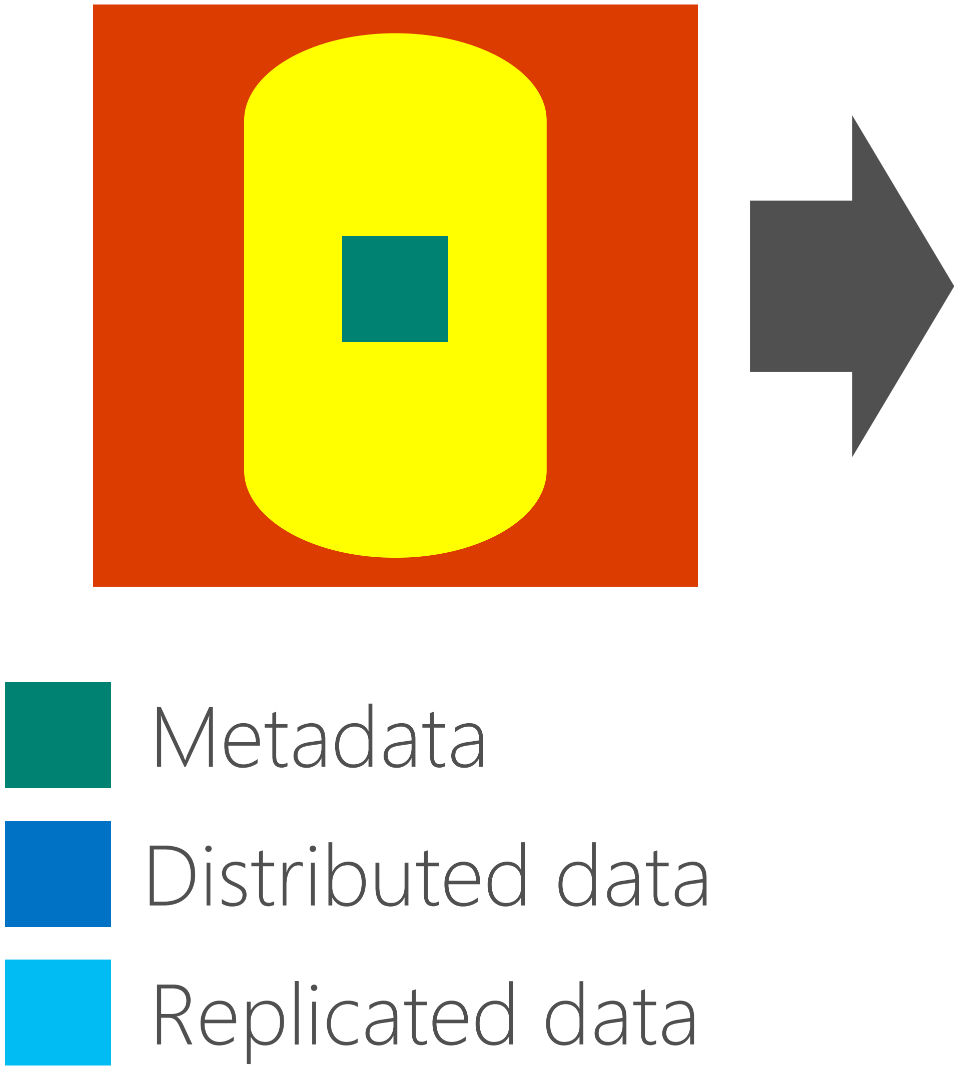
Executes query against the data

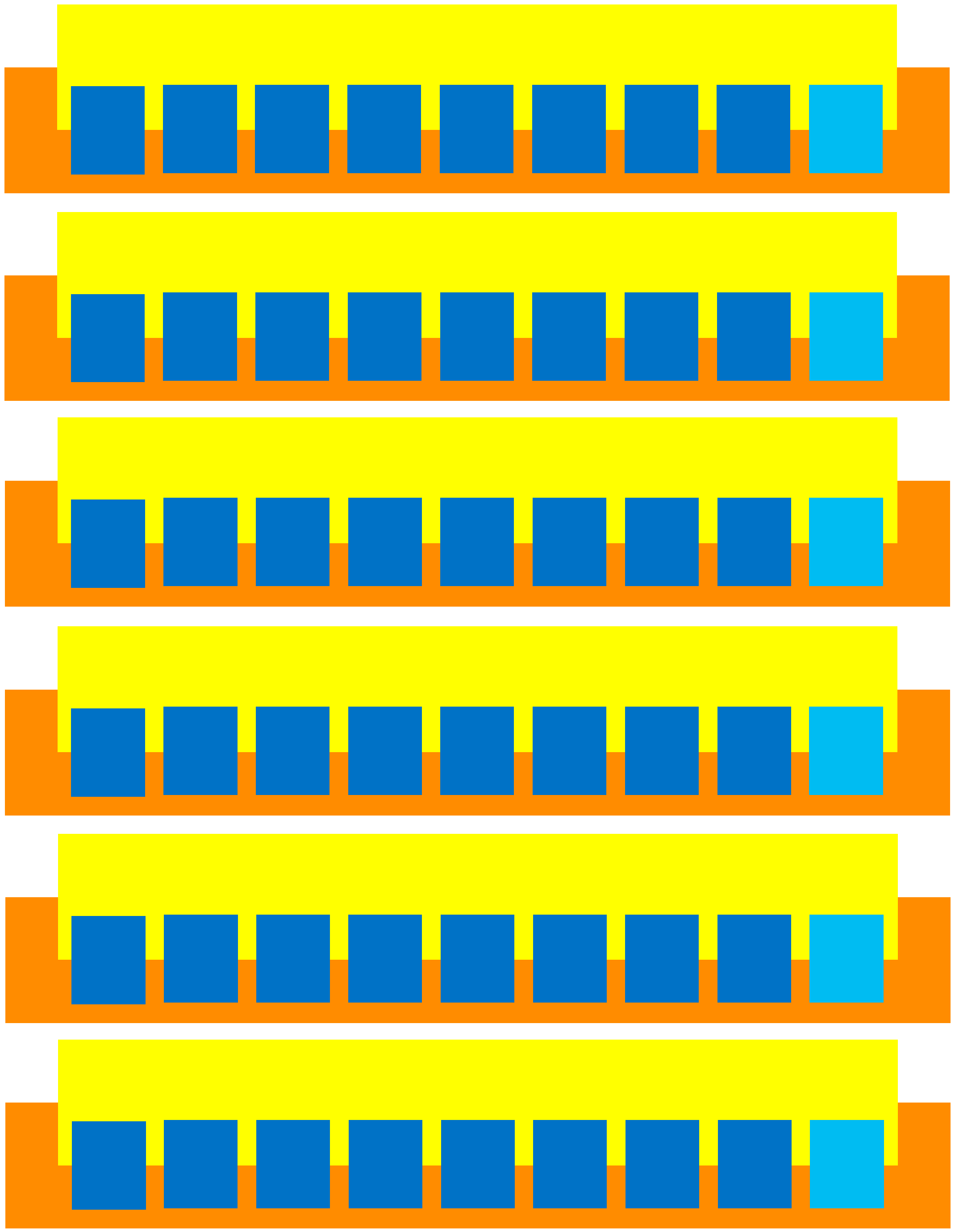
Lots of brawn

No direct user interaction

Uses SQL Server

Logical vs Physical





Replicating Data

Data Replication Concepts

Defined:

"The process of repeating, duplicating or reproducing..."

Why do we replicate data?

Primarily one of two reasons

- Enhanced availability
- Increased (read) performance

What is the cost of replicating data?

Slower Writes!

Instead of writing data once you write it **n** times

- PDW region: **n** = number of compute nodes in your appliance
- Hadoop: **n** = replication factor set in Hadoop (default is 3 times)
- HDInsight: **n** = default is 2 or 3 depending on size of Hadoop region

Data Replication in PDW

- Table Level
- Whole table copied to every Compute node
- Writes synchronised with distributed transactions
- Read optimisation
- Not used for availability
- Facilitates joins to distributed data
- Tends to be smaller, read-heavy tables such as dimensions

