NARRATIV

Leveraging Redshift Spectrum for Fun and Profit

- 2017 Winner -

BEST NATIVE/CONTENT ADVERTISING PLATFORM

DIGIDAY SIGNAL AWARDS

The Digiday Signal Awards recognize the technology platforms that are bringing efficiency, effectiveness and creativity to the media and marketing processes for brands, agencies and publishers.

About This Talk

As a software engineer at a startup, I wear many hats.

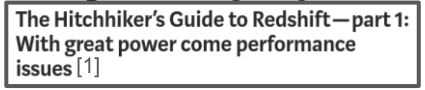
This talk is the story of:

- Big Data at a startup (or, how to spend all your cash on Redshift)
- How we took control of our redshift costs
- How you can do the same (with less pain)

Big Data at a Startup

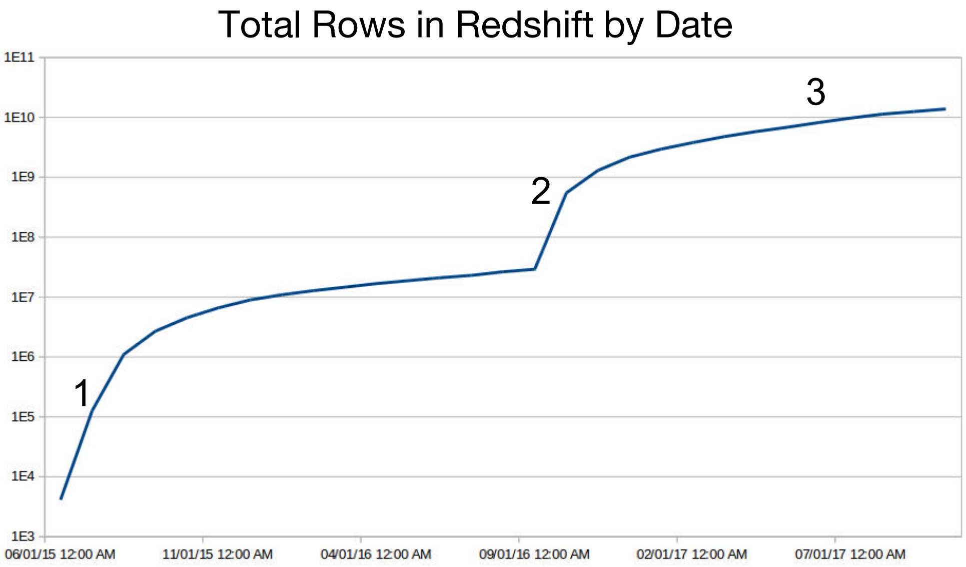
1. Funny Joke!

2. Until it's not…



The Hitchhiker's Guide to Redshift—part 1: With great power come performance issues [1]

1. Then it's expensive



[1] https://blog.atomdata.io/the-hitchhikers-guide-to-redshift-part-1-with-great-power-comes-performance-issues-748d293e0b18

Redshift: Managing Data Volume

“Easy” Stuff Already Done:

- Added nodes (a lot)
- Encoded all columns
- Vacuumed nightly
- Eliminated experiments/old tables

Next up**: Data Lifecycle Management** 💀, classically means:

- New Systems
- More Complexity
- Fragmented Data

But does it have to be painful in 2017?

[1] https://blog.atomdata.io/the-hitchhikers-guide-to-aws-redshift-part-2-let-s-cut-costs-cfdf2d67293b

