How to spend ¾ of your yearly budget in three weeks

Nicole Carlson (she/her/hers)
nicole@parsingscience.com
@parsing_science

How to spend ¾ of your team's yearly budget in three weeks

Nicole Carlson (she/her/hers)
nicole@parsingscience.com
@parsing_science

SHOPRUNNER

Amazon Prime for everyone else: Our 6 million members get free two-day shipping, returns, and deals across our network of 140+ retailers.

Goal: Recommend Similar Products

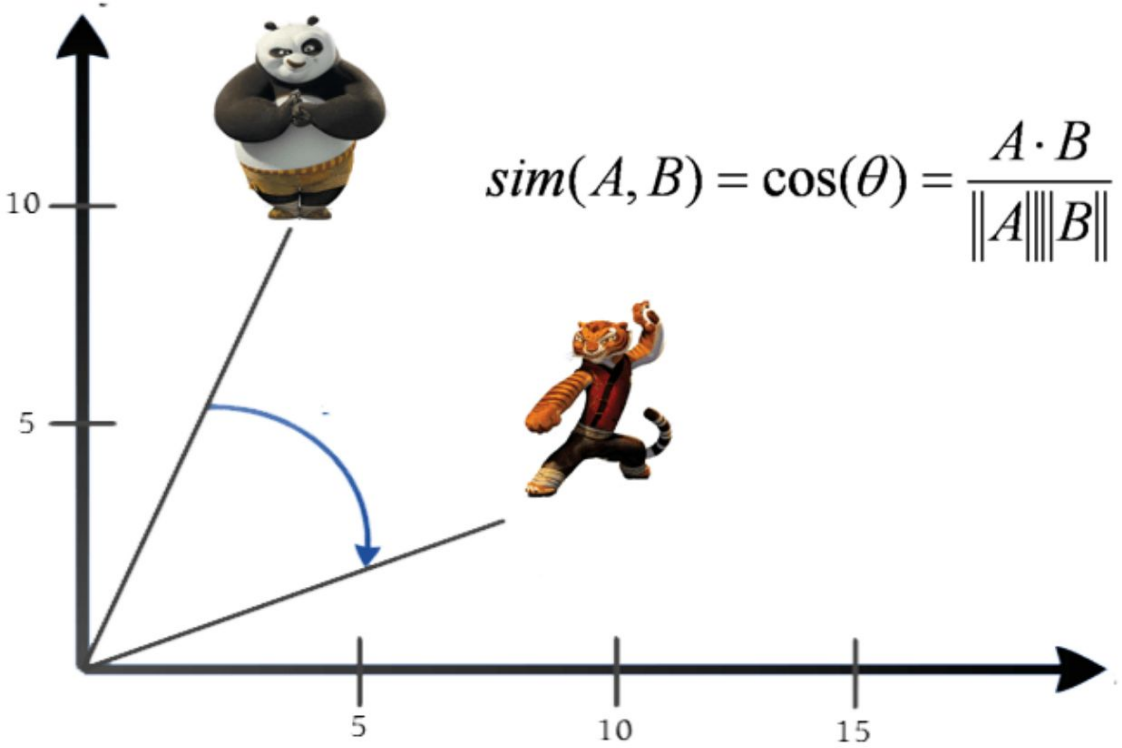
Data: Orders

Member	**Product**
Member1	ProductA
Member1	ProductB
Member2	ProductA
Member2	ProductC
Member3	ProductZ
...	...
Member_	Product_

Turn into Member/Product Interaction Matrix

	ProductA	ProductB	ProductC	...	Product_
Member1	1	1	0		0
Member2	1	0	1		0
Member3	0	0	0		0
Member4	0	0	0		1
...					
Member_	1	1	1		0

Find most similar columns (products)



