BDT401

Amazon Redshift Deep Dive

Tuning and Best Practices

Eric Ferreira, AWS
Ari Miller, TripAdvisor

October 2015

amazon web services

What to Expect from the Session

Architecture Review

Ingestion

- COPY
- PKs and Manifest
- Data Hygiene
- Auto Compression / Sort key Compression

Recent Features

- New Functions
- UDFs
- Interleaved Sort keys

Migration Tips

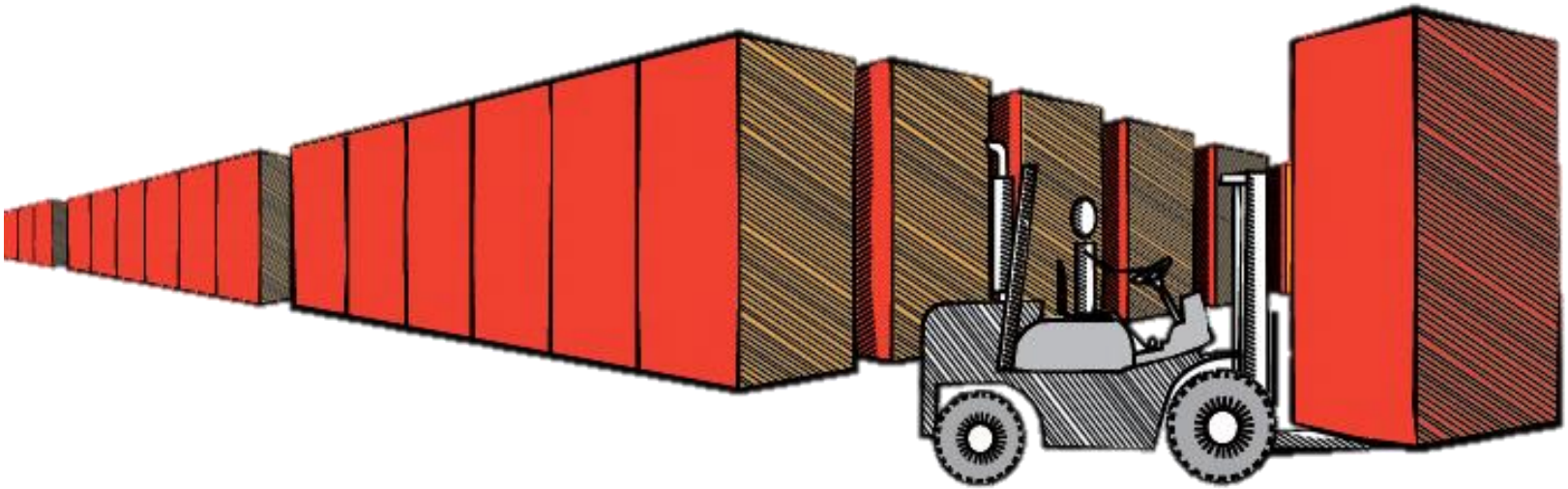
Workload Tuning

- Workload
- WLM
- Console

Sharing Amazon Redshift

- Custom Computation Clusters
- Infra/Automation
- Monitoring
- Views
- Queue Management

Amazon Redshift



Fast, simple, petabyte-scale data warehousing for less than $1,000/TB/year

Amazon Redshift Delivers Performance

"Redshift is twenty times faster than Hive" (5x – 20x reduction in query times) link

…[Redshift] performance has blown away everyone here (we generally see 50-100x speedup over Hive). link

"We saw…2x improvement in query times"

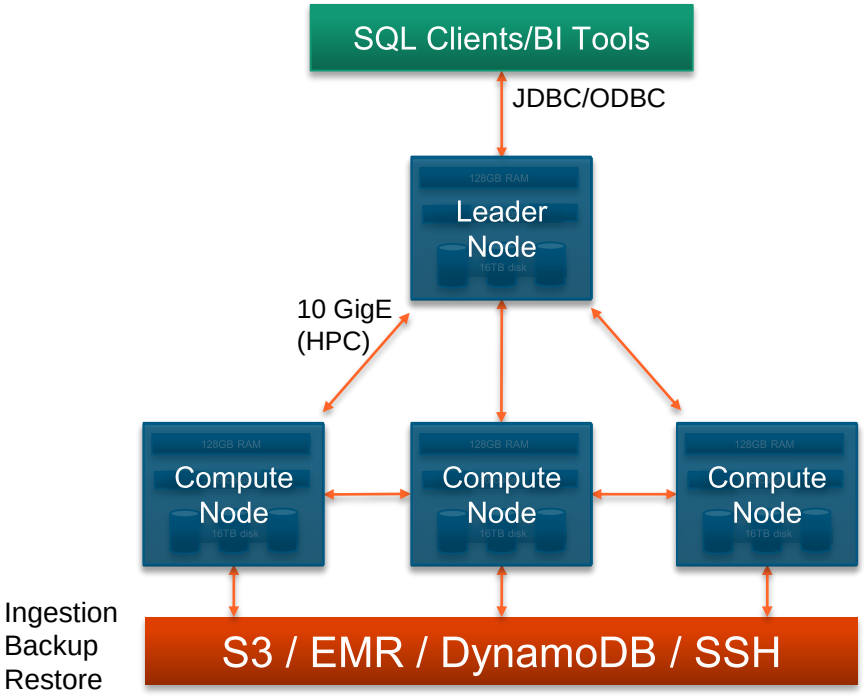
We regularly process multibillion row datasets and we do that in a matter of hours. link

"Queries that used to take hours came back in seconds. Our analysts are orders of magnitude more productive." (20x – 40x reduction in query times) link

"Did I mention it's ridiculously fast? We'll be using it immediately to provide our analysts an alternative to Hadoop."

"Team played with Redshift today and concluded it is ****** awesome. Un-indexed complex queries returning in < 10s."

Amazon Redshift System Architecture

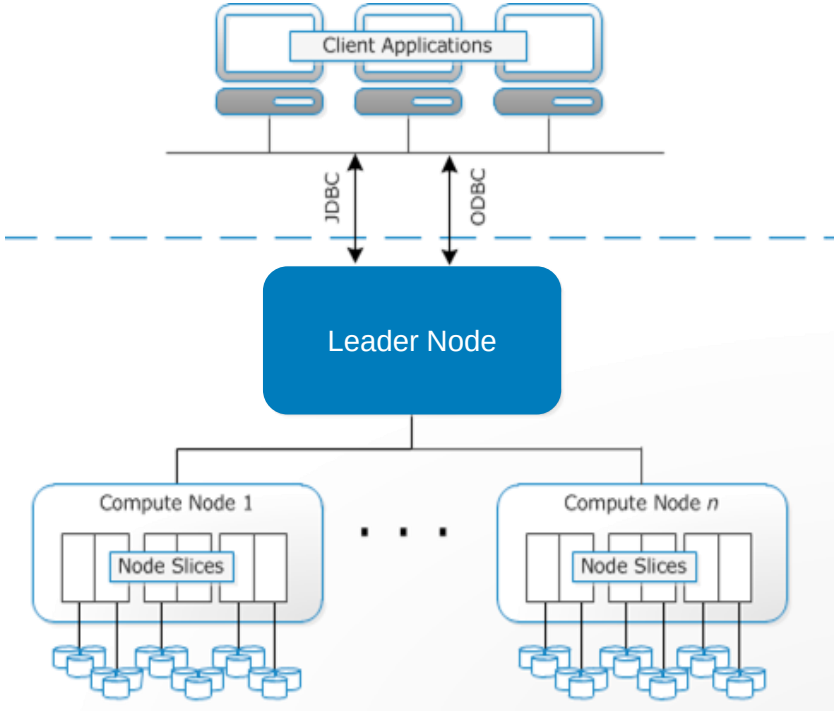


A Deeper Look at Compute Node Architecture

Dense Compute Nodes

Large
- 2 slices/cores
- 15GB RAM
- 160GB SSD

8XL
- 32 slices/cores
- 244 GB RAM
- 2.56TB SSD



Dense Storage Nodes

X-Large
- 2 slices/ 4 cores
- 31GB RAM
- 2TB HDD

8XL
- 16 slices/ 36 cores
- 244 GB RAM
- 16TB HDD

Ingestion

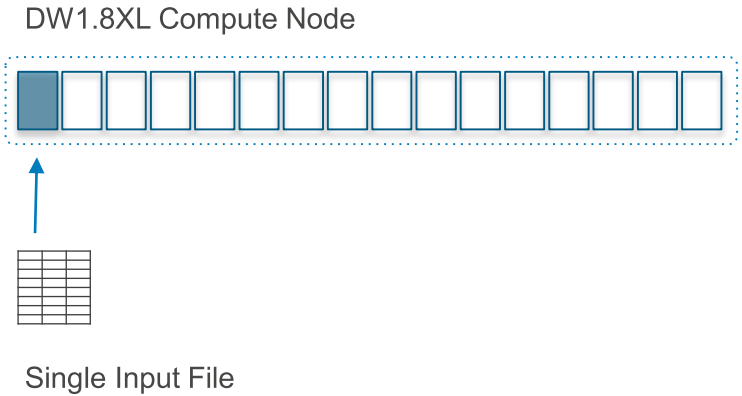
Use Multiple Input Files to Maximize Throughput

