Deep Dive: Amazon Redshift

Julien Simon
Principal Technical Evangelist
julsimon@amazon.fr
@julsimon

amazon web services

Pop-up Loft
TEL AVIV

Agenda

- Architecture overview

- Optimization
    - Schema / table design
    - Ingestion
    - Maintenance and database tuning

Architecture overview

Amazon Redshift

a lot faster
a lot simpler
a lot cheaper

Relational data warehouse

Massively parallel; petabyte scale

Fully managed

HDD and SSD platforms

$1,000/TB/year; starts at $0.25/hour

Selected Amazon Redshift customers

BEACHMINT

Amazon Redshift architecture

Leader node

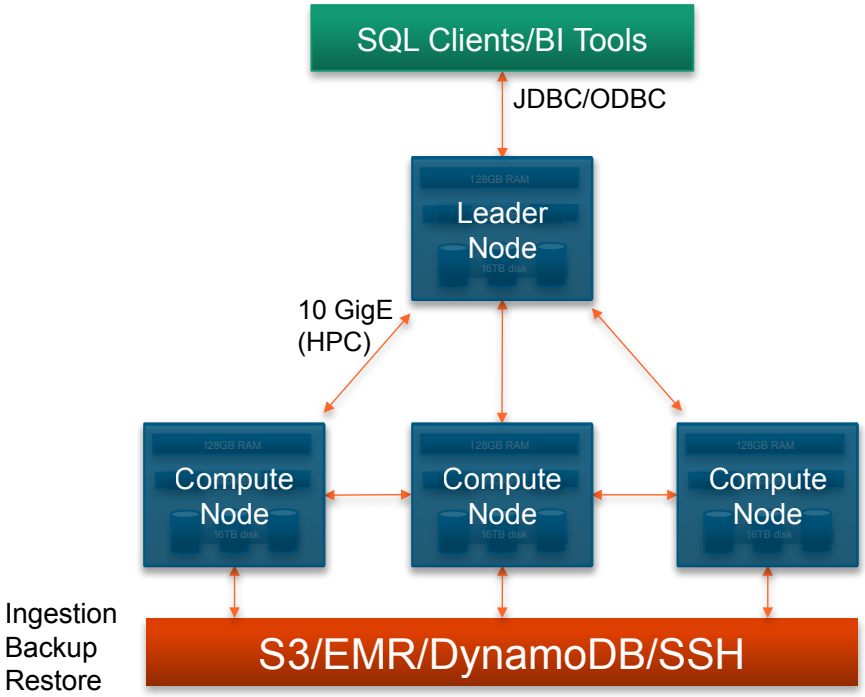
- SQL endpoint
- Stores metadata
- Coordinates query execution

Compute nodes

- Local, columnar storage
- Executes queries in parallel
- Load, backup, restore via Amazon S3; load from Amazon DynamoDB, Amazon EMR, or SSH

Two hardware platforms

- Optimized for data processing
- DS2: HDD; scale from 2 TB to 2 PB
- DC1: SSD; scale from 160 GB to 326 TB