$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

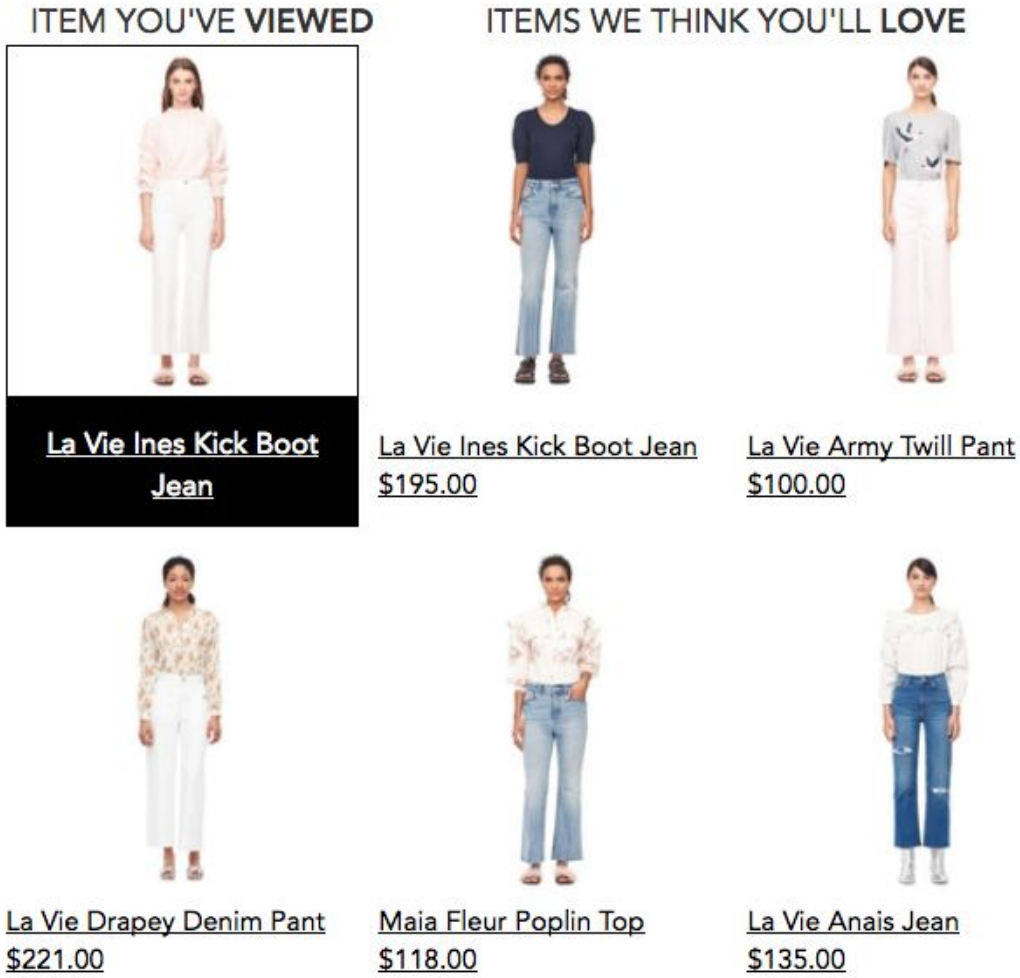
http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/

Tools

First, we found similar products by orders

- Code in python package: ✓
- Code is unittested: ✓
- Databricks cluster ready to go: ✓

Orders worked great!



Data: Member Views

Member	**Product**
Member1	ProductA
Member1	ProductB
Member1	ProductY
Member1	ProductZ
Member2	ProductZ
...	...
Member_	Product_

Now let’s find similar products with member views

- Code in python package: ✓
- Code is unittested: ✓
- Databricks cluster ready to go: ✓
- Code has been tested with orders: ✓

All I had to do was use a new datasource. Easy, right?

member_view

```
product_order_cosine_sim_job = SimilarityJob(
    job_type='product',
    similarity_method='cosine_similarity',
    datasource='orders',
)
```

member_views

NO !!!!!!!