Data Replication in Hadoop

- Affects all tables
- Availability & performance

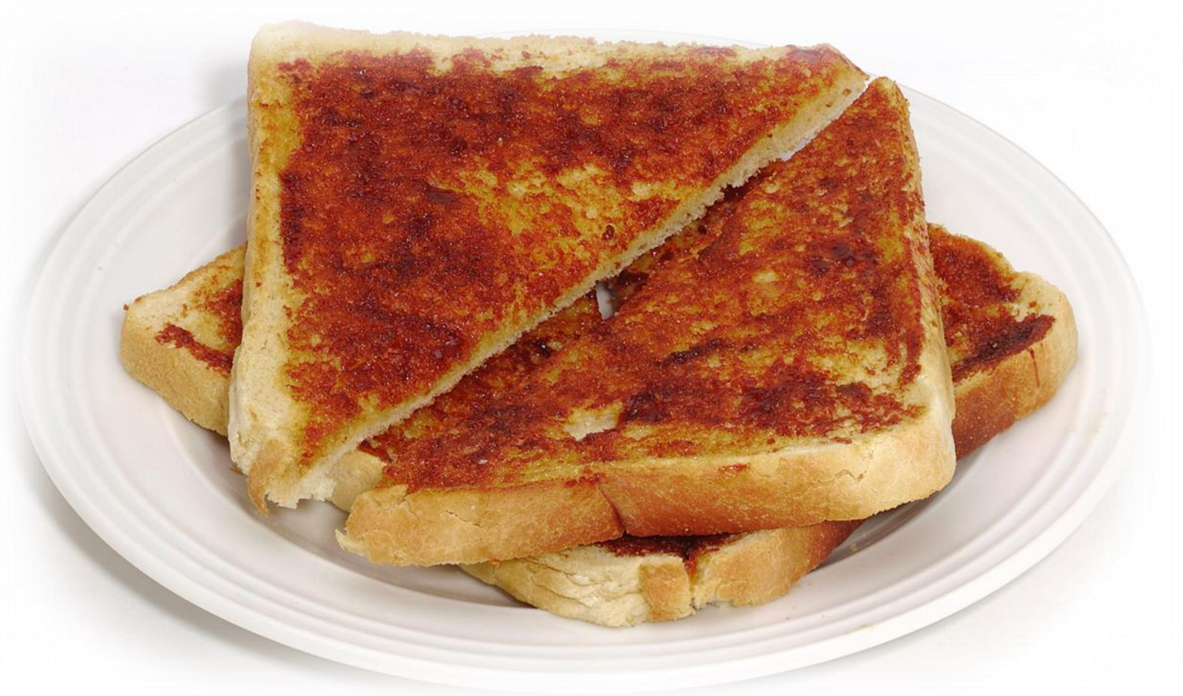
Default replication factor is 3

- 1 data node with the data
- 1 data node locally (same switch)
- 1 data node remotely (different switch)