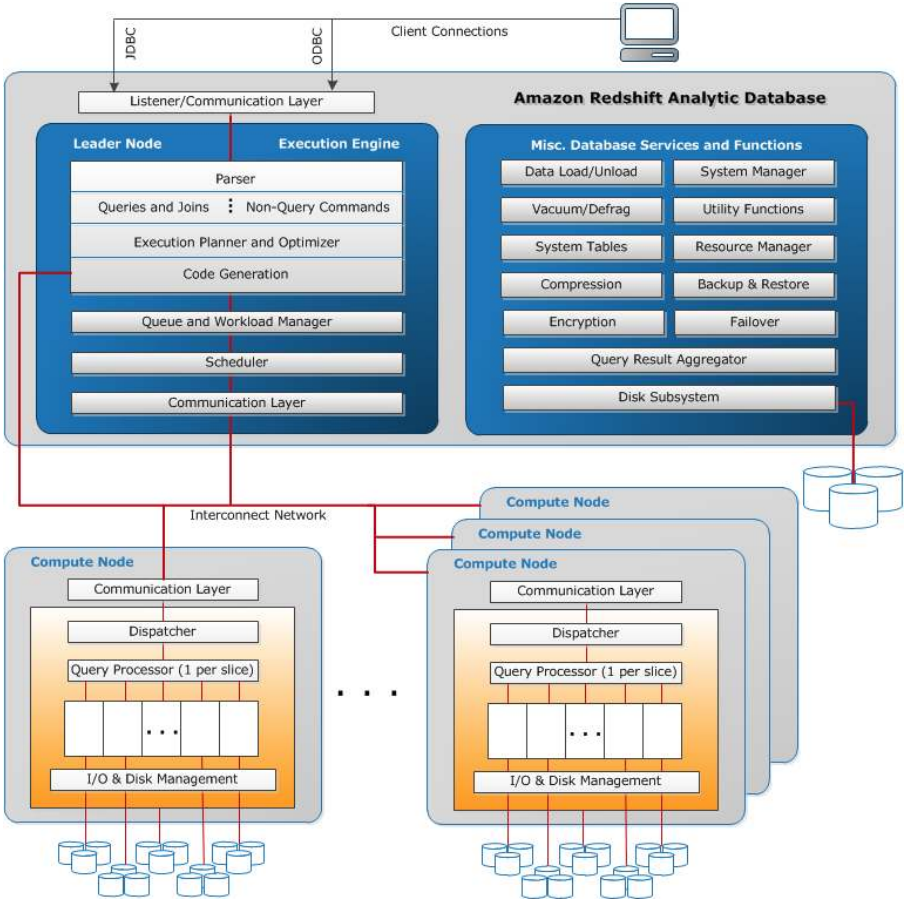
A deeper look at compute node architecture

Each node contains multiple slices

- DS2 – 2 slices on XL, 16 on 8 XL
- DC1 – 2 slices on L, 32 on 8 XL

Each slice is allocated CPU and table data

Each slice processes a piece of the workload in parallel



https://docs.aws.amazon.com/fr_fr/redshift/latest/dg/c_high_level_system_architecture.html

The best way to look through large amounts of data is to NOT look through large amounts of data

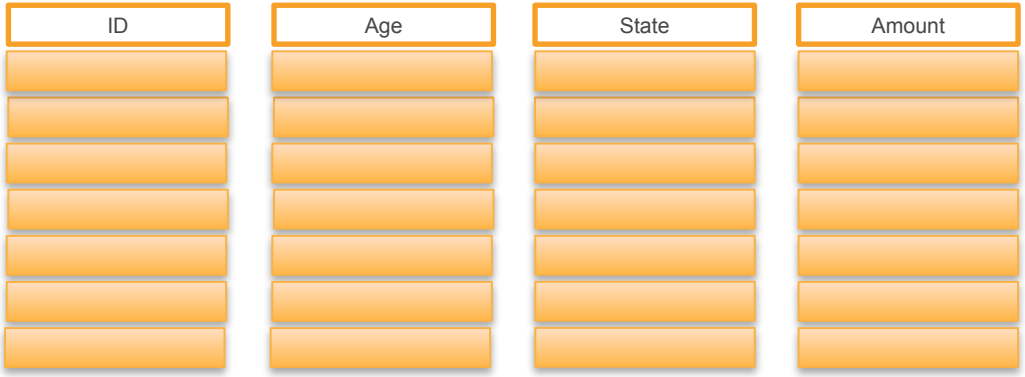
Amazon Redshift dramatically reduces I/O

Column storage

Data compression

Zone maps

ID	Age	State	Amount
123	20	CA	500
345	25	WA	250
678	40	FL	125
957	37	WA	375



Calculating SUM(Amount) with row storage:

- Need to read everything
- Unnecessary I/O

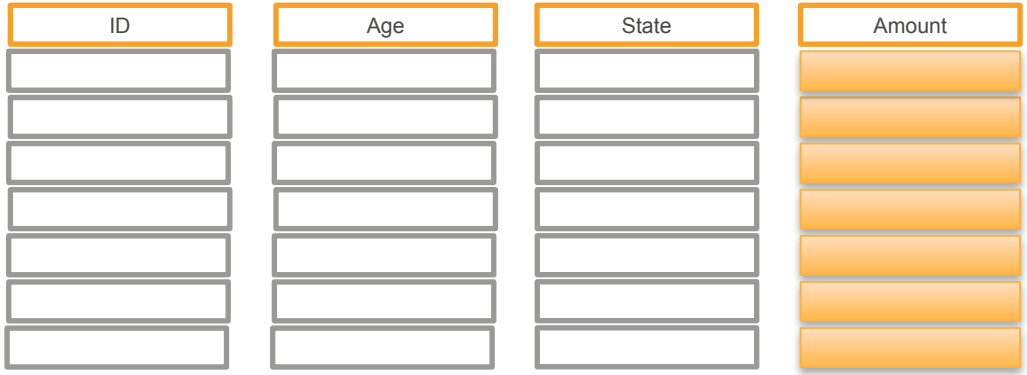
Amazon Redshift dramatically reduces I/O

Column storage

Data compression

Zone maps

ID	Age	State	Amount
123	20	CA	500
345	25	WA	250
678	40	FL	125
957	37	WA	375



Calculating SUM(Amount) with column storage:

- Only scan the necessary blocks

Amazon Redshift dramatically reduces I/O

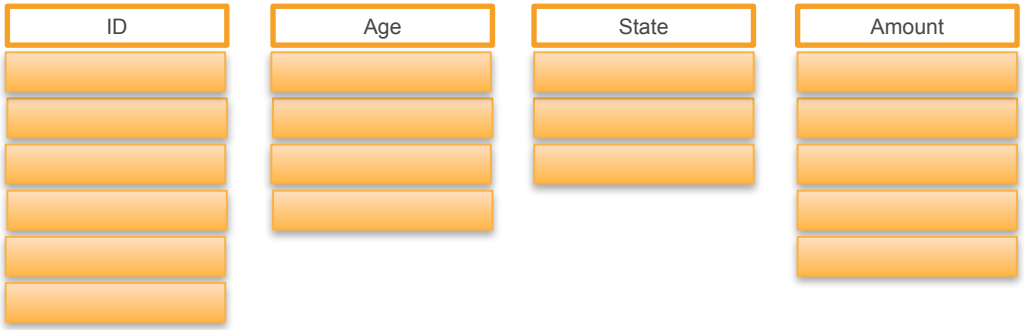
Column storage

Data compression

Zone maps

```
analyze compression orders;

  Table |   Column    | Encoding
--------+-------------+----------
 orders | id          | mostly32
 orders | age         | mostly32
 orders | state       | lzo
 orders | amount      | mostly32
```



Columnar compression

- Effective thanks to similar data
- Reduces storage requirements
- Reduces I/O

https://docs.aws.amazon.com/redshift/latest/dg/c_Compression_encodings.html

Column storage: less I/O, more compression

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

101259797|SMITH|88|899 FIRST ST|JUNO|AL|892375862|CHIN|37|16137 MAIN ST|POMONA|CA|318370701|HANDU|12|42 JUNE ST|CHICAGO|IL

Block 1 Block 2 Block 3

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

101259797|892375862|318370701|468248180|378568310|231346875|317346551|770336528|277332171|455124598|735885647|387586301

Block 1

https://docs.aws.amazon.com/fr_fr/redshift/latest/dg/c_columnar_storage_disk_mem_mgmnt.html

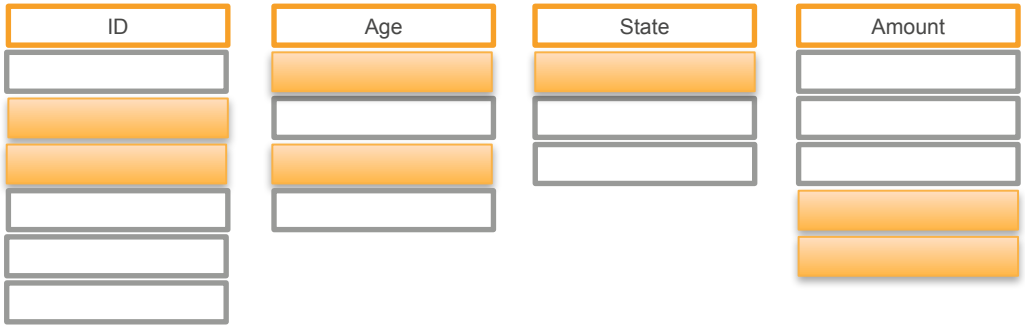
Amazon Redshift dramatically reduces I/O

