Apache Spark Dynamic Clusters

MonBUG October 2016

Dunarel Badescu

Research Associate in Bioinformatics | Ragoussis Lab
McGill University and Genome Quebec Innovation Centre

Apache Spark

- Complete Framework for Cluster Computing
- Map/Reduce
- Apache Hive SQL implementation
- In memory partitioned datasets
  - Worst case scenario – Apache Hadoop
  - Best case scenario – Map/Reduce MPI
- Multiple API
  - Scala / Java / SQL / Python / R
  - In this order

Access to Spark Clusters

- Local install for testing on one node

  - your own laptop
- Virtual Machines with all installed software from Hadoop business providers:
  - MapR / HortonWorks / Cloudera etc...
- Standalone deployment on your own cluster
- Amazon Elastic Cloud
- Dynamic spawn on an already available (almost free) cluster
  - Compute Canada
  - Institutional (MUGQIC abacus)

Standalone Cluster

- Needs a Master and one or more slaves
- Hive Thrift Server needs a Hive Metastore
  - Apache Derby
- Optional use of Hadoop native libraries for faster data access
- Snappy compression library is hard to get working
  - Otherwise .gzip
- No need for HDFS. When cluster ends:
  - each table is a folder containing compressed partitions
  - On regular file mount

Dynamic Configuration

- Runtime scripts are generating config files
  - Separate /config, /log, /pid folders
  - hive-site.xml, log4j.properties, spark-defaults.conf, spark-env.sh
- Setting Environment variables
- Spawn server using “nohup”
- Redirect clients to master ip location
- Ensure persistence of Derby Hive Metastore server config files
- Each cluster group is a bunch of processes writing to the same project folder, inside different subfolders each
- SSH Tunnel ports to local machine for:
  - Database access (Hive JDBC)
  - Web Jupyter server

Hive database

- Analytical Database
  - Non-transactional, Single User (The Data Scientist !!!)
- Spark allows full range of server side user defined functions
  - Hive UDFs
    - Row
    - Tables
    - Aggregates
- Java / Scala for this, no python yet
  - Suitable for parsing GTF/VCF/SAM/BED

User Access

- Command line (REPL)
- Python code (pycharm)
- Jupyter notebook
- Jdbc clients
  - Squirrel SQL to Spark Thrift Server

General Applications

- Parallel Graph Algorithms
  - Only Scala for GraphX for now
  - Very few parallel libraries for all languages
  - Virtually unexplored field
    - You are welcomed to develop
- ADAM genomic pipelines
- Range overlap problem - UCSC binning scheme
    - Hive has no indexes, binning scheme is one option, coupled with partitioning
    - Runtime transcoding parquet -> bam

Ragoussis Lab Applications

- Apache Spark is launched along our own DNA/RNA Variation/Expression pipelines
  - We have same long-standing process architecture
  - Redis server / multiple Python clients
  - Dynamic parallelism
- Single cell optimized
  - De-novo genomes annotation
    - Fusion-Genes detection
  - Cross experiments annotation comparisons and metrics
- Genomic variant / expression database

Test pyspark

- ssh mp2
- cd /nfs3_ib/ioannisr-mp2.nfs/tank/nfs/ioannisr/nobackup/share/expr/a000/pysrc/
- ./cdf.sh
- pyspark --conf spark.ui.port=4056
- textFile = sc.textFile("README.md")
- textFile.count() # Number of items in this RDD
- textFile.first() # First item in this RDD
- linesWithSpark = textFile.filter(lambda line: "Spark" in line)
- textFile.filter(lambda line: "Spark" in line).count() # How many lines contain "Spark"?

Test Hive SQL interface

- show tables

- select *
- from segs
- limit 200

- select distinct bin
- from segs