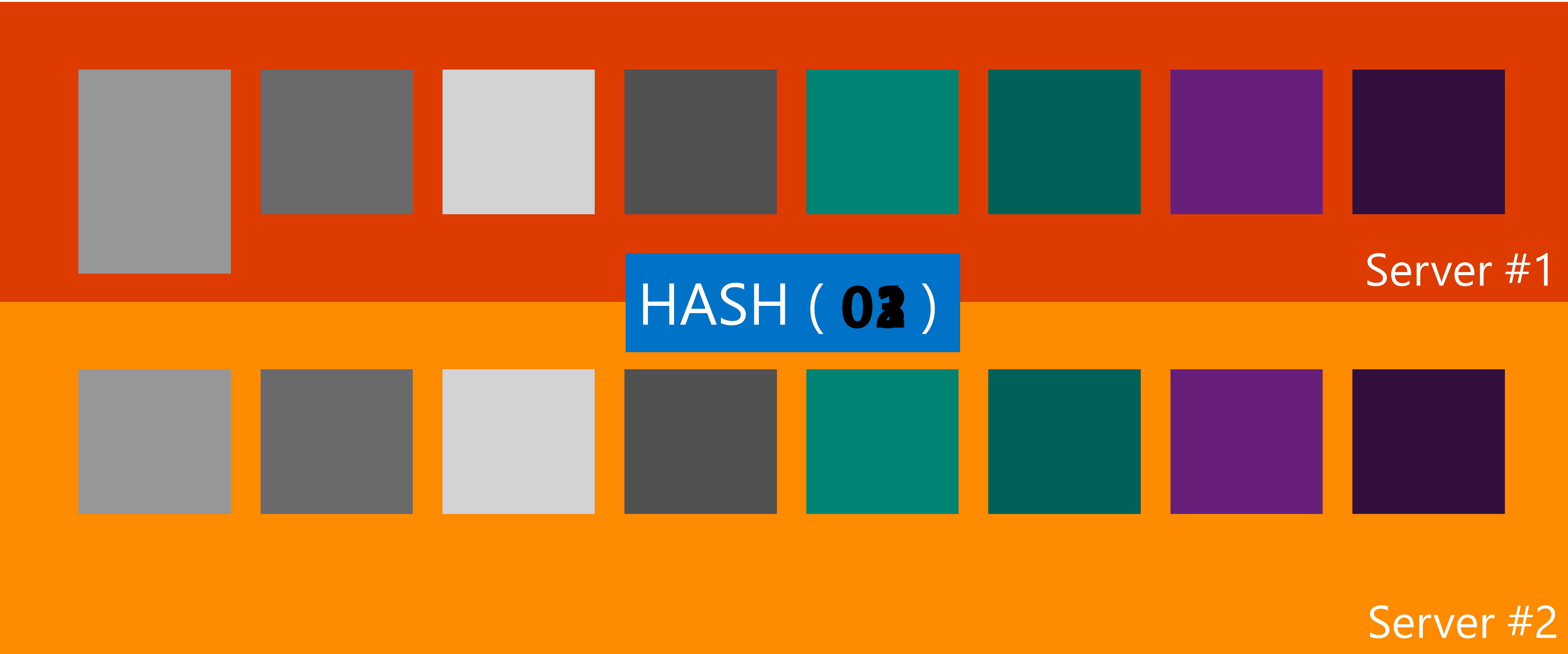
Distributing Data

Why Distribute Data

- Divide and conquer – lots of small queries to solve
- Evenly spreading the data leads to even use of the appliance resources



How Data Gets Distributed



What Data do we Distribute?

PDW

- Tends to be larger fact tables
- Can also be write-heavy tables (log tables)

HDInsight

- All data is distributed across the HDFS file system
- All data is also replicated

Data Distribution

- Table level
- Data spread across all PDW Compute nodes
- Hash function calculates each row location
- DMS moves each row to correct location
- Data is held and written once, hardware provides availability

Understanding Distributions

- Each distributed data location = a bucket of data
- Each bucket is called a distribution
- Each distribution:
  - maps to a physical table
  - allocated physical space
  - contains all rows that have the same distribution value
- 8 distributions on every compute node (A-H)
- Hash based on byte “padded” not actual value

PDW Distributed Table Theory

Recap

PDW holds data in one of two ways:

- Distributed
- Replicated

When distributed

- Data is distributed across the entire cluster
- A hashing function is used to spread the data
- Hash created during either Insert or Load
- Hash performed by DMS not SQL Server
- Hash is based on a single column in a table
- Column chosen for hash is not updateable

When Replicated

- DMS copies data to each compute node
- No hashing is required

Distribution Basics

Distributed tables are split in two directions

- Vertically across compute nodes
- Horizontally within the compute node
- Each Horizontal split is called a distribution
- # Distributions = # compute nodes * 8

Distribution Matrix

	A	B	C	D	E	F	G	H	REPL	LOG	PRI
Compute 01											
Compute 02											
Compute 03											
Compute 04											
Compute 05											
Compute 06											

10 as INT

	A	B	C	D	E	F	G	H	REPL	LOG	PRI
Compute 01											
Compute 02											
Compute 03					10						
Compute 04											
Compute 05											
Compute 06											

10 as BIGINT

	A	B	C	D	E	F	G	H	REPL	LOG	PRI
Compute 01											
Compute 02											
Compute 03					10						
Compute 04											
Compute 05											
Compute 06								10			

72 as INT

	A	B	C	D	E	F	G	H	REPL	LOG	PRI
Compute 01											
Compute 02											
Compute 03					10 72						
Compute 04											
Compute 05											
Compute 06											

What about NULL?

	A	B	C	D	E	F	G	H	REPL	LOG	PRI
Compute 01	NULL										
Compute 02											
Compute 03					10						
Compute 04											
Compute 05											
Compute 06								10			

Distribution Rule Summary

1. Hash on padded data type
2. Hash is not based on actual value
3. Hash is performed on a single column
4. All rows with the same padded value end up in the same distribution
5. "NULL" will always go to Distribution 1A
6. A distribution hold rows for > 1 hashed value

Consequences of Distribution Theory

The hash is consistent across all tables

- Great for joining tables together
- Must have the same data type

Rationalise your data types!

Consequences of Distribution Theory

Distribution key is not updateable!