COPY command

Each slice loads one file at a time

A single input file means
only one slice is ingesting data

Instead of 100MB/s,
you're only getting 6.25MB/s



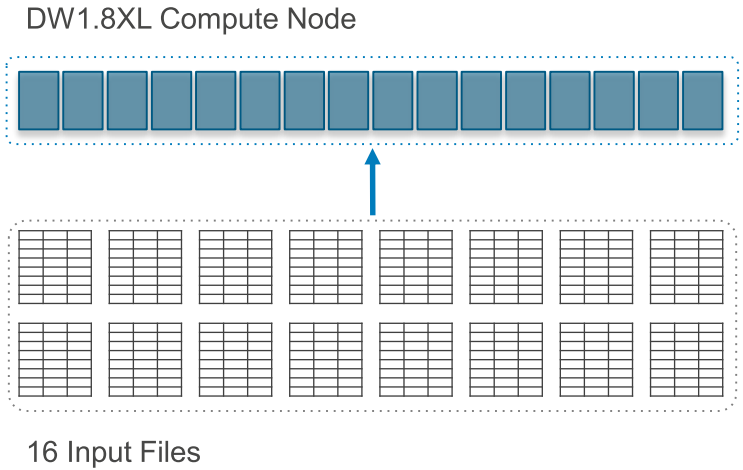
Use Multiple Input Files to Maximize Throughput

COPY command

You need at least as many input files as you have slices

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

Get 100 MB/s per node; scale linearly as you add nodes



Primary Keys and Manifest Files

Amazon Redshift doesn’t enforce primary key constraints

- If you load data multiple times, Amazon Redshift won’t complain
- If you declare primary keys in your DML, the optimizer will expect the data to be unique

Use manifest files to control exactly what is loaded and how to respond if input files are missing

- Define a JSON manifest on Amazon S3
- Ensures that the cluster loads exactly what you want

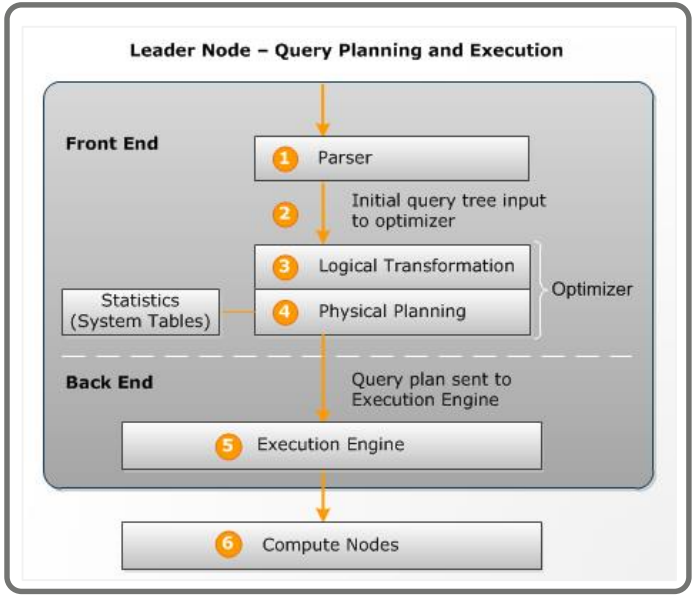
Data Hygiene

Analyze tables regularly

- Every single load for popular columns
- Weekly for all columns
- Look SVV_TABLE_INFO(stats_off) for stale stats
- Look stl_alert_event_log for missing stats

Vacuum tables regularly

- Weekly is a good target
- Number of unsorted blocks as trigger
- Look SVV_TABLE_INFO(unsorted, empty)
- Deep copy might be faster for high percent unsorted (20% unsorted usually is faster to deep copy)



Automatic Compression is a Good Thing (Mostly)

Better performance, lower costs

Samples data automatically when COPY into an empty table

- Samples up to 100,000 rows and picks optimal encoding

Regular ETL process using temp or staging tables:
Turn off automatic compression

- Use analyze compression to determine the right encodings
- Bake those encodings into your DML

Be Careful When Compressing Your Sort Keys

Zone maps store min/max per block

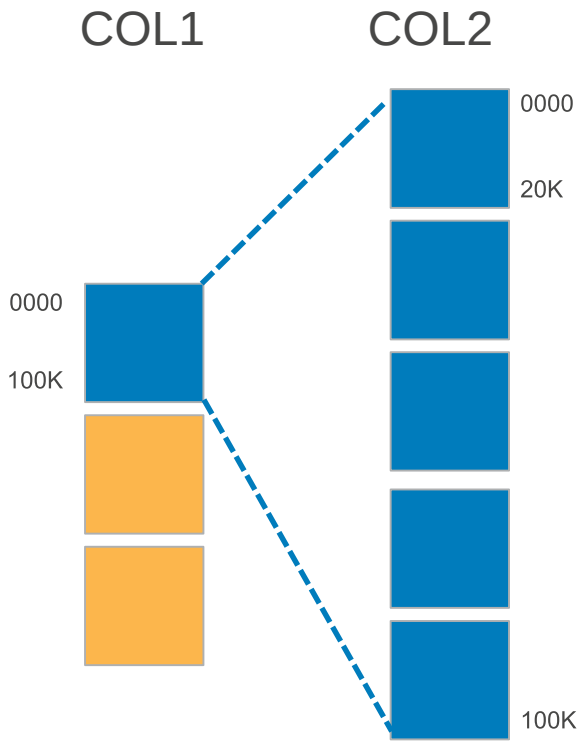
After we know which block(s) contain the range, we know which row offsets to scan