Suddenly: Intermittent PySpark Error

---------- Py4JJavaError Traceback (most recent call last) <command-47720> in <module>() ----> 1 final_sims.select('SKU_1').distinct().count() /databricks/spark/python/pyspark/sql/dataframe.py in count(self) 428 2 429 """ --> 430 return int(self._jdf.count()) 431 432 @ignore_unicode_prefix /databricks/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py in __call__(self, *args) 1131 answer = self.gateway_client.send_command(command) 1132 return_value = get_return_value(-> 1133 answer, self.gateway_client, self.target_id, self.name) 1134 1135 for temp_arg in temp_args: /databricks/spark/python/pyspark/sql/utils.py in deco(*a, **kw) 61 def deco(*a, **kw): 62 try: ---> 63 return f(*a, **kw) 64 except py4j.protocol.Py4JJavaError as e: 65 s = e.java_exception.toString() /databricks/spark/python/lib/py4j-0.10.4-src.zip/py4j/protocol.py in get_return_value(answer, gateway_client, target_id, name) 317 raise Py4JJavaError(318 "An error occurred while calling {0}{1}{2}.\n". --> 319 format(target_id, ".", name), value) 320 else: 321 raise Py4JError(Py4JJavaError: An error occurred while calling o1601.count. : org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 1048.0 failed 4 times, most recent failure: Lost task 0.3 in stage 1048.0 (TID 88917, 10.132.77.36, executor 18): com.amazonaws.services.s3.model.AmazonS3Exception: **The provided token has expired. (Service: Amazon S3; Status Code: 400; Error Code: ExpiredToken; Request ID: 24E7DCB4454C7551),** S3 Extended Request ID: jGGTBbEB9BPRv/FIQDPh0TtYMjhYGYz94UHmWSLpM/OlE6A4Apdc+Ab/inh7vS4Cn6o5jwdUd8Q= at

```
com.amazonaws.http.AmazonHttpClient$RequestExecutor.handleErrorResponse(AmazonHttpClient.java:1588) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1258) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeHelper(AmazonHttpClient.java:1030) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.doExecute(AmazonHttpClient.java:742) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeWithTimer(AmazonHttpClient.java:716) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.execute(AmazonHttpClient.java:699) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.access$500(AmazonHttpClient.java:667) at
com.amazonaws.http.AmazonHttpClient$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649) at
com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513) at
com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4169) at
com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4116) at
com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1365) at
com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1243) at
net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:113) at
net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:89) at
scala.collection.immutable.List.foreach(List.scala:381) at
net.snowflake.spark.snowflake.SnowflakeRDD.compute(SnowflakeRDD.scala:89) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
```

org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:96) at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:53) at org.apache.spark.scheduler.Task.run(Task.scala:110) at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:349) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748) Driver stacktrace: at org.apache.spark.scheduler.DAGScheduler.org$apache$spark$scheduler$DAGScheduler$$failJobAndIndependentStages(DAGScheduler.scala:1678) at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortStage$1.apply(DAGScheduler.scala:1666) at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortStage$1.apply(DAGScheduler.scala:1665) at scala.collection.mutable.ResizableArray$class.foreach(ResizableArray.scala:59) at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:48) at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:1665) at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTaskSetFailed$1.apply(DAGScheduler.scala:931) at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTaskSetFailed$1.apply(DAGScheduler.scala:931) at scala.Option.foreach(Option.scala:257) at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:931) at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:1898) at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:1849) at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:1837) at

```
org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:48) at
org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:733) at
org.apache.spark.SparkContext.runJob(SparkContext.scala:2114) at
org.apache.spark.SparkContext.runJob(SparkContext.scala:2211) at
org.apache.spark.sql.execution.collect.Collector.runSparkJobs(Collector.scala:206) at
org.apache.spark.sql.execution.collect.Collector.collect(Collector.scala:241) at
org.apache.spark.sql.execution.collect.Collector$.collect(Collector.scala:64) at
org.apache.spark.sql.execution.collect.Collector$.collect(Collector.scala:70) at
org.apache.spark.sql.execution.SparkPlan.executeCollectResult(SparkPlan.scala:264) at
org.apache.spark.sql.execution.SparkPlan.executeCollect(SparkPlan.scala:255) at
org.apache.spark.sql.Dataset$$anonfun$count$1.apply(Dataset.scala:2519) at
org.apache.spark.sql.Dataset$$anonfun$count$1.apply(Dataset.scala:2518) at
org.apache.spark.sql.Dataset$$anonfun$59.apply(Dataset.scala:3021) at
org.apache.spark.sql.execution.SQLExecution$.withCustomExecutionEnv(SQLExecution.scala:89) at
org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:127) at
org.apache.spark.sql.Dataset.withAction(Dataset.scala:3020) at org.apache.spark.sql.Dataset.count(Dataset.scala:2518) at
sun.reflect.GeneratedMethodAccessor426.invoke(Unknown Source) at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at
java.lang.reflect.Method.invoke(Method.java:498) at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244) at
py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:380) at py4j.Gateway.invoke(Gateway.java:293) at
py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132) at
```

