Spark Application Carousel

Spark Summit East 2015

databricks™

About Today's Talk

- **About Me:**
  - **Vida Ha** - Solutions Engineer at Databricks.
- **Goal:**
  - For beginning/early intermediate Spark Developers.
  - Motivate you to start writing more apps in Spark.
  - Share some tips I've learned along the way.

Today’s Applications Covered

- **Web Logs Analysis**
  - Basic Data Pipeline
- **Wikipedia Dataset**
  - Machine Learning
- **Facebook API**
  - Graph Algorithms

Application 1: Web Log Analysis

Web Logs

- **Why?**
  - Most organizations have web log data.
  - Dataset is too expensive to store in a database.
  - Awesome, easy way to learn Spark!
- **What?**
  - Standard Apache Access Logs.
  - Web logs flow in each day from a web server.

Reading in Log Files

```
access_logs = (sc.textFile(DBFS_SAMPLE_LOGS_FOLDER)
                # Call the parse_apace_log_line on each line.
                .map(parse_apache_log_line)
                # Caches the objects in memory.
               .cache())
# Call an action on the RDD to actually populate the
cache.
access_logs.count()
```

databricks

Calculate Context Size Stats

```
content_sizes = (access_logs
             .map(lambda row: row.contentSize)
             .cache()) # Cache since multiple queries.
average_content_size = content_sizes.reduce(
          lambda x, y: x + y) / content_sizes.count()
min_content_size = content_sizes.min()
max_content_size = content_sizes.max()
```

databricks

Frequent IPAddresses - Key/Value Pairs

```
ip_addresses_rdd = (access_logs
                .map(lambda log: (log.ipAddress, 1))
                .reduceByKey(lambda x, y : x + y)
                .filter(lambda s: s[1] > n)
                .map(lambda s: Row(ip_address = s[0],
                                   count = s[1])))
                # Alternately, could just collect() the values.
                .registerTempTable(“ip_addresses”)
```

databricks

Other Statistics to Compute

- Response Code Count.
- Top Endpoints & Distribution.
- …and more.

Great way to learn various Spark Transformations and actions and how to chain them together.

*** BUT Spark SQL makes this much easier!**

Better: Register Logs as a Spark SQL Table

```
sqlContext.sql(“CREATE EXTERNAL TABLE access_logs
   ( ipaddress STRING … contextSize INT … )
   ROW FORMAT
   SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES (
    "input.regex" = '^(\\S+) (\\S+) (\\S+) …’)
  LOCATION \”/tmp/sample_logs\”
”)
```

databricks

Context Sizes with Spark SQL

```
sqlContext.sql(“SELECT
    (SUM(contentsize) / COUNT(*)),  # Average
    MIN(contentsize),
    MAX(contentsize)
FROM access_logs”)
```

Frequent IPAddress with Spark SQL

```
sqlContext.sql("SELECT
    ipaddress,
    COUNT(*) AS total
  FROM access_logs
  GROUP BY ipaddress
  HAVING total > N")
```

databricks

Tip: Use Partitioning

- Only analyze files from days you care about.

```
sqlContext.sql(“ALTER TABLE access_logs
  ADD PARTITION (date='20150318')
  LOCATION ‘/logs/2015/3/18’”)
```

- If your data rolls between days - perhaps those few missed logs don’t matter.

Tip: Define Last N Day Tables for caching

- Create another table with a similar format.
- Only register partitions for the last N days.
- Each night:
  - Uncache the table.
  - Update the partition definitions.
  - Recache:

```
sqlContext.sql("CACHE access_logs_last_7_days")
```

databricks

Tip: Monitor the Pipeline with Spark SQL

- Detect if your batch jobs are taking too long.
- Programmatically create a temp table with stats from one run.

```
sqlContext.sql("CREATE TABLE IF NOT EXISTS
pipelineStats (runStart INT, runDuration INT)")

sqlContext.sql("insert into TABLE pipelineStats select
runStart, runDuration from oneRun limit 1")
```

- **Coalesce** the table from time to time.

databricks

Demo: Web Log Analysis

Application: Wikipedia

Tip: Use Spark to Parallelize Downloading

Wikipedia can be downloaded in one giant file, or you can download the 27 parts.

```
val articlesRDD = sc.parallelize(articlesToRetrieve.toList, 4)
val retrieveInPartitions = (iter: Iterator[String]) => {
  iter.map(article => retrieveArticleAndWriteToS3(article)) }
val fetchedArticles =
  articlesRDD.mapPartitions(retrieveInPartitions).collect()
```

Processing XML data

- Excessively large (> 1GB) compressed XML data is hard to process.
  - Not easily splittable.
  - Solution: Break into text files where there is one XML element per line.

ETL-ing your data with Spark

- Use an XML Parser to pull out fields of interest in the XML document.
- Save the data in Parquet File format for faster querying.
- Register the Parquet format files as Spark SQL since there is a clearly defined schema.

Using Spark for Fast Data Exploration

- CACHE the dataset for faster querying.
- Interactive programming experience.
- Use a mix of Python or Scala combined with SQL to analyze the dataset.

Tip: Use MLLib to Learn from Dataset

- Wikipedia articles are an rich set of data for the English language.
- Word2Vec is a simple algorithm to learn synonyms and can be applied to the wikipedia article.
- Try out your favorite ML/NLP algorithms!

databricks

Demo: Wikipedia App

Application: Facebook API

Tip: Use Spark to Scrape Facebook Data

- Use Spark to Facebook to make requests for friends of friends in parallel.
- NOTE: Latest Facebook API will only show friends that have also enabled the app.
- If you build a Facebook App and get more users to accept it, you can build a more complete picture of the social graph!

Tip: Use GraphX to learn on Data

- Use the Page Rank algorithm to determine who's the most popular**.
- Output User Data: Facebook User Id to name.
- Output Edges: User Id to User Id

** In this case it's my friends, so I'm clearly the most popular.

Demo: Facebook API

Conclusion

- I hope this talk has inspired you to want to write Spark applications on your favorite dataset.
- Hacking (and making mistakes) is the best way to learn.
- If you want to walk through some examples, see the Databricks Spark Reference Applications:
  - https://github.com/databricks/reference-apps

THE END

databricks™