Pick a distribution column that:

- Is static
- Does not need inferred members (-1)
- Does not contain NULL

Changing distribution key = Re-create the table!

Locality & Orchestration

Understanding Locality

- MPP systems distribute the data across servers
- Good for scalability
- Introduces overheads
- Data is not all in the same place!
- First we may need to move the data then process it

Local vs Global Operations

Local operation

- Occurs on one or more Compute nodes
- Operates on a subset of the data

Global operation

- Occurs on the Control node
- Operates on the set of data

Location, Location, Location

Data is considered to be local (co-located) when

- No data movement is required to resolve a query
- All data is present on a given node to process the query

Example – Sales Table

- Distributed by item
- Sales for an item are held in one distribution
- Sales for a store are in all distributions
- Joining by store *may* require movement
- Grouping by store *may* require movement

Sales Table – Distributed by Item

Sales Item 1 Cust 1..N Store 1..N	Sales Item 2 Cust 1..N Store 1..N	Sales Item 3 Cust 1..N Store 1..N	Sales Item 4 Cust 1..N Store 1..N	Sales Item 5 Cust 1..N Store 1..N	Sales Item 6 Cust 1..N Store 1..N	Sales Item 7 Cust 1..N Store 1..N	Sales Item 8 Cust 1..N Store 1..N

Server #1

Sales Item 9 Cust 1..N Store 1..N	Sales Item 10 Cust 1..N Store 1..N	Sales Item 11 Cust 1..N Store 1..N	Sales Item 12 Cust 1..N Store 1..N	Sales Item 13 Cust 1..N Store 1..N	Sales Item 14 Cust 1..N Store 1..N	Sales Item 15 Cust 1..N Store 1..N	Sales Item 16 Cust 1..N Store 1..N

Server #2

Sales Table – Distributed by Store

Sales Store 1 Cust 1..N Item 1..N	Sales Store 2 Cust 1..N Item 1..N	Sales Store 3 Cust 1..N Item 1..N	Sales Store 4 Cust 1..N Item 1..N	Sales Store 5 Cust 1..N Item 1..N	Sales Store 6 Cust 1..N Item 1..N	Sales Store 7 Cust 1..N Item 1..N	Sales Store 8 Cust 1..N Item 1..N

Server #1

Sales Store 9 Cust 1..N Item 1..N	Sales Store 10 Cust 1..N Item 1..N	Sales Store 11 Cust 1..N Item 1..N	Sales Store 12 Cust 1..N Item 1..N	Sales Store 13 Cust 1..N Item 1..N	Sales Store 14 Cust 1..N Item 1..N	Sales Store 15 Cust 1..N Item 1..N	Sales Store 16 Cust 1..N Item 1..N

Server #2

Data Location

When data is distributed PDW:

- Knows which column the data is distributed on
- Does not know which distribution contains the value x
- Queries all distributions

Orchestration

- PDW take in a request from a user
- Breaks the request down into series of steps
- Steps can have parallel operations
- Serially iterates over the steps
- Manages return of answer back to user

Orchestration steps happen in series
Operations within step occur in parallel

Orchestration Example

1. Generate random name for temp table
2. Generate temp table (using random name)
3. Move data into temp table
4. Execute query using temp table returning data to user
5. Perform clean-up

Skew

Distributed Data = Buckets of Water

Know Your Enemy!