py4j.commands.CallCommand.execute(CallCommand.java:79) at py4j.GatewayConnection.run(GatewayConnection.java:226) at java.lang.Thread.run(Thread.java:748) Caused by: com.amazonaws.services.s3.model.AmazonS3Exception: The provided token has expired. (Service: Amazon S3; Status Code: 400; Error Code: ExpiredToken; Request ID: 24E7DCB4454C7551), S3 Extended Request ID: jGGTBbEB9BPRv/FIQDPh0TtYMjhYGYz94UHmWSLpM/OlE6A4Apdc+Ab/inh7vS4Cn6o5jwdUd8Q= at com.amazonaws.http.AmazonHttpClient$RequestExecutor.handleErrorResponse(AmazonHttpClient.java:1588) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1258) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeHelper(AmazonHttpClient.java:1030) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.doExecute(AmazonHttpClient.java:742) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeWithTimer(AmazonHttpClient.java:716) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.execute(AmazonHttpClient.java:699) at com.amazonaws.http.AmazonHttpClient$RequestExecutor.access$500(AmazonHttpClient.java:667) at com.amazonaws.http.AmazonHttpClient$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649) at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513) at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4169) at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4116) at com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1365) at com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1243) at net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:113) at net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:89) at scala.collection.immutable.List.foreach(List.scala:381) at

```
net.snowflake.spark.snowflake.SnowflakeRDD.compute(SnowflakeRDD.scala:89) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:96) at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:53) at
org.apache.spark.scheduler.Task.run(Task.scala:110) at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:349) at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) ... 1 more
```

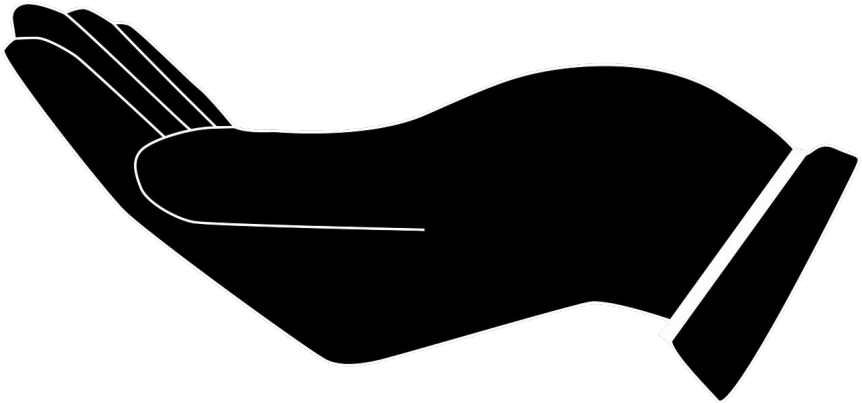
Possibly Relevant Part of Stacktrace

“The provided token has expired. (Service: Amazon S3; Status Code: 400; Error Code: ExpiredToken”