Highly compressed sort keys means many rows per block

You'll scan more data blocks than you need

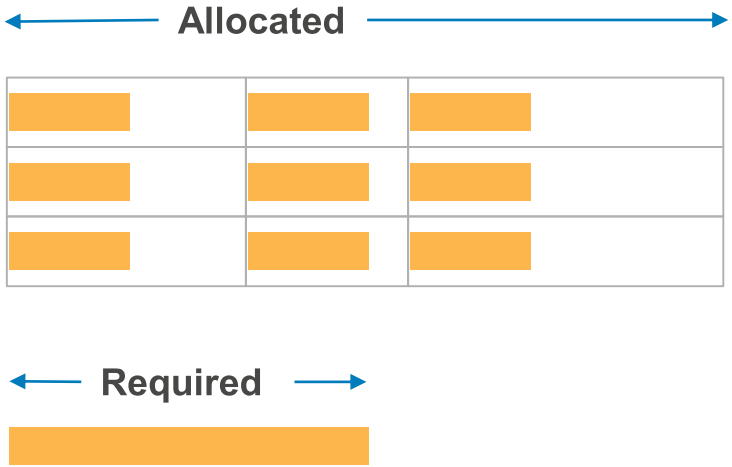
If your sort keys compress significantly more than your data columns, you may want to skip compression of sortkey column(s)

Check SVV_TABLE_INFO(skew_sortkey1)



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)



Recent Features

New SQL Functions

We add SQL functions regularly to expand Amazon Redshift’s query capabilities

Added 25+ window and aggregate functions since launch, including:

- LISTAGG
- [APPROXIMATE] COUNT
- DROP IF EXISTS, CREATE IF NOT EXISTS
- REGEXP_SUBSTR, _COUNT, _INSTR, _REPLACE
- PERCENTILE_CONT, _DISC, MEDIAN
- PERCENT_RANK, RATIO_TO_REPORT

We’ll continue iterating but also want to enable you to write your own

Scalar User Defined Functions

You can write UDFs using Python 2.7

- Syntax is largely identical to PostgreSQL UDF syntax
- System and network calls within UDFs are prohibited

Comes with Pandas, NumPy, and SciPy pre-installed

- You'll also be able import your own libraries for even more flexibility

Scalar UDF Example

```
CREATE FUNCTION f_hostname (VARCHAR url)
  RETURNS varchar
IMMUTABLE AS $$
  import urlparse
  return urlparse.urlparse(url).hostname
$$ LANGUAGE plpythonu;
```

Rather than using complex REGEX expressions, you can import standard Python URL parsing libraries and use them in your SQL

Scalar UDF Examples from Partners

Looker: http://www.looker.com/blog/amazon-redshift-user-defined-functions

Periscope: https://www.periscope.io/blog/redshift-user-defined-functions-python.html