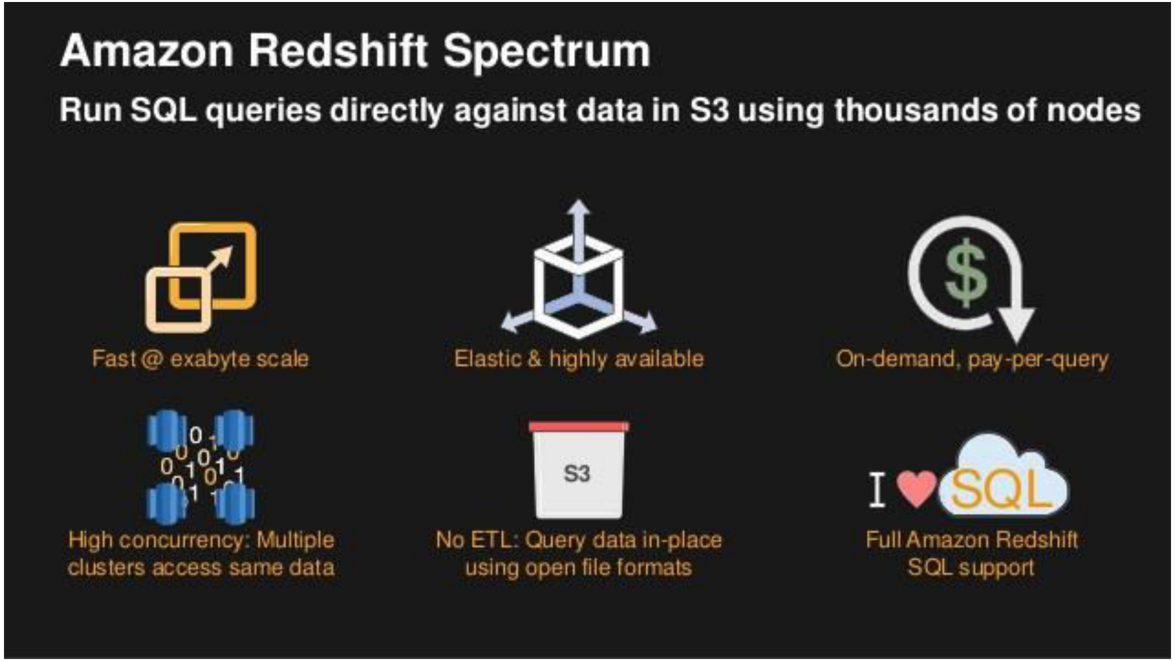
Querying Cold Data

Solution	Pay-per-query	Quick Setup	SQL	Join to Redshift
Non-hosted Solutions	No	No	Maybe	No
Amazon EMR	No	Yes	Maybe	No
2nd Redshift Cluster	No	Yes	Yes	Not really
Google BigQuery	Yes	Yes	Yes	No
Amazon Athena	Yes	Yes	Yes	No
Redshift Spectrum	Yes	Yes	Yes	Yes

Spectrum allows us to use our current BI/reporting queries with almost no changes

How We Took Control of Costs

What is Spectrum?



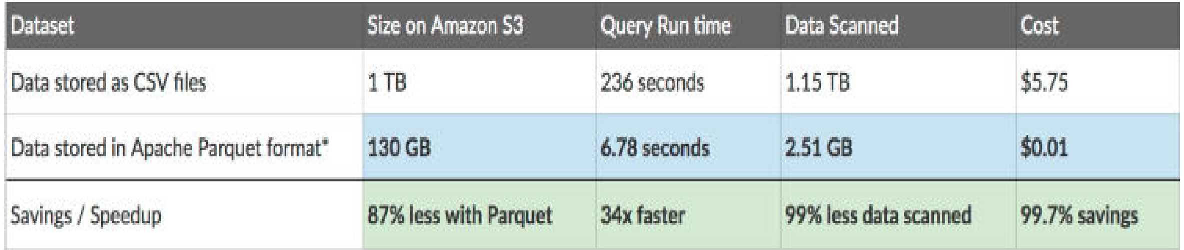
Amazon Redshift Spectrum

Run SQL queries directly against data in S3 using thousands of nodes

- Fast @ exabyte scale
- Elastic & highly available
- On-demand, pay-per-query
- High concurrency: Multiple clusters access same data
- No ETL: Query data in-place using open file formats
- Full Amazon Redshift SQL support

[1] https://aws.amazon.com/redshift/spectrum/

Spectrum: CSV vs Parquet



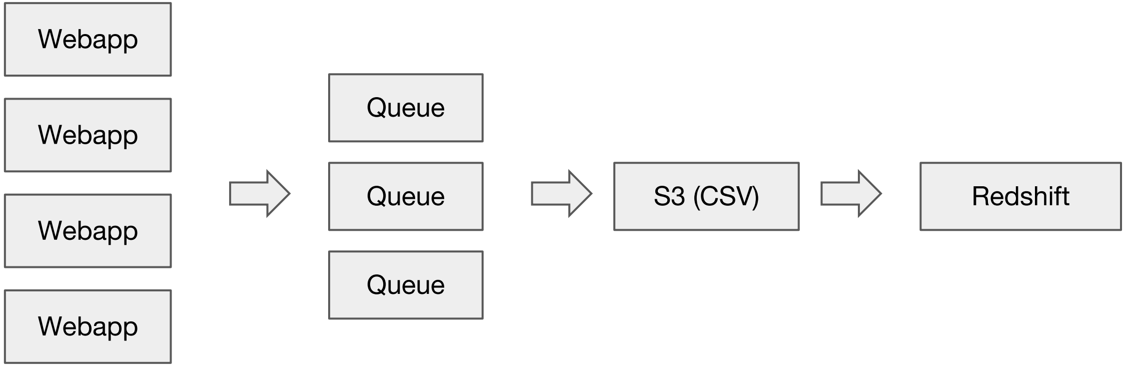
Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	130 GB	6.78 seconds	2.51 GB	$0.01
Savings / Speedup	87% less with Parquet	34x faster	99% less data scanned	99.7% savings