My ideas to speed things up and avoid the s3 timeout error:

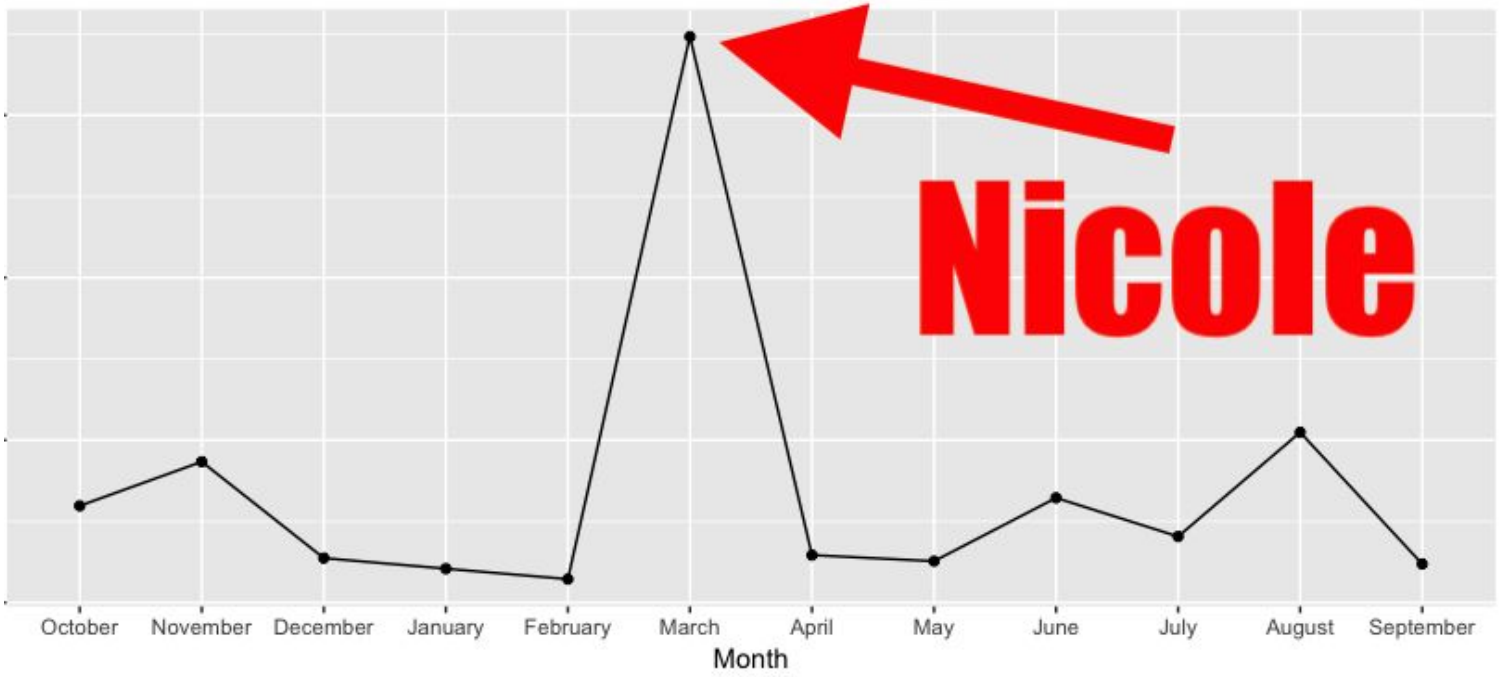
- Just run it again
- Add more nodes
- Increase individual node size
- All of these at the same time?

So I asked my boss: How big can my clusters be?