awsmarketplace

1-click deployment to launch, on multiple regions around the world

Pay-as-you-go pricing with no long term contracts required

Data Integration

ATTUNITY

syncsort

Matillion Business Intelligence

informatica

Semarchy THE DATA CONVERGENCE COMPANY

AURIX

Advanced Analytics

ZEMENTIS EMPOWERING BIG DATA INSIGHTS

sas

Atigeo

Digital Reasoning

bigml

PredictionIO

Business Intelligence

TIBCO Spotfire

tableau SOFTWARE

Logi ANALYTICS

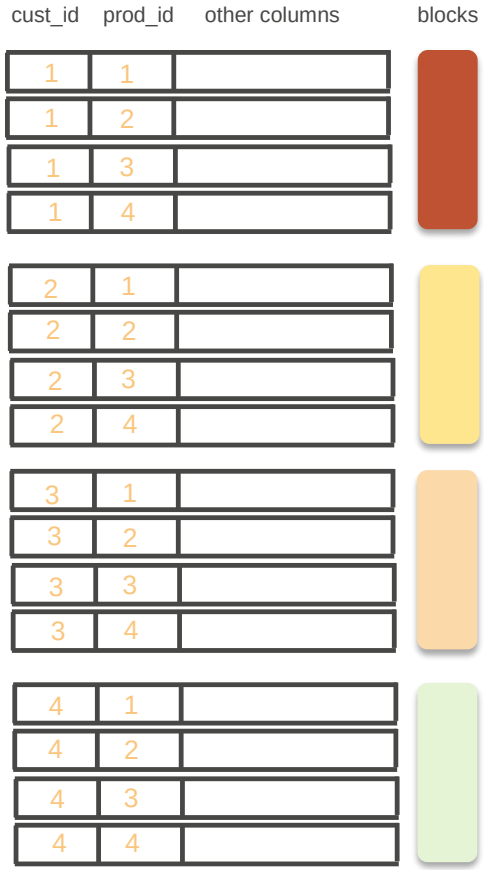
SAP Business Objects

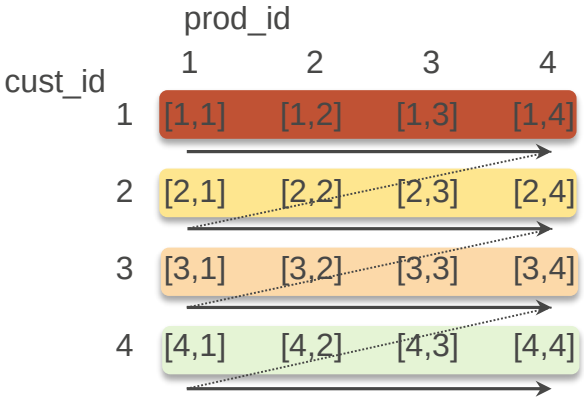
MicroStrategy

looker

Interleaved Sort Keys

Compound Sort Keys






```
Select sum(amt)
From big_tab
Where cust_id = (1234);
```

```
Select sum(amt)
From big_tab
Where prod_id = (5678);
```

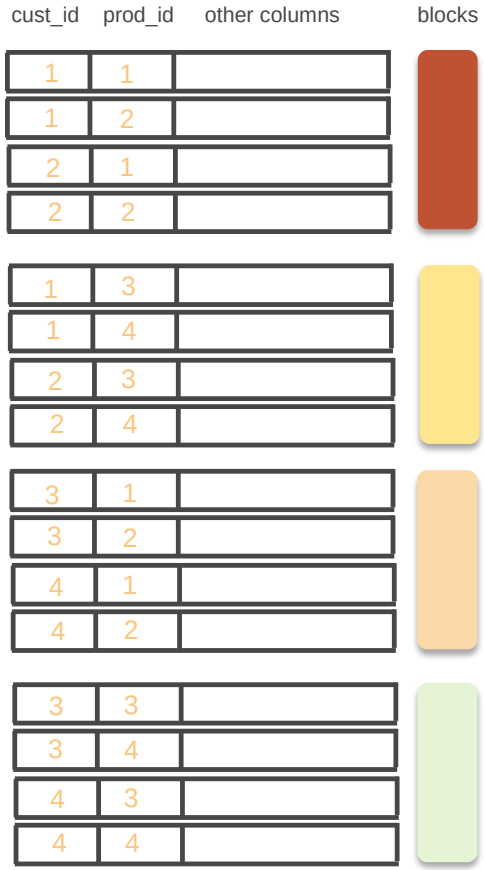
Records in Amazon Redshift are stored in blocks.

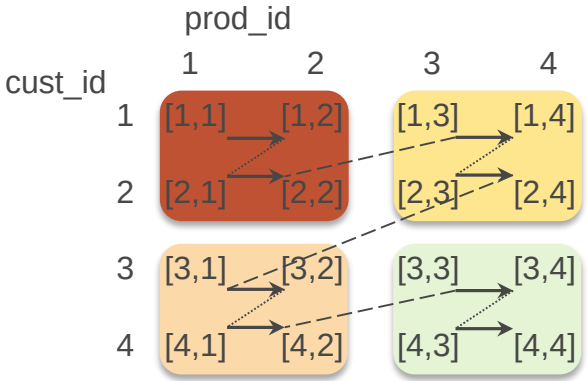
For this illustration, let's assume that four records fill a block

Records with a given cust_id are all in one block

However, records with a given prod_id are spread across four blocks

Interleaved Sort Keys





Records with a given cust_id are spread across two blocks

Records with a given prod_id are also spread across two blocks

Data is sorted in equal measures for both keys

Usage

```
[[ COMPOUND | INTERLEAVED ] SORTKEY ( column_name [, ...] ) ]
```

New keyword ‘INTERLEAVED’ when defining sort keys

- Existing syntax will still work and behavior is unchanged
- You can choose up to 8 columns to include and can query with any or all of them

No change needed to queries

We’re just getting started with this feature

- Benefits are significant; load penalty is higher than we’d like and we’ll fix that quickly
- Check SVV_INTERLEAVED_COLUMNS(interleaved_skew) to decide when to VACUUM REINDEX
- A value greater than 5 will indicate the need to VACUUM REINDEX

Migrating Existing Workloads

Forklift = BAD

Typical ETL/ELT on Legacy DW

- One file per table, maybe a few if too big
- Many updates (“massage” the data)
- Every job clears the data, then load
- Count on PK to block double loads
- High concurrency of load jobs
- Small table(s) to control the job stream

Two Questions to Ask

Why you do what you do?

- Many times, they don’t even know

What is the customer need?

- Many times, needs do not match current practice
- You might benefit from adding other AWS services

On Amazon Redshift

Updates are delete + insert of the row

- Deletes just mark rows for deletion

Blocks are immutable

- Minimum space used is one block per column, per slice

Commits are expensive

- 4 GB write on 8XL per node
- Mirrors WHOLE dictionary
- Cluster-wide serialized

On Amazon Redshift

- Not all aggregations created equal
  - Pre-aggregation can help
  - Order on group by matters
- Concurrency should be low for better throughput
- Caching layer for dashboards is recommended
- WLM parcels RAM to queries. Use multiple queues for better control.