https://dzone.com/articles/how-to-be-a-hero-with-powerful-parquet-google-and

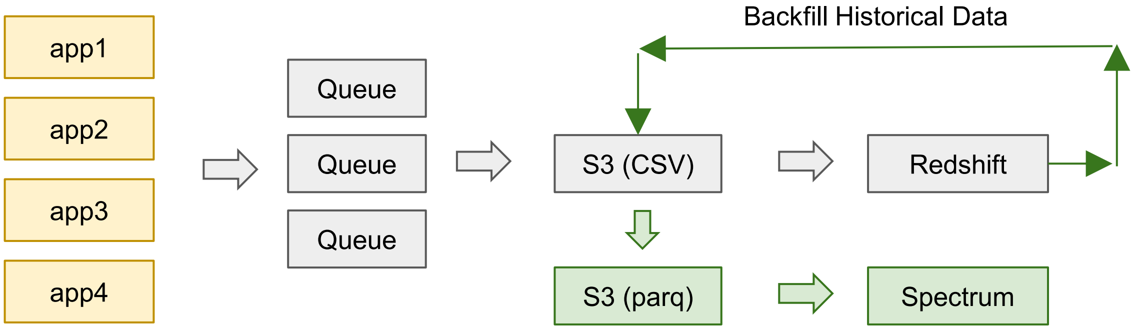
“If you are running this query once a day for a year, using uncompressed CSV files will cost $7,300. Even compressed CSV queries will cost over $1,800. However, using the Apache Parquet file format, it will cost about $460.”

A columnar storage format is imperative for optimized performance and cost

Previous Architecture



Current Architecture



Steps:

1. **Redshift → CSV**
2. **CSV → Parquet** (surprisingly tricky)
3. **Parquet → Spectrum table**
4. Remove data from Redshift as necessary

Other Stuff:

- Microservice transition: 50%
- Next up: Message Bus

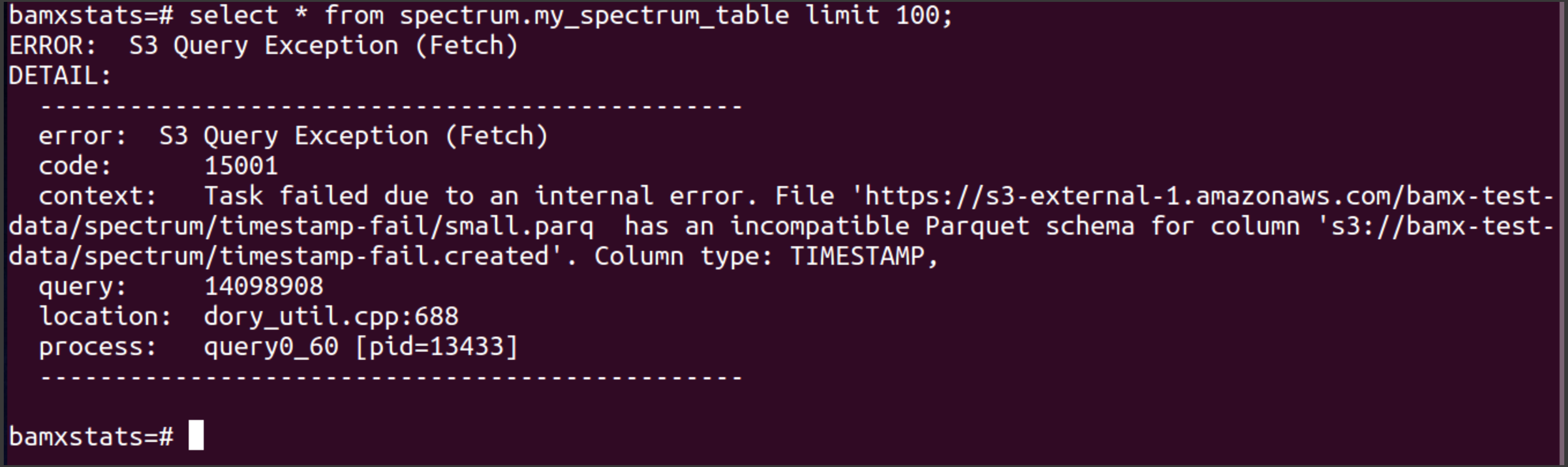
Challenge #1
CSV → Parquet

CSV to Parquet: What Are Your Options?