Column storage

Data compression

Zone maps

- In-memory block metadata
- Contains per-block MIN and MAX value
- Effectively ignores blocks that don't contain data for a given query
- Minimizes unnecessary I/O



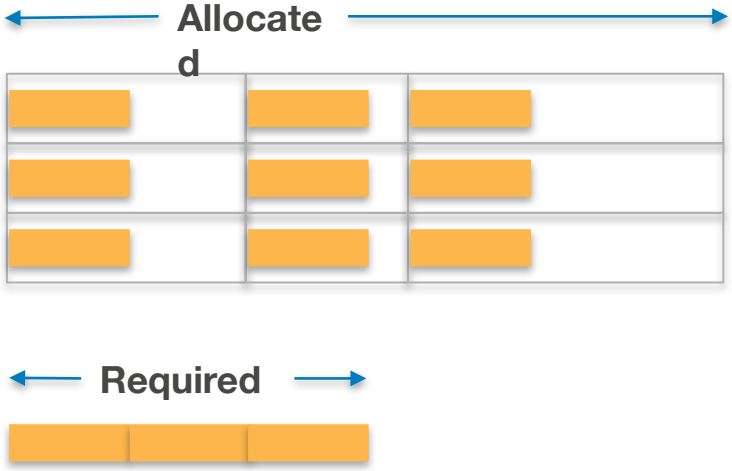
- Zone maps work best with sort keys ☺

Optimization: schema design

Distribution
Sorting
Compression

Keep Your Columns as Narrow as Possible

- Buffers allocated based on declared column width
- Wider than needed columns mean memory is wasted
- Fewer rows fit into memory; increased likelihood of queries spilling to disk
- Check `SVV_TABLE_INFO(max_varchar)`



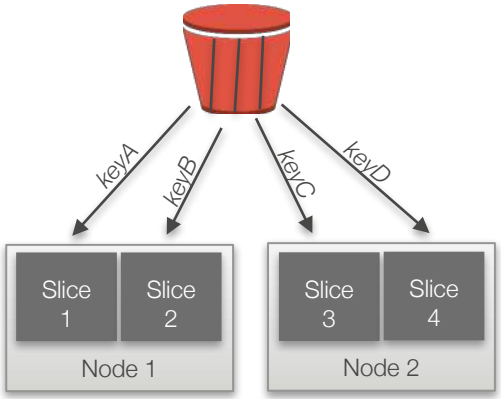
https://docs.aws.amazon.com/redshift/latest/dg/r_SVV_TABLE_INFO.html

Goals of distribution

- Distribute data evenly for parallel processing
- Minimize data movement: co-located joins, localized aggregations

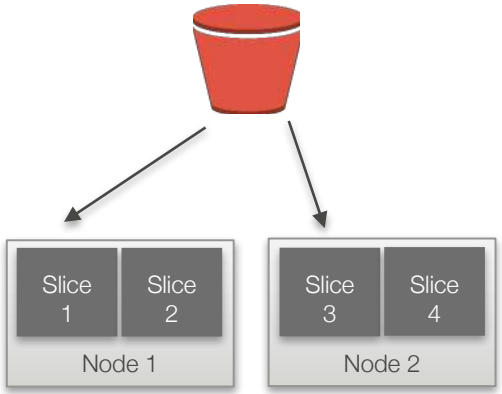
Distribution key

Same key to same location



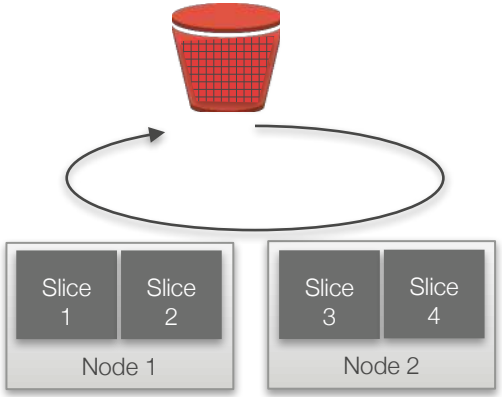
All

Full table data on first slice of every node



Even

Round-robin distribution



Choosing a distribution style

Key

- Large fact tables
- Rapidly changing tables used in joins
- Localize columns used within aggregations