Resultset Metadata

Using SQL or JDBC, you have access to column names.

```
ResultSet rs = stmt.executeQuery("SELECT * FROM emp");
ResultSetMetaData rsmd = rs.getMetaData();
String name = rsmd.getColumnName(1);
```

Unload does not provide columns names (yet).

- Use SELECT top 0…
- Instead of adding 0=1 to your WHERE clause

Open Source Tools

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

Admin Scripts

- Collection of utilities for running diagnostics on your cluster.

Admin Views

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

Column Encoding Utility

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

Analyze and Vacuum Utility

- Gives you the ability to automate VACUUM and ANALYZE operations.

Unload and Copy Utility

- Helps you to migrate data between Amazon Redshift clusters or databases.

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

Workload Management (WLM)

Concurrency and memory can now be changed dynamically.
You can have distinct values for load time and query time.

Queue	Concurrency	% Memory to Use
1	4	50%
2	4	50%

Queue	Concurrency	% Memory to Use
1	3	75%
2	4	25%

Use wlm_apex_hourly.sql to monitor “queue pressure”

Using the Console for Query Tuning

SQL

```
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (     select
        n_name as nation,
        extract(year
    from
        o_orderdate) as o_year,
        l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount

    from
        part,
        supplier,
        lineitem,
        partsupp,
        orders,
        nation
    where
        s_suppkey = l_suppkey
        and ps_suppkey = l_suppkey
        and ps_partkey = l_partkey
        and p_partkey = l_partkey
        and o_orderkey = l_orderkey
        and s_nationkey = n_nationkey
        and p_name like '%goldenrod%'     ) as profit
    group by
        nation,
        o_year
    order by
        nation,
        o_year desc     ;
```

Query Execution Details

Plan | Actual

```
XN Merge  (cost=1584237740253.27..1584237740401.14 rows=59150 width=76)
  ->  XN Network  (cost=1584237740253.27..1584237740401.14 rows=59150 width=76)
        ->  XN Sort  (cost=1584237740253.27..1584237740401.14 rows=59150 width=76)
              ->  XN HashAggregate  (cost=584237734677.76..584237735565.01 rows=59150 width=76)
                    ->  XN Hash Join DS_DIST_NONE  (cost=584187791435.09..584237725680.43 rows=1199644 width=76)
                          ->  XN Seq Scan on orders  (cost=0.00..15360000.00 rows=1536000000 width=12)
                          ->  XN Hash  (cost=584187788623.53..584187788623.53 rows=1124623 width=80)
                                ->  XN Hash Join DS_BCAST_INNER  (cost=100672761152.79..584187788623.53 rows=1124623 width=
                                      ->  XN Hash Join DS_BCAST_INNER  (cost=100672761152.48..584176761631.23 rows=1199644
                                      ->  XN Hash  (cost=0.25..0.25 rows=25 width=33)
                                            ->  XN Seq Scan on lineitem  (cost=0.00..61440087.04 rows=6144008704 width=52)
                                            ->  XN Hash  (cost=100672756119.50..100672756119.50 rows=1006596 width=39)
                                                  ->  XN Hash Join DS_DIST_INNER  (cost=12813253.54..100672756119.50 rows=1
                                                        ->  XN Seq Scan on supplier  (cost=0.00..102400.00 rows=10240000 wi
                                                        ->  XN Hash  (cost=12810737.05..12810737.05 rows=1006596 width=31)
```

AWS re:Invent

Sharing Amazon Redshift

Amazon Redshift performance as a shared resource

Ari Miller, TripAdvisor

October 2015

amazon web services

The Shared Resource Problem

TripAdvisor

- 375 million unique monthly visitors
- 8 ds2.8xlarge cluster
- Largest table 12 TB (1 month of data)

The Shared Resource Problem

- 460 engineers, data scientists, analysts, and product managers with personal logins to Amazon Redshift.
- > 2500 tables
- If you don't build it, they will still come
    - Company motto is "speed wins" -- no tolerance for delayed analysis
    - 293 million row table imported using INSERT statements

The Solution

Reduce contention (custom clusters)

Infrastructure/automation

Monitoring (tables/performance/usage)

Own all common processing

Custom Computation Clusters

stay out of the way

Custom Computation Clusters

Spin up specialized ephemeral clusters. Guidelines:

- Mighty mouse cluster (32 dc1.large nodes) -- great bang for the buck ($8/hr)
- 4 hour job on 8 d1s.8xlarge -> 40 minutes on mighty mouse
- Much smaller wins vs d2s
- Customize the configuration:
  - 2 slots only, to maximize memory
  - Rely on Amazon Redshift to parallelize
- Primary limitation is on the size of the copy

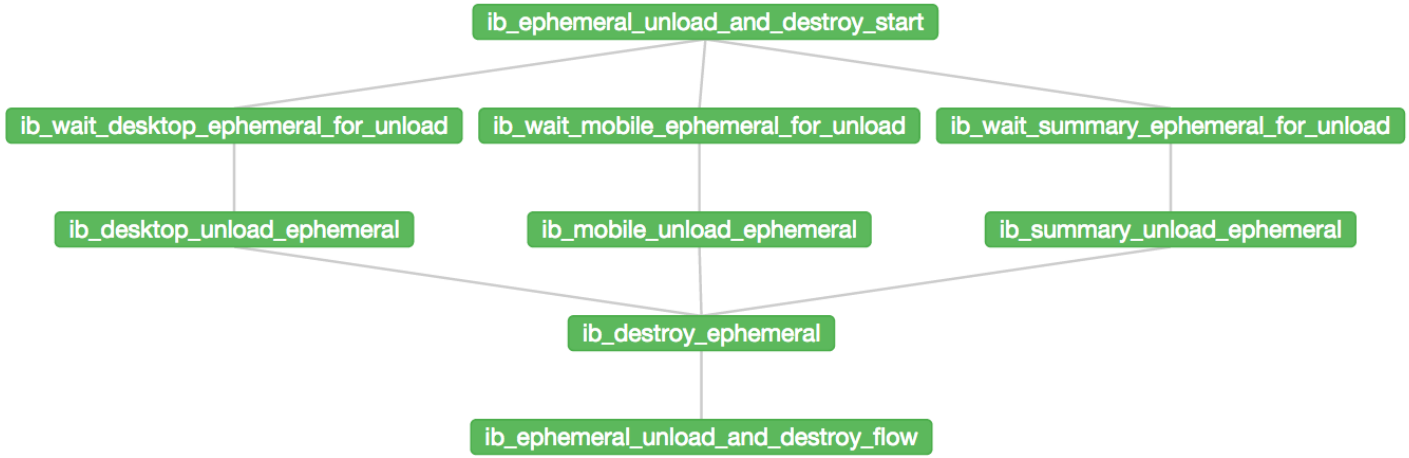
Custom Computation Clusters

Ephemeral Infrastructure (Java)

- sync_redshift_objects.sh -- create schema/tables into a new cluster from an existing cluster
- unload.sh
    - manifest file
    - automated checks for Amazon S3 consistency
- copy.sh
    - Parameters: schema, table
    - mirrors unload

Infrastructure/Automation

Azkaban – batch workflow + UI (LinkedIn)

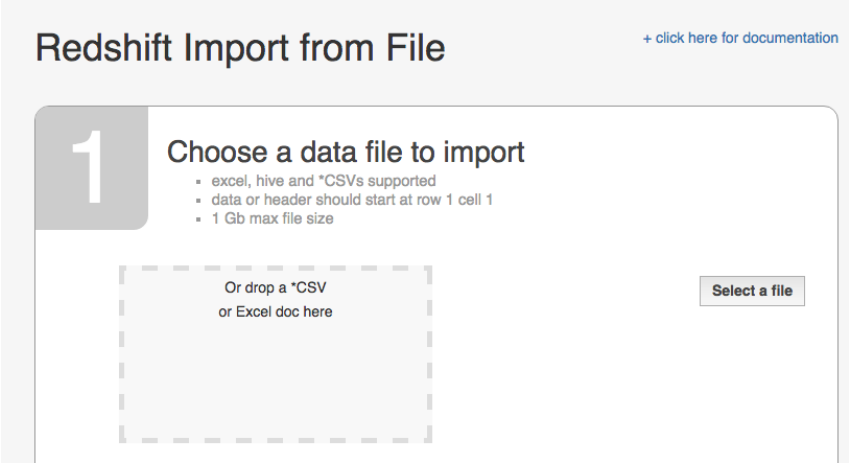


Azkaban calls data transfer infrastructure, SQL, etc.

Infrastructure/Automation

Custom UI for non engineers -- point and click.

- Import from file or Hive
- Auto-analyze source data, suggest table ddl (sort, distribution, compression), schedule