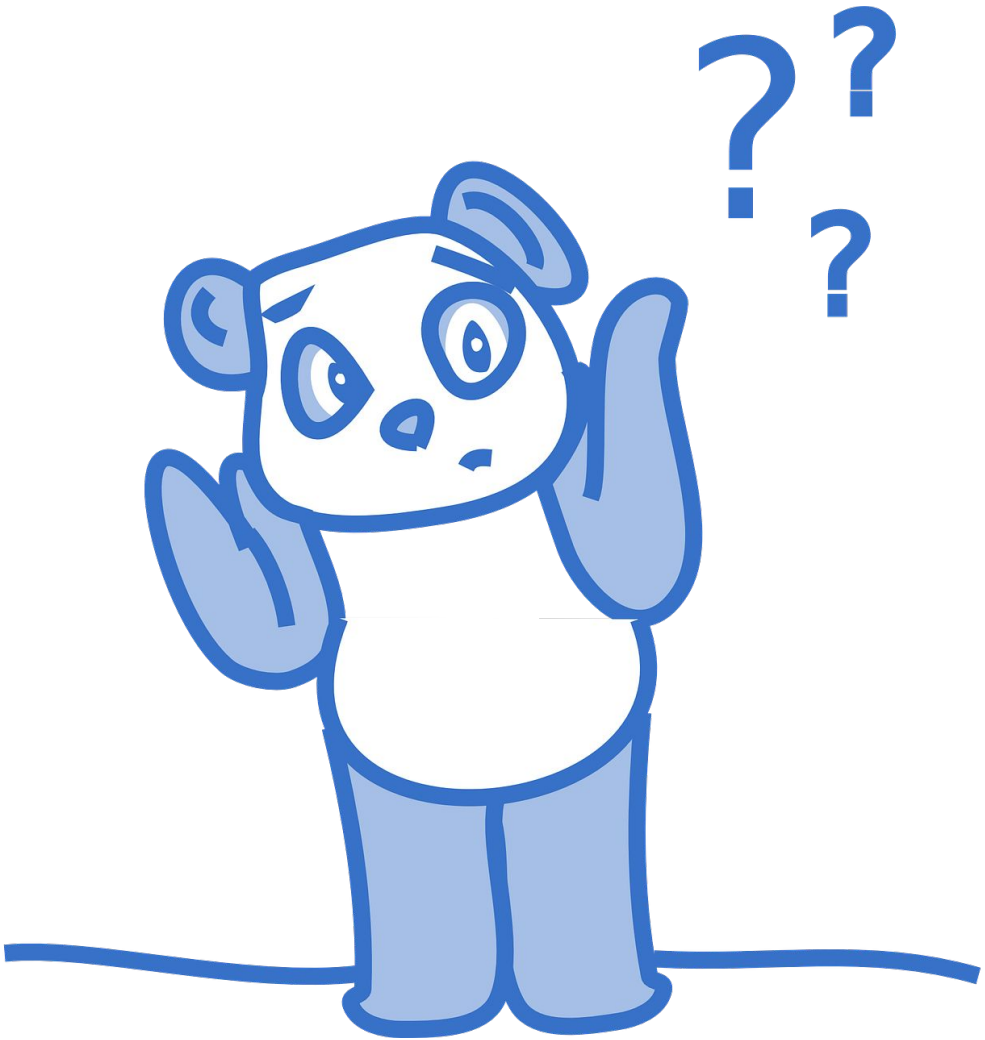


Answer: As big as you want!