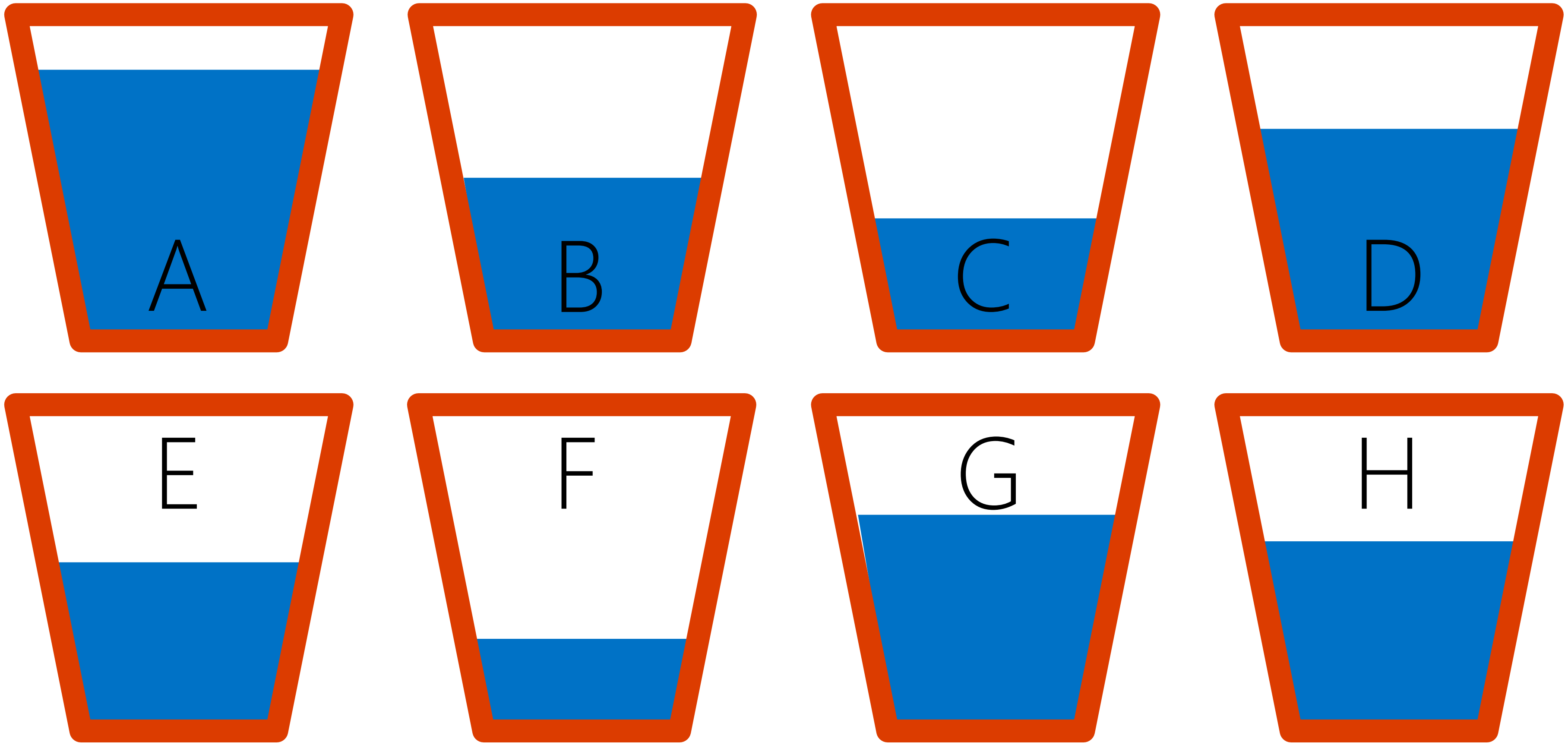
If MPP is the analytics superhero…

…then "Data Skew" is the evil arch enemy