- AWS How-to
  - EMR + Spark
  - Heavy solution for converting some files? “Use 2 r3.8xlarge nodes”
  - AWS Big Data Blog
- FastParquet
  - Python
  - Uses Pandas; does not handle nullable integer columns
- Apache Arrow ✔

  After a day or two, have a working CSV --> Parquet converter using Arrow

  - Python/C++
  - Uses reference Parquet implementation, parquet-cpp



```
bamxstats=# select * from spectrum.my_spectrum_table limit 100;
ERROR:  S3 Query Exception (Fetch)
DETAIL:
  -----------------------------------------------
  error:  S3 Query Exception (Fetch)
  code:      15001
  context:   Task failed due to an internal error. File 'https://s3-external-1.amazonaws.com/bamx-test-
data/spectrum/timestamp-fail/small.parq  has an incompatible Parquet schema for column 's3://bamx-test-
data/spectrum/timestamp-fail.created'. Column type: TIMESTAMP,
  query:     14098908
  location:  dory_util.cpp:688
  process:   query0_60 [pid=13433]
  -----------------------------------------------

bamxstats=# 
```

Uh Oh

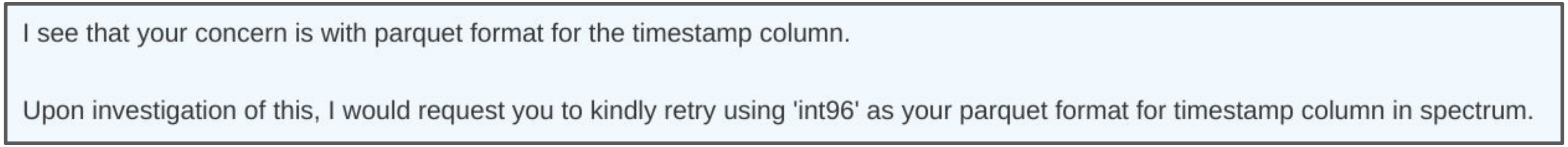
About Parquet Datatypes

- 2 types per column: logical and physical
- Logical: meaning
- Physical: layout

col_name	logical_type	physical_type
datetime_created	timestamp (usec)	int64

CSV to Parquet: Timestamps

AWS Support



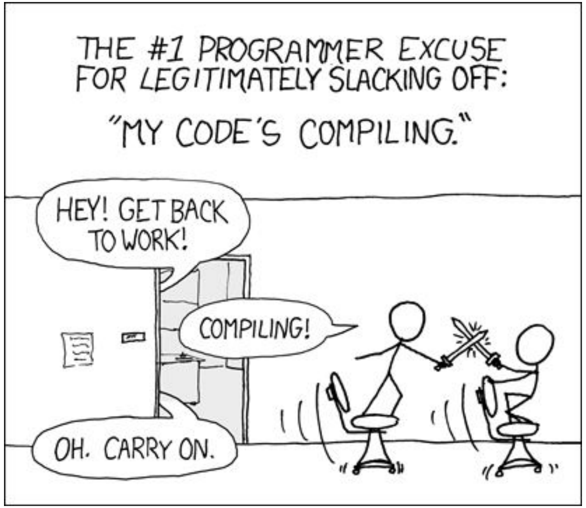
I see that your concern is with parquet format for the timestamp column.

Upon investigation of this, I would request you to kindly retry using 'int96' as your parquet format for timestamp column in spectrum.