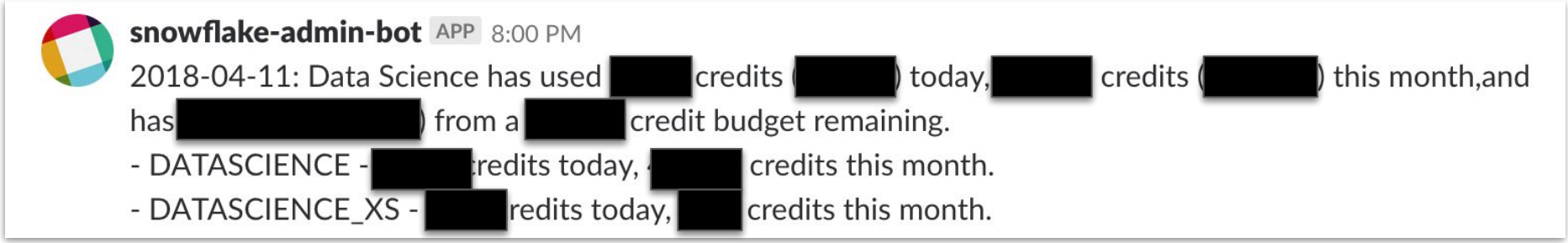
Three weeks later...



What should we have done?



1. Check on Cluster and Database Usage



2. Test with a smaller dataset

Member	**Product**
Member1	ProductA
Member1	ProductB
Member1	ProductY
~~Member1~~	~~ProductZ~~
~~Member2~~	~~ProductZ~~
~~...~~	~~...~~
~~Member_~~	~~Product_~~

3. Partition data appropriately

```
num_partitions = int(
  (num_rows * mb_per_row) / max_mb_per_partition
)
```

Suggested max_mb_per_partition: 256

Good python code can be hard to debug in PySpark

```
sim_job = SimilarityJob(
    job_type='product',
    similarity_method='cosine_similarity',
    datasource='member_views',
)

sims_dict = sim_job.generate_recs(spark)

sim_job.write_recs_to_dbs(sims_dict)
```

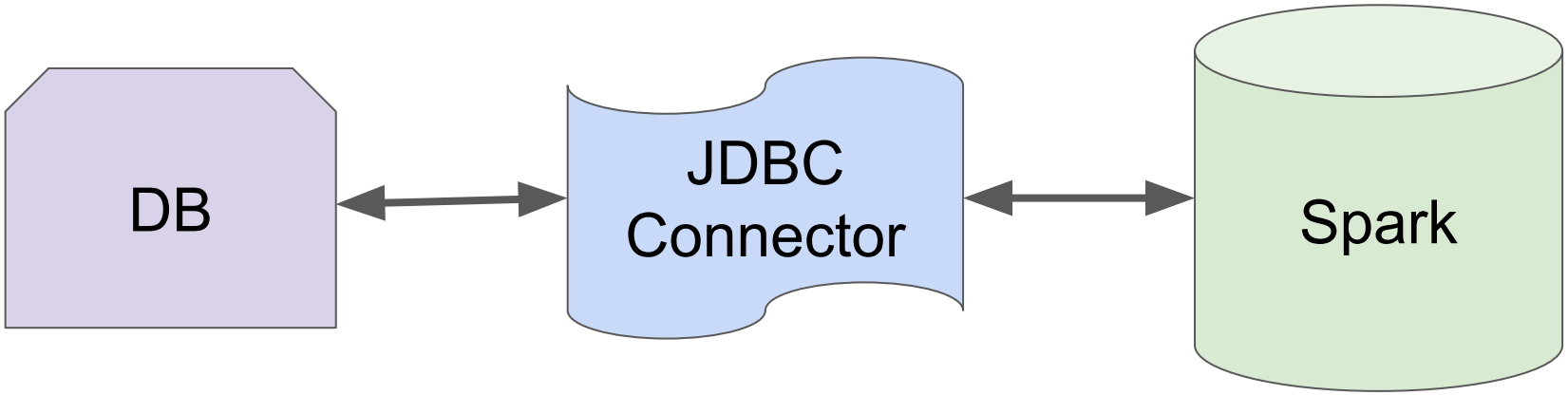
4. Run each step separately

```
output = (
    CoordinateMatrix(
        df
        .rdd
        .map(tuple)
    )
    .toRowMatrix()
    .columnSimilarities()
)
```

```
matrix =  CoordinateMatrix(
        df
        .rdd
        .map(tuple)
)

matrix.entries.take(1)

row_matrix = matrix.toRowMatrix()

row_matrix.rows.take(1)

sims = row_matrix.columnSimilarities()
```

5. Save intermediate steps

Let’s talk about JDBC Connectors



6. Remove JDBC Connector by saving data to S3

```
orders = load_from_snowflake(spark, sf_options, sql)
write_to_s3(orders, 'raw_orders', full_s3_path)
read_in_orders = read_from_s3(spark, 'raw_orders', full_s3_path)
```

So what was causing the S3 expired token error?