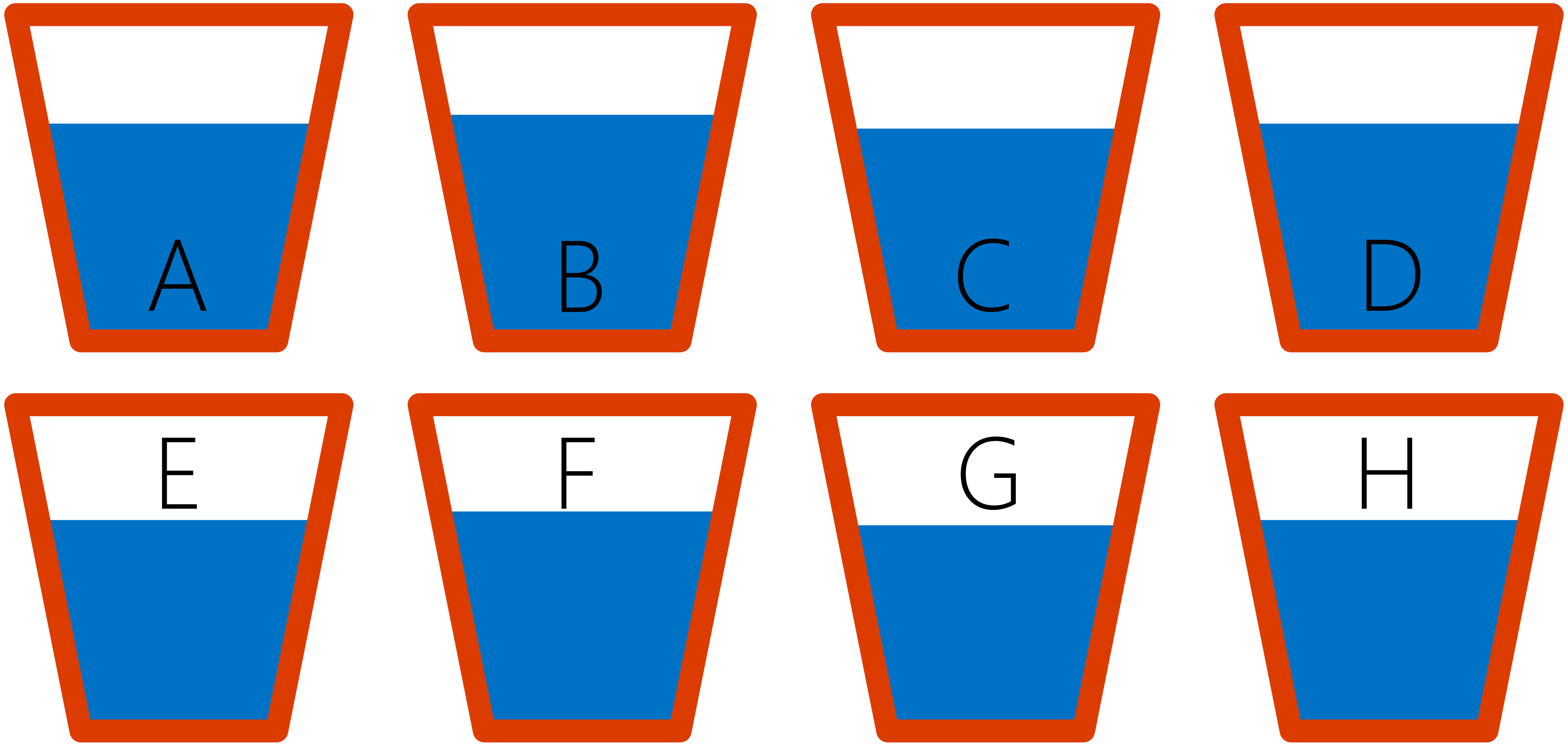
Performance

Storage

If this was a drinking race who wins?



What about now?

