DLRL Cluster

Matt Bollinger, Joseph Pontani, Adam Lech

Client: Sunshin Lee

CS4624 Capstone Project

March 3, 2014

Virginia Tech, Blacksburg, VA

Background - Hadoop

- Open source framework for large scale data storage and processing
- Scalable, distributed computing software
- Fundamental technology invented by Google to index rich textual/structural data and present meaningful/actionable results
- Helps solve issue of data that won't fit into tables
- Any number of nodes without shared memory or disk space
  - Local computation and storage
- Hardware failure detection and handling is built-in
  - Provides for a highly-available computation and storage cluster
- Aspects:
  - Hadoop Common (libraries and utilities)
  - Hadoop Distributed File System (HDFS)
  - Hadoop YARN (resource management)
  - Hadoop MapReduce

Background – MapReduce

- Batch query processor
- Queries run on entire data set
- Suited to when data is written once, but read many times
- Designed to interpret data at processing time
  - Good for unstructured and semi-structured data

Hive

- Framework for data warehousing on top of Hadoop
  - built by Jeff Hammerbacher at Facebook
- Grew out of need to learn from the huge amount of data that Facebook was producing from burgeoning data network
- Used by many organizations

Hive – How it works

- Converts SQL queries into series of MapReduce jobs on a Hadoop cluster
- Shell is primary means of communication by issuing commands in HiveQL
- Like RDBM it organizes tables in queries
- Configured using XML configuration files
- Can use Hive Web Interface as means of communication

Hive – Clients

- The Hive Thrift Client
  - Makes it easy to run Hive commands from a wide range of programming languages (C++,Java, PHP, Python, Ruby)
- Provides Type 4 (pure Java) JDBC driver
  - Allows applications that support the ODBC protocol to connect to Hive
  - Still in development

HiveQL

- Dialect of SQL
  - Does not support SQL-92 specifications fully
- Organize tables into partitions or buckets
- Supports:
  - views, joins , subqueries
  - User defined functions that are written in Java

Hbase

- Distributed, column-oriented, database built on top of HDFS (Hadoop Distributed File System)
- Use when you require real-time read/write random-access to very large datasets
- Is *not* SQL
  - NOT relational
  - Does not support SQL

Hbase Usage

- Able to do what an RDBMS cannot: host very large, sparsely populated tables on clusters made from commodity hardware
- Canonical use
  - Web table
    - table of crawled web pages and their attributes
      - language and MIME type
      - keyed by the web page URL
    - very large with rows running into billions

Hbase – How it works

- Modelled with HBase master node orchestrating region of one or more regionslave nodes
- Ships with Avro, Thrift and Rest interfaces
- No real indexes
- Automatic partitioning of tables
- Scale linearly and automatically with addition of nodes
- Commodity hardware is cheaper ($1000-$5000 per node)
- Fault tolerant
  - Lots of nodes, meaning each is insignificant
- Batch processing
  - MapReduce integration allows for it

Mahout

- Scalable machine learning and data mining
- 3 primary methods
  - Classification
  - Clustering
  - Collaborative Filtering (Recommendation)
- Primarily implemented on top of Hadoop
  - Clustering
- Java APIs for custom solutions

Mahout – Basic Item Recommendation

Item/User History Matrix - R

	Item 1	Item 2	Item 3	Item 4
User 1	1	1	0	1
User 2	1	0	0	0
User 3	0	1	1	1

User's History - h

	User X
Item 1	0
Item 2	1
Item 3	1
Item 4	0

Use linear algebra to formulate a recommendation vector

- $R^T (R\ h)$ – recommend other users
- $(R^T R)\ h$ – recommend other items

Mahout – Basic Item Recommendation

Make an Item Matrix: $R^T R$

	Item 1	Item 2	Item 3	Item 4
Item 1	2	1	0	1
Item 2	1	2	1	2
Item 3	0	1	1	1
Item 4	1	2	1	2

$(R^T R)$ h is Recommended Items for User X

	User X
Item 1	1
Item 2	3
Item 3	2
Item 4	3

For user X, based on previous activity or existing data (from user X and from other users), items 4 and 1 are recommended, in that order.

Mahout – Item Recommendation Issues

- Linear algebra based implementation has issues
  - User cold start
    - No previous activity makes it difficult to provide recommendations
  - Disjoint data sets
    - No co-occurrence between items limits recommendations to specific item clusters
  - Outliers/minimal item data
    - If item is limited to small data set, hard to accurately recommend that item to others
  - Multiple sources of data
    - Transaction history, rankings, viewing data, etc. aren't all included in the same recommendation matrix

Impala

- A distributed, massively parallel processing database engine
- Specifically, parallel SQL queries over data stored in HDFS
  - No data movement
  - No data transformation
- Selling points
  - Run SQL queries on existing Hadoop systems simultaneously
  - Don’t have to move data around

Impala – How it works

- Impala Daemon
  - Runs on each node of your cluster
  - Accepts, distributes, and responds to queries written in HiveQL
- Impala Statestore
  - Checks health of daemons
  - Only one in cluster, informs other nodes
- Impala Catalog Service
  - Relays metadata changes to nodes
  - Reduces overhead to maintain coherency between nodes

Project – Goals and Status

- Goals
  - In depth tutorials for Mahout, Hbase + Hive, and Impala
  - Demonstration of the use-cases and capabilities of each
- Status
  - Completed tutorials by Thursday March 6, 2014
  - Demonstrations ready by the beginning of May, 2014

Project Goals

- Tutorials will be delivered as either PDF or Word Documents
  - Highly detailed, easy-to-follow guides on each of the technologies
  - Include visuals to better aid the introductory process to these technologies
- Demos will be available either on laptop on personal VM installs (VirtualBox) or on the DLRL cluster with sample data
- Designed to be used by members of the DLRL team new to these technologies