Even

- Tables not frequently joined or aggregated
- Large tables without acceptable candidate keys

All

- Have slowly changing data
- Reasonable size (i.e., few millions but not 100s of millions of rows)
- No common distribution key for frequent joins
- Typical use case: joined dimension table without a common distribution key

Table skew

- If data is not spread evenly across slices, you have table skew
- Workload is unbalanced: some nodes will word harder than others
- Query is as fast as the slowest slice…

https://docs.aws.amazon.com/redshift/latest/dg/c_analyzing-table-design.html

Goals of sorting

- Physically sort data within blocks and throughout a table
- Enable range scans to ignore blocks by leveraging zone maps
- Optimal SORTKEY is dependent on:
  - Query patterns
  - Data profile
  - Business requirements

Choosing a SORTKEY

- Primarily as a query predicate (date, identifier, …)
  - Optionally, choose a column frequently used for aggregates
  - Optionally, choose same as distribution key column for most efficient joins (merge join)

COMPOUND

- Most common
- 1st column in compound key always used
- Prefix / Time-series data

INTERLEAVED

- No common filter criteria
- Key columns are “equivalent”
- No prefix / Non time-series data

https://docs.aws.amazon.com/redshift/latest/dg/t_Sorting_data-compare-sort-styles.html

Compressing data

- COPY automatically analyzes and compresses data when loading into empty tables
- CREATE TABLE AS supports automatic compression (Nov'16)
- ANALYZE COMPRESSION estimates storage space savings (Nov'16)
    - Be careful: this locks the table!
    - Changing column encoding requires a table rebuild
- Zstandard compression encoding, very good for large VARCHAR like product descriptions, JSON, etc. (Jan'17)

https://aws.amazon.com/about-aws/whats-new/2016/11/amazon-redshift-announces-new-data-compression-connection-management-and-data-loading-features/
https://aws.amazon.com/about-aws/whats-new/2017/01/amazon-redshift-now-supports-the-zstandard-high-data-compression-encoding-and-two-new-aggregate-functions/

Automatic compression is a good thing (mostly)

"*The definition of insanity is doing the same thing over and over again, but expecting different results*".

- Albert Einstein

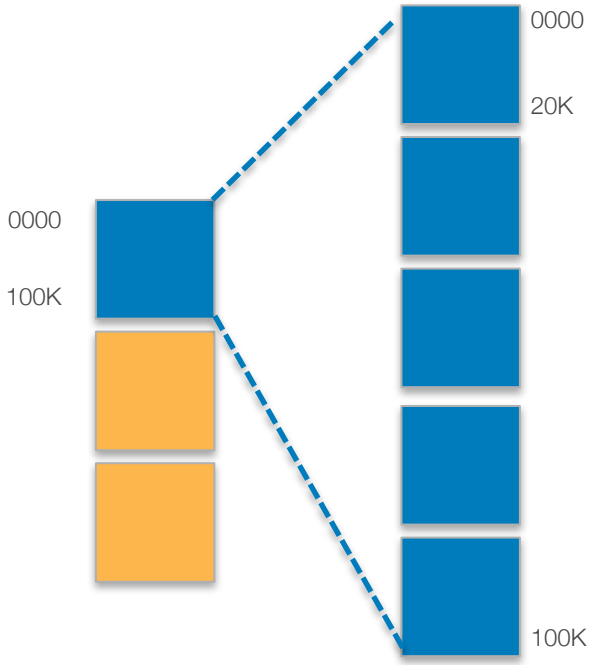
If you have a regular ETL process and you use temporary tables or staging tables, turn off automatic compression

- COMPDUPDATE OFF in COPY command
- Use ANALYZE COMPRESSION to determine the right encodings
- Use CREATE TABLE … LIKE with the appropriate compression setting

Automatic compression is a good thing (mostly)

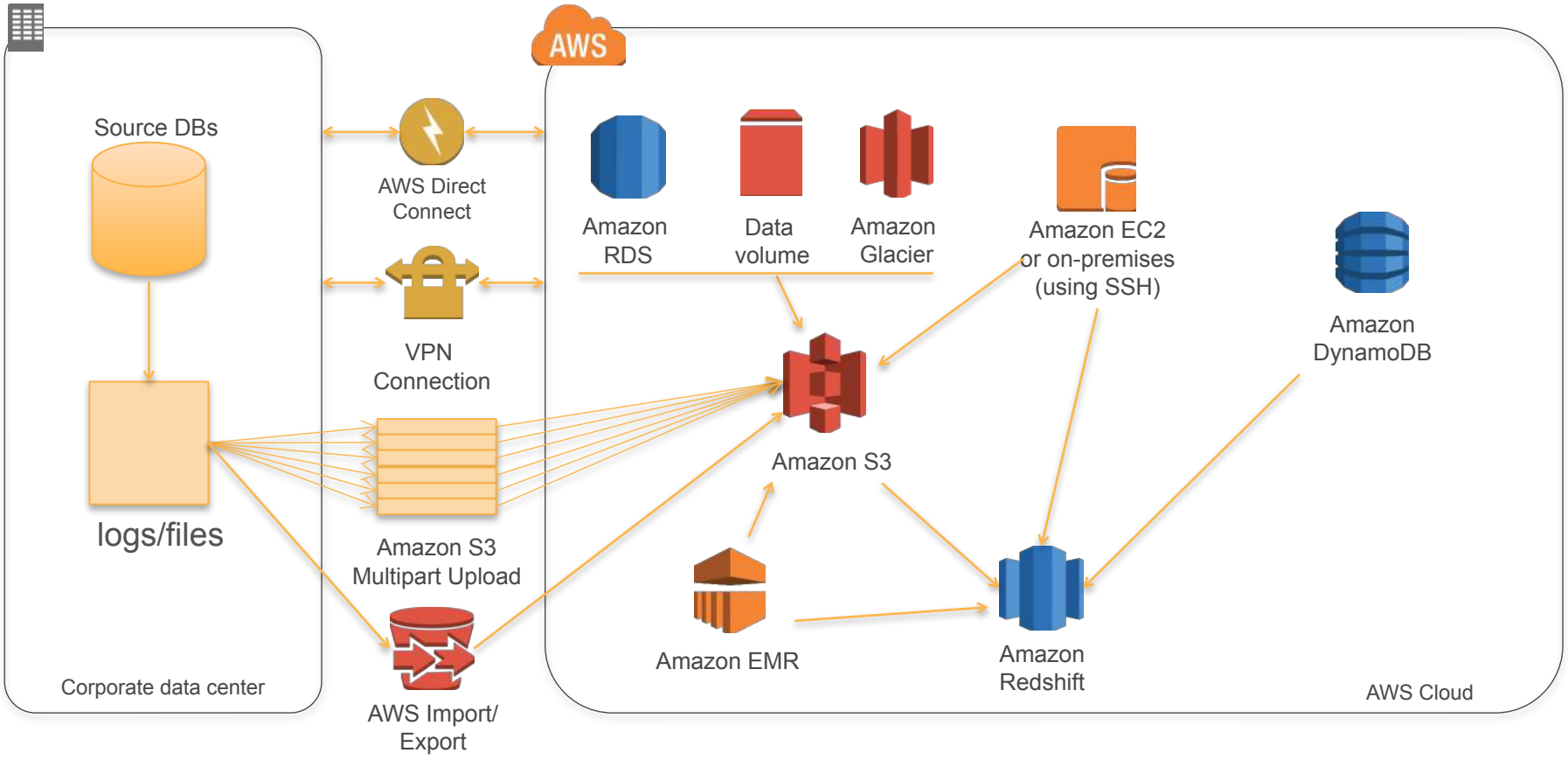
- From the zone maps we know:
  - Which blocks contain the range
  - Which row offsets to scan
- If SORTKEYs are much more compressed than other columns:
  - Many rows per block
  - Large row offset
  - Unnecessary I/O