- Int96: unofficial physical type for timestamp
- Not part of parquet standard, but used by many popular projects - spark, impala, etc.
- Spectrum officially deprecated int96 in favor of int64; int64 support still in progress upstream
- **parquet-cpp and arrow: can read but not write int96**

CSV to Parquet: Challenge Accepted

- PR’s to arrow and parquet-cpp
- Great experience with both communities
- Working release within a week or two



https://xkcd.com/303/

Challenge #2
Schema Management

Managing Schemas

All of these must be compatible and managed over time	**Challenges**
Redshift	Modifications alter historical data
CSV	Does not have schema info embedded Ordering must be preserved
Parquet	Heterogenous over time
Spectrum	Validated at query time; must fit heterogenous parquet data

Possible solution: end-to-end versioning

You Can Control Costs Too

(with less pain)

Spectrum: Takeaways

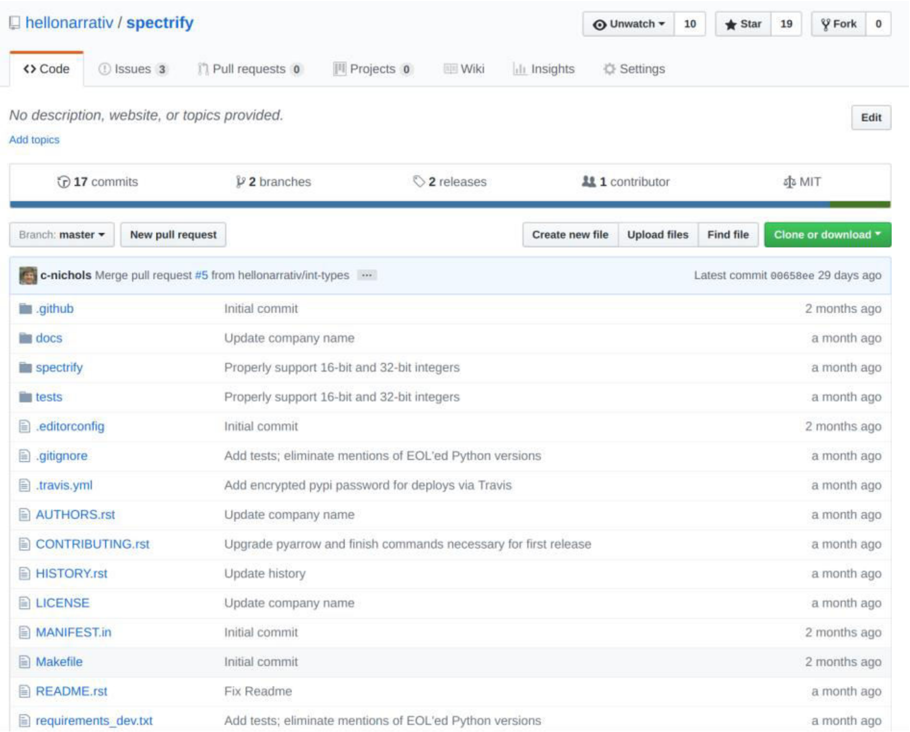
- Use Parquet (or another columnar format)
- Be careful about data types
- Have a plan for schema changes at each stage
- Use partitioning
- Use small files (no longer necessary?)

Today: The coldest 20% of our data is in Spectrum
We are now in control of our Redshift costs!

Spectrify

- **Easy Mode:**
  - Export to CSV
  - S3 CSV → S3 Parquet
  - Create external table
  - Redshift table schema as single-source-of-truth

- **Key Enabling Tech:**
  - Pyarrow
    - parquet conversion



No description, website, or topics provided.

c-nichols Merge pull request #5 from hellonarrativ/int-types … Latest commit 00658ee 29 days ago

.github	Initial commit	2 months ago
docs	Update company name	a month ago
spectrify	Properly support 16-bit and 32-bit integers	a month ago
tests	Properly support 16-bit and 32-bit integers	a month ago
.editorconfig	Initial commit	2 months ago
.gitignore	Add tests; eliminate mentions of EOL'ed Python versions	a month ago
.travis.yml	Add encrypted pypi password for deploys via Travis	a month ago
AUTHORS.rst	Update company name	a month ago
CONTRIBUTING.rst	Upgrade pyarrow and finish commands necessary for first release	a month ago
HISTORY.rst	Update history	a month ago
LICENSE	Update company name	a month ago
MANIFEST.in	Initial commit	2 months ago
Makefile	Initial commit	2 months ago
README.rst	Fix Readme	a month ago
requirements_dev.txt	Add tests; eliminate mentions of EOL'ed Python versions	a month ago

https://github.com/hellonarrativ/spectrify

Directions for Future Work

- Easy Partitioning
- Schema Versioning
- BigQuery or Athena Support

Questions?

Thank You