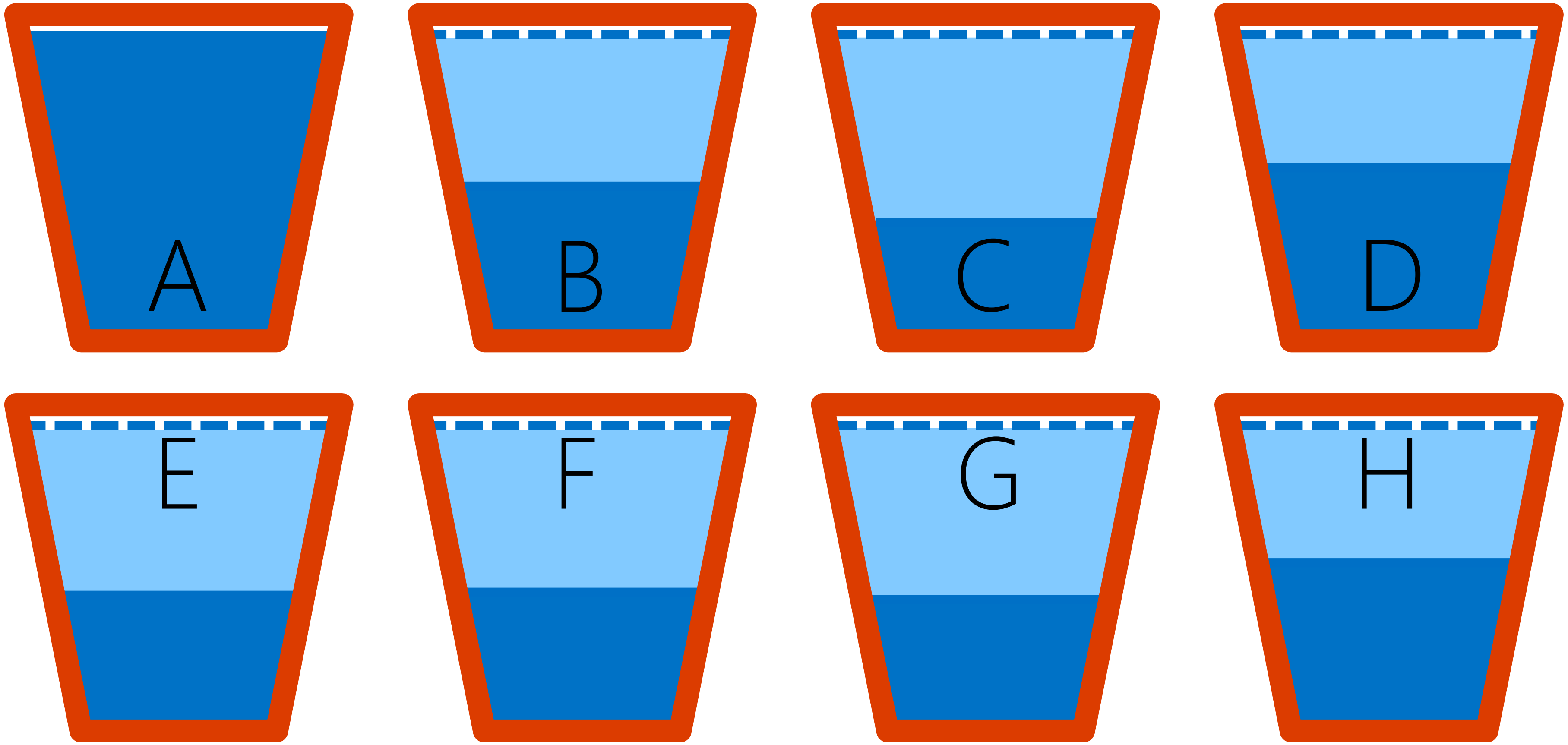


Performance Impact of Skew

When the data is skewed…

- Some queries finish very quickly
- Others take disproportionately longer
- The user query is only complete when *all* queries have finished

Storage & Skew



Impact of Skew on Storage

- When one bucket is full all buckets are full
- Skewed data leads to accelerated consumption of storage capacity
- Available storage therefore is a logical concept
- Calculated as

    MAX(storage used by bucket) * # of buckets

Parallelism

How do we get parallelism?

- Via distributed tables
- Each distribution can be queried in parallel

Partial Parallelism

- Performed in parallel across compute nodes
- Performed in series across the distributions
- Guarantees transactional behaviour
- Examples:
  - INSERT
  - UPDATE
  - DELETE

Partial parallelism in action

Sales A	Sales B	Sales C	Sales D	Sales E	Sales F	Sales G	Sales H

Server #1

Sales A	Sales B	Sales C	Sales D	Sales E	Sales F	Sales G	Sales H

Server #2

Full Parallelism

- Performed in parallel across compute nodes
- Performed in parallel across the distributions
- Used for
  - New object creation
  - Reading data from a distributed table
- Examples:
  - CTAS (Create Table As Select)
  - SELECT

Full Parallelism in action

MINDOP

- Witnessed when querying distributed tables
- Each distribution is queried simultaneously
- Every compute node will receive 8 queries
- Distributed queries achieve MINDOP by default
- Sets MAXDOP to 4 for each Distribution

DOP – Min & Max Values

# Compute Nodes	# Distributions	MINDOP	MAXDOP (Physical Query)	MAXDOP (Logical Query)
2	16	16	4	64
3	24	24	4	96
4	32	32	4	128
6	48	48	4	192
8	64	64	4	256
9	72	72	4	288

Being Agnostic

Better Together

- APS offered via Multiple Hardware Vendors
  - Hewlett Packard
  - Dell
  - Quanta
- Open Hadoop integration
  - Operating System
  - Location
  - Hadoop Distribution
  - Hadoop Version

Monitoring

- PDW exposes management data via DMVs
- Any monitoring platforms can be extended to poll for configuration or performance data from PDW
- Alternative is to use Management Packs for PDW Region and/or Hadoop Region

Backups

- PDW generates database backup files to a windows file share
- Backup agents can be configured to archive these backups from disk to alternate libraries

BI Tools & ISV Applications

- MSBI Tools Supported including Power BI
- Applications connect to PDW via
  - ADO.NET
  - Native Client (ODBC / OLEDB)
  - JDBC
- Data is transferred via TDS

Microsoft