Consider reducing/not using compression on SORTKEYs
→ less SORTKEYs per block
→ smaller offsets
→ finer granularity
→ less I/O



Optimization: ingestion

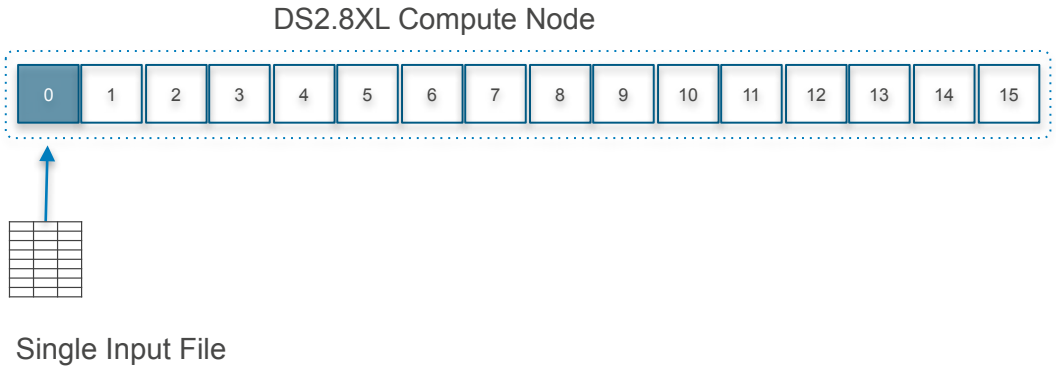
Amazon Redshift loading data overview



Parallelism is a function of load files

Each slice can load one file at a time:

- Streaming decompression
- Parse
- Distribute
- Write



A single input file means only one slice is ingesting data

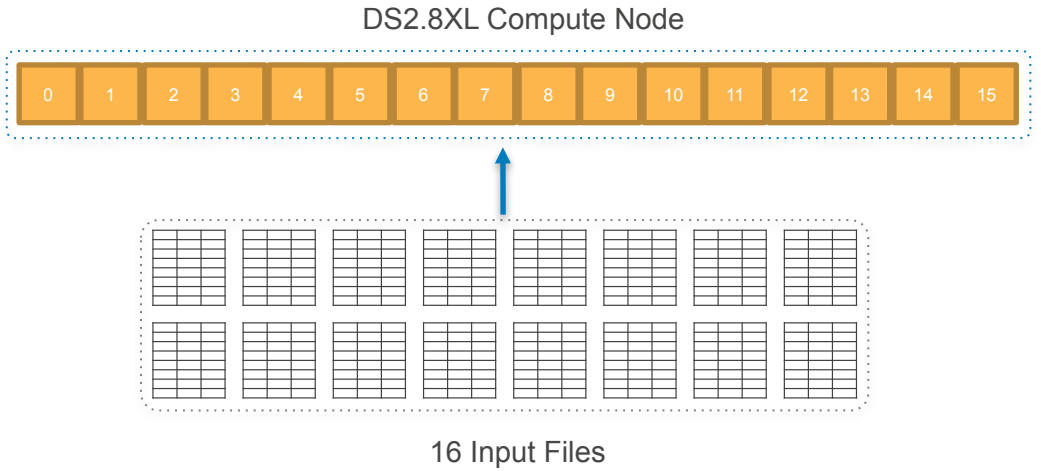
Realizing only partial cluster usage as 6.25% of slices are active

Maximize throughput with multiple files

Use at least as many input files as there are slices in the cluster

With 16 input files, all slices are working so you maximize throughput

COPY continues to scale linearly as you add nodes



Optimization: performance tuning

Statistics

- Amazon Redshift query optimizer relies on up-to-date statistics
- Statistics are necessary only for data that you are accessing
- Updated stats are important on:
    - SORTKEY
    - DISTKEY
    - Columns in query predicates