Secret buried in the Stacktrace

com.amazonaws.http.AmazonHttpClient$RequestExecutor.handleErrorResponse(AmazonHttpClient.java:1588) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1258) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeHelper(AmazonHttpClient.java:1030) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.doExecute(AmazonHttpClient.java:742) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.executeWithTimer(AmazonHttpClient.java:716) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.execute(AmazonHttpClient.java:699) at
com.amazonaws.http.AmazonHttpClient$RequestExecutor.access$500(AmazonHttpClient.java:667) at
com.amazonaws.http.AmazonHttpClient$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649) at
com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513) at
com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4169) at
com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4116) at
com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1365) at
com.amazonaws.services.s3.AmazonS3Client.getObject(AmazonS3Client.java:1243) at
net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:113) at
net.snowflake.spark.snowflake.SnowflakeRDD$$anonfun$compute$1.apply(SnowflakeRDD.scala:89) at
scala.collection.immutable.List.foreach(List.scala:381) at
net.snowflake.spark.snowflake.SnowflakeRDD.compute(SnowflakeRDD.scala:89) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:332) at org.apache.spark.rdd.RDD.iterator(RDD.scala:296) at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38) at

Spark caching: data from S3

```
1  read_in_orders = read_from_s3(spark, 'raw_orders', full_s3_path)
```

▸ (1) Spark Jobs

▸ read_in_orders: pyspark.sql.dataframe.DataFrame = [member_id: string, order_date: date ... 5 more fields]

Command took 1.41 seconds -- by ncarlson@shoprunner.com at 9/23/2018, 11:03:34 AM on ncarlson_pydata

Cmd 13

```
1  read_in_orders.cache()
```

```
Out[10]: DataFrame[member_id: string, order_date: date, order_id: decimal(
38,0), partner_code: string, doc_id: string, name: string, brand_name: str
ing]
```

Command took 0.01 seconds -- by ncarlson@shoprunner.com at 9/23/2018, 11:03:34 AM on ncarlson_pydata

Spark caching: data from Snowflake JDBC Connector

```
1  orders = load_from_snowflake(spark, sf_options, sql)
```

orders: pyspark.sql.dataframe.DataFrame = [member_id: string, order_date: date ... 5 more fields]

Command took 1.51 seconds -- by ncarlson@shoprunner.com at 9/23/2018, 11:03:31 AM on ncarlson_pydata

Cmd 8

```
1  orders.cache()
```

```
Out[5]: DataFrame[member_id: string, order_date: date, order_id: decimal(3
8,0), partner_code: string, doc_id: string, name: string, brand_name: stri
ng]
```

Command took 45.60 seconds -- by ncarlson@shoprunner.com at 9/23/2018, 11:03:32 AM on ncarlson_pydata

Underlying Cause: Secret S3 Temp Storage Timeout

7. Ask your vendors if they have any unexposed settings

(e.g. timeouts)

Recap

1. Regularly check on cluster and database usage
2. Test with a smaller dataset
3. Partition data correctly
4. Run each PySpark calculation step separately
5. Save intermediate steps to S3
6. Write any JDBC Connector results immediately to S3
7. Check unexposed third party settings (e.g. secret vendor timeouts)

Epilogue: Recouping costs

Thanks!

- ShopRunner
- Data Science Team: Michelangelo, Hanna, Ali, Rishi, Scott
- Tudor Radoaca

nicole@parsingscience.com
@parsing_science