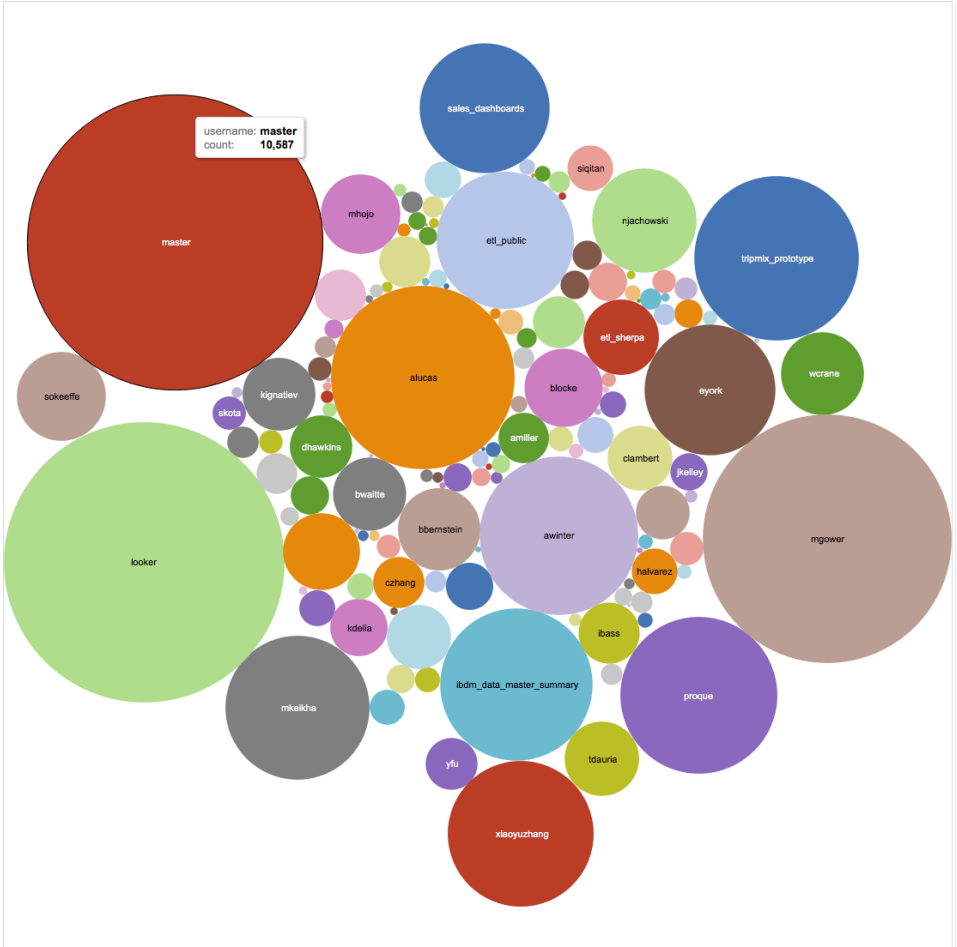


Monitoring

- Long running query alerts
  - Accounts created through a UI that maps to our LDAP
  - Every account mapped to an email address
- Daily and peak usage reports, by user and type of activity
- Daily table size, space taken up by deleted rows by table report – deep_copy.sh instead of vacuum
- Initial monitoring cost can be minimal because of psql compatibility
  - psql --quiet --html --table-attr cellspacing='0' --file $QUERY_FILE --output $RESULT_FILE

Monitoring

Tableau helps



Shared Development Cluster

- Calisi syntax allows for parallel development on a single shared development cluster
- Each developer has their own personal schema
- Any workflow can be run against their personal schema if the SQL is marked up in this way for substitution.

/*{*/sherpa/*}*/ -> jbezos

/*{*/sherpa/*}*/ -> wvogels

Shared Development Cluster

```
String.format("create schema %s %s", targetSchema,
authorizationSchema));

List<Table> tables =
syncRedshiftObjects.loadTablesForSchema(originalSche
ma);
for (Table table : tables)
{
    String tableName = table.getTable();
    String ddl =
table.getCreateTableStatement().replace(table.getSch
ema() + ".", targetSchema + ".");
```

View Performance

- Create tables by month, with views spanning the months.
- Use Calisi syntax to surround SQL using the view:

  /*{*/rio_flookback/*}*/ -> rio_flookback_aug
- Syntax is completely legitimate SQL against a view -- autocomplete columns in IDE, run.
- When run through a framework, can substitute automatically to run against the underlying monthly table where the query doesn't span multiple months.

Queue Management

- Keep it simple, the memory is reserved
  - Mission critical
  - DBA
  - Quick
  - Default
- Include a default timeout
- Groups allow you to dynamically switch default queues
- Tableau
  - Can’t run set query group
  - Use group membership to switch queues

Wrap Up

- Amazon Redshift has transformed analytics at TripAdvisor
  - Thousands of queries a day, hundreds of users
  - Tableau dashboards built on Amazon Redshift
  - Iterative real time exploration, hundreds of times faster
  - Open LDAP access for any employee
- Open sourcing these solutions
  - Feb 2016
  - Find the link here: http://engineering.tripadvisor.com/

Related Sessions

Hear from other customers discussing their Amazon Redshift use cases:

- DAT201—Introduction to Amazon Redshift (RetailMeNot)
- ISM303—Migrating Your Enterprise Data Warehouse to Amazon Redshift (Boingo Wireless and Edmunds)
- ARC303—Pure Play Video OTT: A Microservices Architecture in the Cloud (Verizon)
- ARC305—Self-Service Cloud Services: How J&J Is Managing AWS at Scale for Enterprise Workloads
- BDT306—The Life of a Click: How Hearst Publishing Manages Clickstream Analytics with AWS
- DAT308 - How Yahoo! Analyzes Billions of Events a Day on Amazon Redshift
- DAT311—Large-Scale Genomic Analysis with Amazon Redshift (Human Longevity)
- BDT314—Running a Big Data and Analytics Application on Amazon EMR and Amazon Redshift with a Focus on Security (Nasdaq)
- BDT316—Offloading ETL to Amazon Elastic MapReduce (Amgen)
- Building a Mobile App using Amazon EC2, Amazon S3, Amazon DynamoDB, and Amazon Redshift (Tinder)

Remember to complete your evaluations!

Thank you!