ANALYZE tables regularly

- After every single load for popular columns
- Weekly for all columns

- Look at `SVV_TABLE_INFO(stats_off)` for stale stats
- Look at `stl_alert_event_log` for missing stats
- Look for uncompressed columns

https://docs.aws.amazon.com/redshift/latest/dg/t_Analyzing_tables.html
https://docs.aws.amazon.com/redshift/latest/dg/r_STL_ALERT_EVENT_LOG.html

VACUUM tables regularly

- Weekly is a good target
- Number of unsorted blocks is a good indicator
- Look at `SVV_TABLE_INFO(unsorted, empty)`
- Deep copy might be faster for high percent unsorted
  - 20% unsorted or more : usually faster to deep copy
  - CREATE TABLE AS

https://docs.aws.amazon.com/redshift/latest/dg/r_VACUUM_command.html

Cluster status: commits and WLM

WLM queue

- Identify short/long-running queries and prioritize them
- Define multiple queues to route queries appropriately
- Default concurrency of 5
- Use `wlm_apex_hourly.sql` to tune WLM based on peak concurrency requirements

Commit queue

- How long is your commit queue?
- Identify needless transactions
- Group dependent statements within a single transaction
- Offload operational workloads
- `STL_COMMIT_STATS`

Open source tools

https://github.com/awslabs/amazon-redshift-utils
https://github.com/awslabs/amazon-redshift-monitoring
https://github.com/awslabs/amazon-redshift-udfs

Admin scripts

Collection of utilities for running diagnostics on your cluster

Admin views

Collection of utilities for managing your cluster, generating schema DDL, etc.

ColumnEncodingUtility

Gives you the ability to apply optimal column encoding to an established schema with data already loaded

Tuning Your Workload

top_queries.sql

db	n_qry	qrytext	min	max	avg	total	max_query_id	last_run	aborted	event
demo	1	COPY flights from 's3://data-airline-performance/' credentials '' CSV DELIMITER	637.00	637.00	637.00	637.00	757	2015-07-08	0	
demo	29	analyze compression phase 1	1.00	2.00	1.62	47.00	755	2015-07-08	0	
demo	2	padb_fetch_sample: select * from flights	1.00	23.00	12.00	24.00	787	2015-07-08	0	Filter
demo	1	COPY ANALYZE flights	18.00	18.00	18.00	18.00	694	2015-07-08	0	
demo	29	analyze compression phase 2	0.00	3.00	0.20	6.00	756	2015-07-08	0	
demo	13	select * from testschema.category_stage	0.00	2.00	0.15	2.00	1208	2015-07-08	0	Stats
demo	1	padb_fetch_sample: select count(*) from flights	1.00	1.00	1.00	1.00	785	2015-07-08	0	
demo	2	insert into testschema.category_stage values (12, 'Concerts', 'Comedy', 'All sta	0.00	1.00	0.50	1.00	1183	2015-07-08	0	Stats

perf_alerts.sql

table	minutes	rows	event	solution	sample_query	count
			Missing query planner statistics	Run the ANALYZE command	1208	16
flights	1	34236	Very selective query filter	Review the choice of sort key to enable range restricted sca	787	8

analyze-vacuum-schema.py

Must-read articles

Amazon Redshift Engineering's Advanced Table Design Playbook

https://aws.amazon.com/blogs/big-data/amazon-redshift-engineerings-advanced-table-design-playbook-preamble-prerequisites-and-prioritization/

Tutorial: Tuning Table Design

https://docs.aws.amazon.com/redshift/latest/dg/tutorial-tuning-tables.html

Top 10 Performance Tuning Techniques for Amazon Redshift

https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-techniques-for-amazon-redshift/

Interleaved Sort Keys

https://aws.amazon.com/blogs/aws/quickly-filter-data-in-amazon-redshift-using-interleaved-sorting/

Thank You

Julien Simon
julsimon@amazon.fr
@julsimon

Your feedback is important to us!

amazon web services

Pop-up Loft
TEL AVIV