

Bulletproof Jobs: Patterns for Large Scale Processing

Sim Simeonov

Founder & CTO, Swoop

@simeons



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Spark Magic @ Swoop

5-10x faster exploration & ML

8x more reliable job execution

10-100x faster root cause analysis



SPARK SUMMIT 2016

swcop There it is.



SPARK SUMMIT 2016

looker

 Redshift

Spark 

 databricks™

 S3



 mongoDB®



SPARK SUMMIT 2016

Data Aggregation Example

ad_id	views	clicks
1234567890	1000000	10000



account_name	campaign_name	views	clicks	ctr_pct
ABC Inc.	Widgets	10,000,000	100,000	1.00



Get Magic

Email spark@swoop.com

Connect @simeons



SPARK SUMMIT 2016

```
sqlContext.table("ads")  
  .join(sqlContext.table("accounts"),  
        expr("get_account_id(ad_id)") === 'account_id')  
  .join(sqlContext.table("accounts"),  
        expr("get_campaign_id(ad_id)") === 'campaign_id')  
  .groupBy("account_name", "campaign_name")  
  .agg(sum('views').as("views"), sum('clicks').as("clicks"))  
  .withColumn("ctr_pct",  
              format_number(expr("100*clicks/views"), 2))  
  .orderBy('account_name', 'campaign_name')
```



Breaking Down the Code

- Red what?
- Green for whom?
- Blue how?



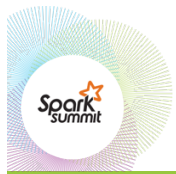
SPARK SUMMIT 2016

```
sqlContext.table("ads")  
  .join(sqlContext.table("accounts"),  
        expr("get_account_id(ad_id)") === 'account_id')  
  .join(sqlContext.table("accounts"),  
        expr("get_campaign_id(ad_id)") === 'campaign_id')  
  .groupBy("account_name", "campaign_name")  
  .agg(sum('views').as("views"), sum('clicks').as("clicks"))  
  .withColumn("ctr_pct",  
              format_number(expr("100*clicks/views"), 2))  
  .orderBy('account_name', 'campaign_name')
```



Mixing Explicit Dependencies

```
.withColumn("ctr_pct",  
            format_number(expr("100*clicks/views"), 2))
```



SPARK SUMMIT 2016



Machine Learning: The High-Interest Credit Card of Technical Debt

<http://research.google.com/pubs/pub43146.html>



SPARK SUMMIT 2016

Code Tied to Input Data

```
sqlContext.table("keywords")  
.join(sqlContext.table("accounts"),  
      expr("get_account_id(keyword_id)") === 'account_id')  
.join(sqlContext.table("accounts"),  
      expr("get_campaign_id(keyword_id)") === 'campaign_id')
```



SPARK SUMMIT 2016



Magical APIs take care of the
devil in the details
so that you don't have to



SPARK SUMMIT 2016

Bulletproof Data Processing

Reliable operations
over long periods of time
and fast recovery from failures



SPARK SUMMIT 2016

Simple Real-World Example

Process the **next batch of data**,
append to existing data on S3 and
append to analytics aggregates in Redshift



SPARK SUMMIT 2016

90+% of job failures relate to I/O

Can't we just **retry** failed operations?



SPARK SUMMIT 2016

Problem

Appending is not idempotent



SPARK SUMMIT 2016

Idempotency in Spark

Spark has one idempotent way to save data

```
dataframe.write.mode(SaveMode.Overwrite).save(...  
)
```



SPARK SUMMIT 2016

Overwrite is All or Nothing

Spark has no idempotent operation to write **just the next batch** of data but we can add it ourselves



Idempotent “append” in SQL

-- a.k.a., SQL merge operation

```
begin transaction;
```

```
    delete from tbl where timestamp =  
batchPeriod;
```

```
    insert into tbl select * from staging_tbl;  
end transaction;
```

```
drop table staging_tbl;
```



Devil in the Details

- Column order dependencies
- Schema evolution
- Index management



SPARK SUMMIT 2016

Idempotent “append” with Spark

Problem: can't delete entries in Spark tables

Solution: Resilient Partitioned Tables



SPARK SUMMIT 2016

Partitioned Tables

- Put data in different directories
- Put some of the data on the file path
 - Use it to filter which files are processed

/my_tbl/year=2016/month=06/day=01/hour=00/...

/my_tbl/year=2016/month=06/day=01/hour=01/...



Idempotent “append” with Spark

```
// Overwrite relevant partition  
df.write  
  .mode(SaveMode.Overwrite)  
  .save(".../year=yyyy/month=mm/...")
```



SPARK SUMMIT 2016

Devil in the Details

- Non-atomic file operations
- Eventual consistency file systems
- Other jobs/clusters using the data
- Dynamic partition discovery issues



Resilient Partitioned Tables

- Partitioning convention
 - `par_ts` time dimension, UTC @ minute granularity
 - `par_job` identifies the operation that wrote the data
 - `par_cat` identifies the category of data
- Enable magical framework code

`/my_tbl/par_ts=201606010000/par_job=load/par_cat=catA/...`



Root Cause Analysis

While ingesting 100M pieces of data, your code throws 1,000 exceptions of 5 different types.
How will you **find & fix all** the problems?



SPARK SUMMIT 2016

Spark Records

- An envelope around your data
 - Data chain of custody
 - Row-level logging
 - Fine-grained issue handling (errors, data quality, ...)
- Enable magical framework code



Transparent Data Pattern

```
// Get the data, skip all error records  
sql("  
    select data.*  
    from my_records  
    where (features & 1) = 0  
").registerTempTable("my_data")
```



SPARK SUMMIT 2016

Create Magic with Spark!

Magic requires intent.
If you don't aim to create magic
you never will.



SPARK SUMMIT 2016

Spark Magic Checklist

- Simple
- Outcomes, not actions
- Implicit context management
- Thoughtful conventions
- Resilient, idempotent I/O
- Fast root cause analysis



Swoop's Spark Magic Toolkit

- Smart Data Warehouse
- Idempotent “next batch” operations
- Fast root cause analysis
- Spark metrics, feature engineering tools & more...



SPARK SUMMIT 2016

Get Magic

Email spark@swoop.com

Connect @simeons



SPARK SUMMIT 2016

THANK YOU.

Email spark@swoop.com to get access to our Spark extensions.

Sim Simeonov, CTO, Swoop
sim@swoop.com | [@simeons](https://twitter.com/simeons)



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO