15-319 / 15-619 Cloud Computing

Recitation 10

March 24th and 26th, 2015

Overview

- **Administrative issues**
  - Tagging, 15619Project
- **Last week's reflection**
  - Project 3.3
- **This week's schedule**
  - Project 3.4
  - Unit 4 - Modules 14 & 15
  - Quiz 4
- **Demo**
- **Twitter Analytics: The 15619Project**

Caution!

- Tag spot instances in the FIRST 59 mins.
    - Otherwise, it will be considered as an untagged instance for that hour.
- 15619Project is in progress!
    - Phase 2 due on next Wednesday, 4/1
    - Fix Q1 and Q2, start on Q3 and Q4
    - Meet your TA mentor every week to get the token for the Query Reference Server.
    - Tag all resources used for 15619Project as
      Key: 15619project, Value: phase2

      Key: 15619backend, Value: hbase/mysql

Project 3.3 : FAQs - 1

Problem 1:The request received by server is not actually GET /step#?id=XX&pwd=XXX, but something like GET /step#?callback=returnRes&id=XX&pwd=XXX&_=XXX ?

- the request you need to deal with on backend server is http://server-public-dns:8080/step#?id=XX&pwd=XXX However, since JavaScript is used in the frontend and it cannot make a cross domain http request, we're using a callback function to make a request to a remote server. However, you can simply treat the request as shown in the writeup but use the response as a parameter of returnRes (that's why the response is actually "returnRes(response)").

Project 3.3 : FAQs - 2

Problem 2: Does not receive any mark on submission but my answer is correct.

- Very likely that you didn't keep the front end server (Undertow) running. Refer to "Grading" section of the writeup.

Project 3.3 : FAQs - 3

Problem 3: What is key in the constructor for GetItemRequest?

- The key should be in Map format (Map<String, AttributeValue> key)

  An example is:

```
Map<String, AttributeValue> key = new HashMap<String,
AttributeValue>();
key.put("userid", new AttributeValue().withN(id));
GetItemRequest greq = new GetItemRequest().
  withTableName(table).withKey(key);
```

This week: Project

- P3.1 Files vs Databases
- P3.2 Partitioning and Replication
- P3.3 Database-as-a-Service
- P3.4 Cloud Data Warehousing
- P3.5 Consistency in Distributed Databases

OLTP vs OLAP

- OLTP (Online Transaction Processing)
  - Deals with operational data.
  - Characterized by a large number of short online transactions (INSERT, UPDATE, DELETE). In an OLTP system data are frequently updated and queried. So a fast response to a request is required.
  - Example OLTP query: What is the Salary of Mr.John?

- OLAP (Online Analytical Processing)
  - Deals with historical data or archival data.
  - Data from OLTP are collected over a period of time and stored in a very large database called a Data Warehouse. Data warehouses are highly optimized for read(SELECT) and aggregation(SUM) operation.
  - Example OLAP query: How is the profit changing over the last three months across different regions?

OLTP vs OLAP

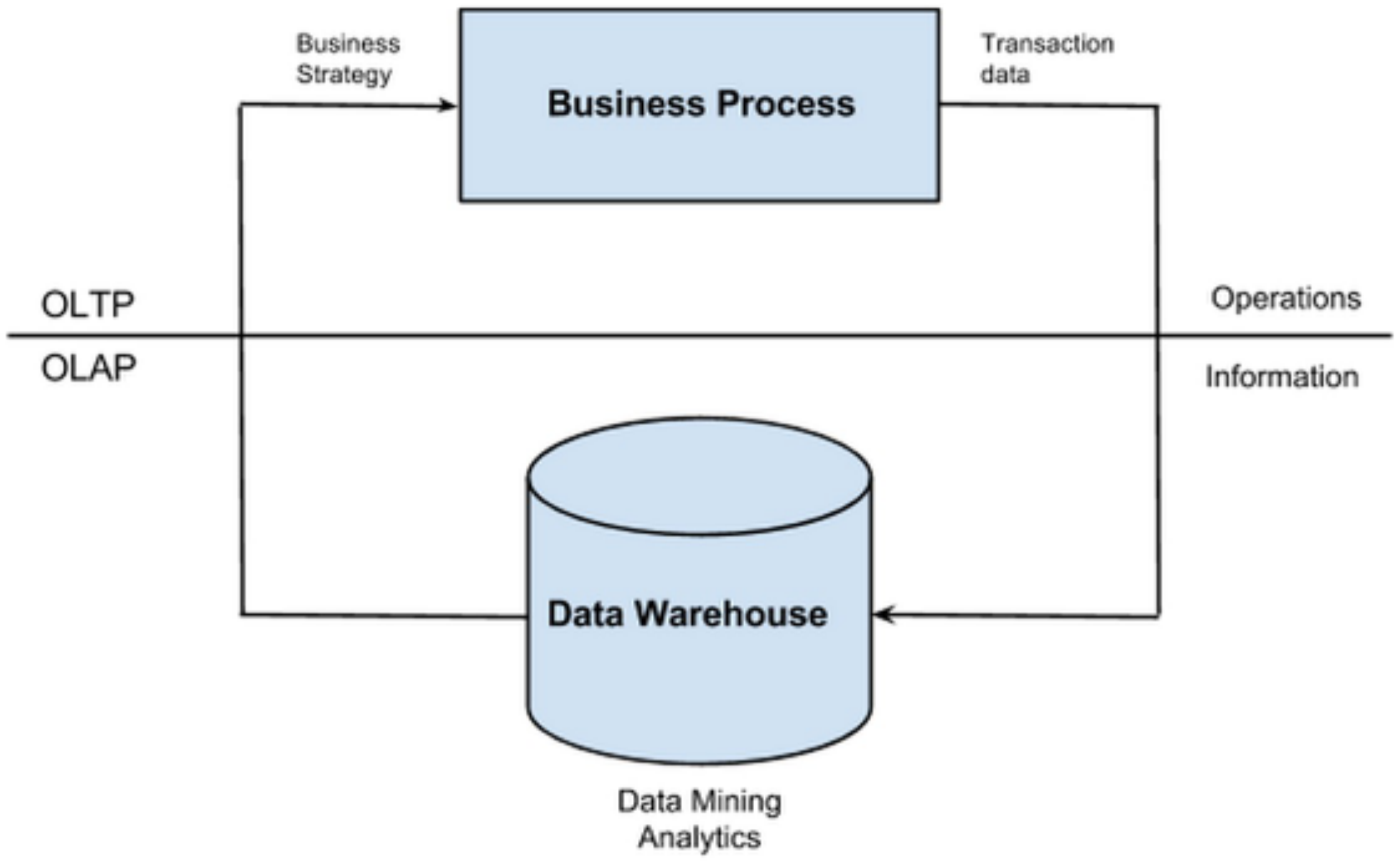


Figure 1. Feedback loop in a Business Intelligence system

Data Warehouse

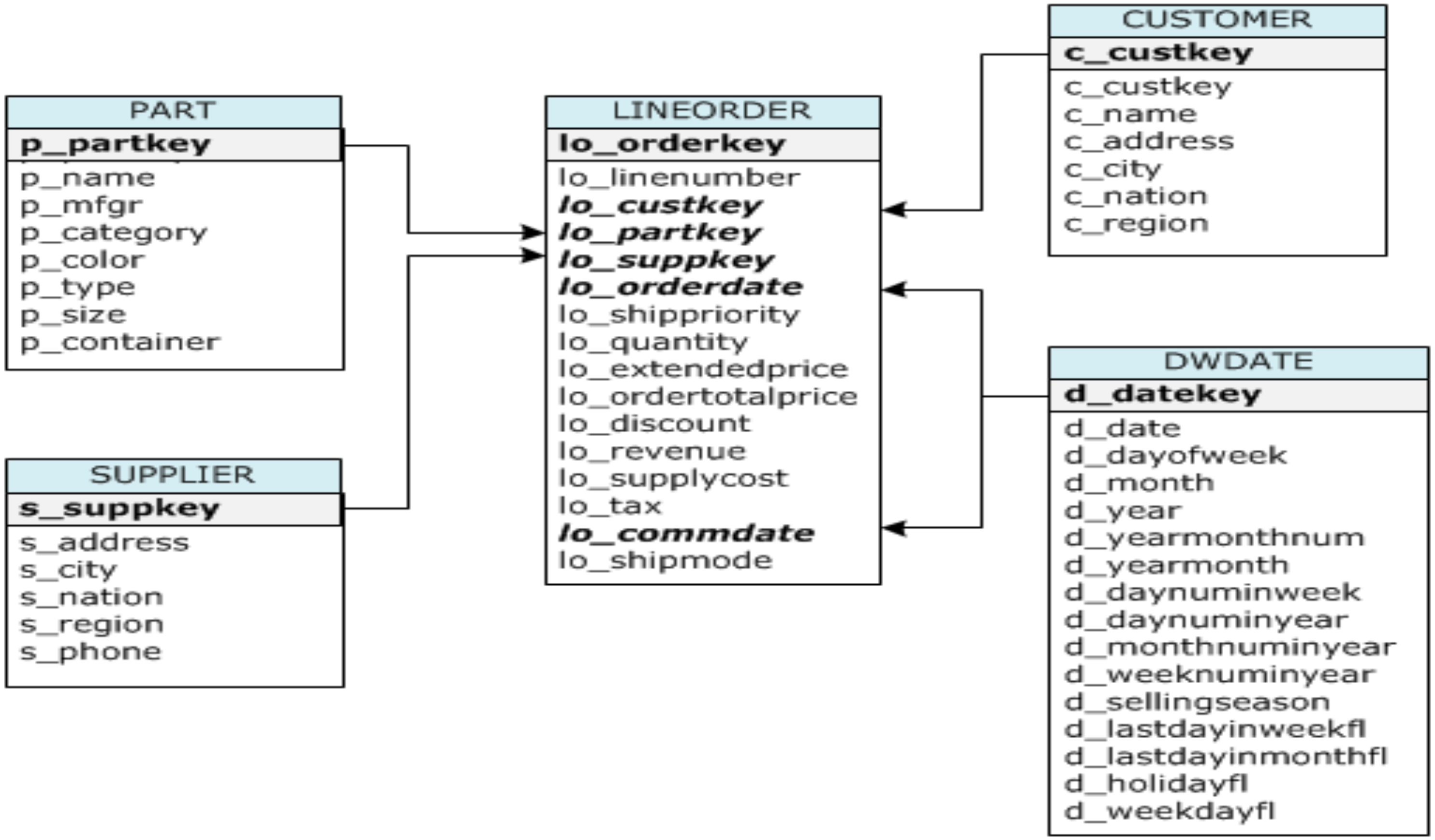
- Data Warehouses(DW) are large storage systems which store historical data that can be used for strategic decision making.
  - For business intelligence systems (data warehouse) OLAP (Online Analytical Processing) is used.

- Data Warehouse table design is structured for efficient querying and reporting.

Introduce Star Schema in DW

- Data warehouse databases commonly use a star schema design
  - A star schema is characterized by
    - one or more very large *fact* tables that contain the primary information in the data warehouse and
    - a number of much smaller *dimension* tables (or *lookup* tables), each of which contains information about the entries for a particular attribute in the fact table.

Example Star Schema



P3.4: Background

- Carnegie Records(CR) is booming with sales in more than a 100 countries and half a billion customers worldwide.
- CR want to analyze their historical sales data which has **several hundred million records**.
- CR asked you to benchmark multiple Business Intelligence (BI) tools so that they use the best system available for their workload
  - Benchmark multiple products for their requirements.
  - You have to use the Star Schema Benchmark (SSB).

P3.4: Tasks

The set of tasks that you have to complete:

Part 1: Compare

- MySQL
- Hive
- AWS Redshift

Part 2: Optimize Redshift

- Optimize table design for query performance

P3.4: Overview

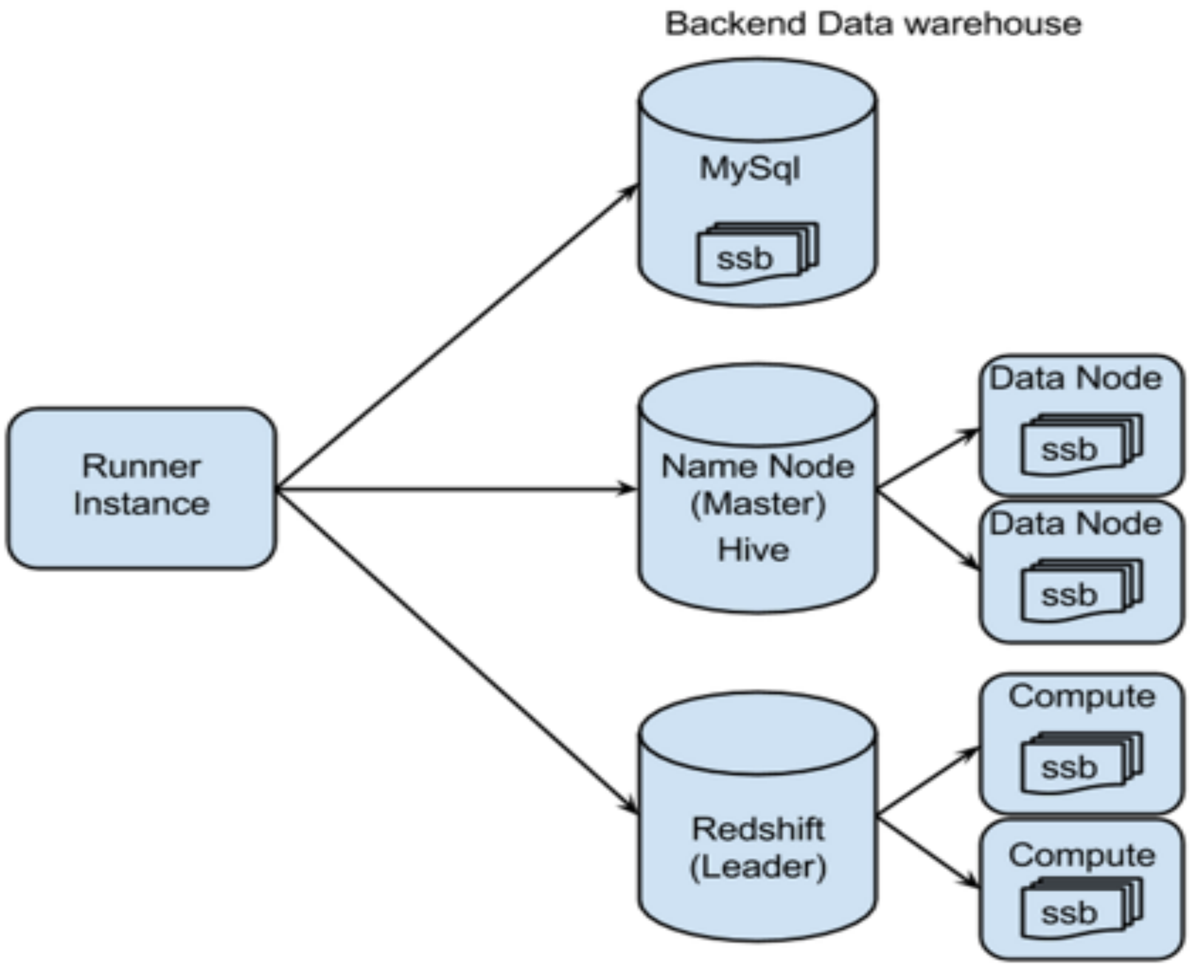
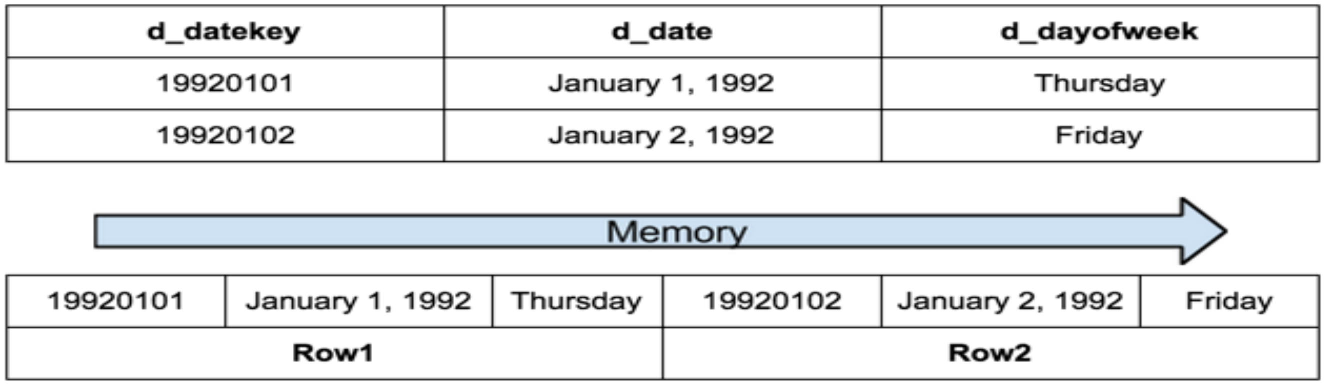


Figure 2. Project 3.4 Task Overview

Part 1: MySQL

- An open source row-store database.
- Row-wise database storage, data blocks store values sequentially for each consecutive column making up the entire row.
- Typically used for Online Transaction processing (OLTP)
  - Volume: 1K to 1M transactions/sec
  - Latency: > 1-50 ms
  - Database Size: 100s GB to 10s TB
  - typically more writes than reads



d_datekey	d_date	d_dayofweek
19920101	January 1, 1992	Thursday
19920102	January 2, 1992	Friday

19920101	January 1, 1992	Thursday	19920102	January 2, 1992	Friday
Row1			Row2		

Figure 4: Block level storage of rows in a row-store

Part 1: Hive

- The Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage (HDFS).

- Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL.

- Hive is best used for batch jobs over large sets of append-only data (like web logs).

Part 1: Hive

- OLAP System:
  - Volume: A couple queries per second
  - Latency: 1-60 seconds
  - Database Size: 100s TB to 10sPB
  - Many more reads than writes

- The most important characteristics of Hive are:
  - scalability (scale out with more machines added dynamically to the Hadoop cluster)
  - extensibility (with MapReduce framework)
  - fault-tolerance (provided by HDFS)

Part 1: Hive

- A high level overview of Hive and its relationship with HDFS

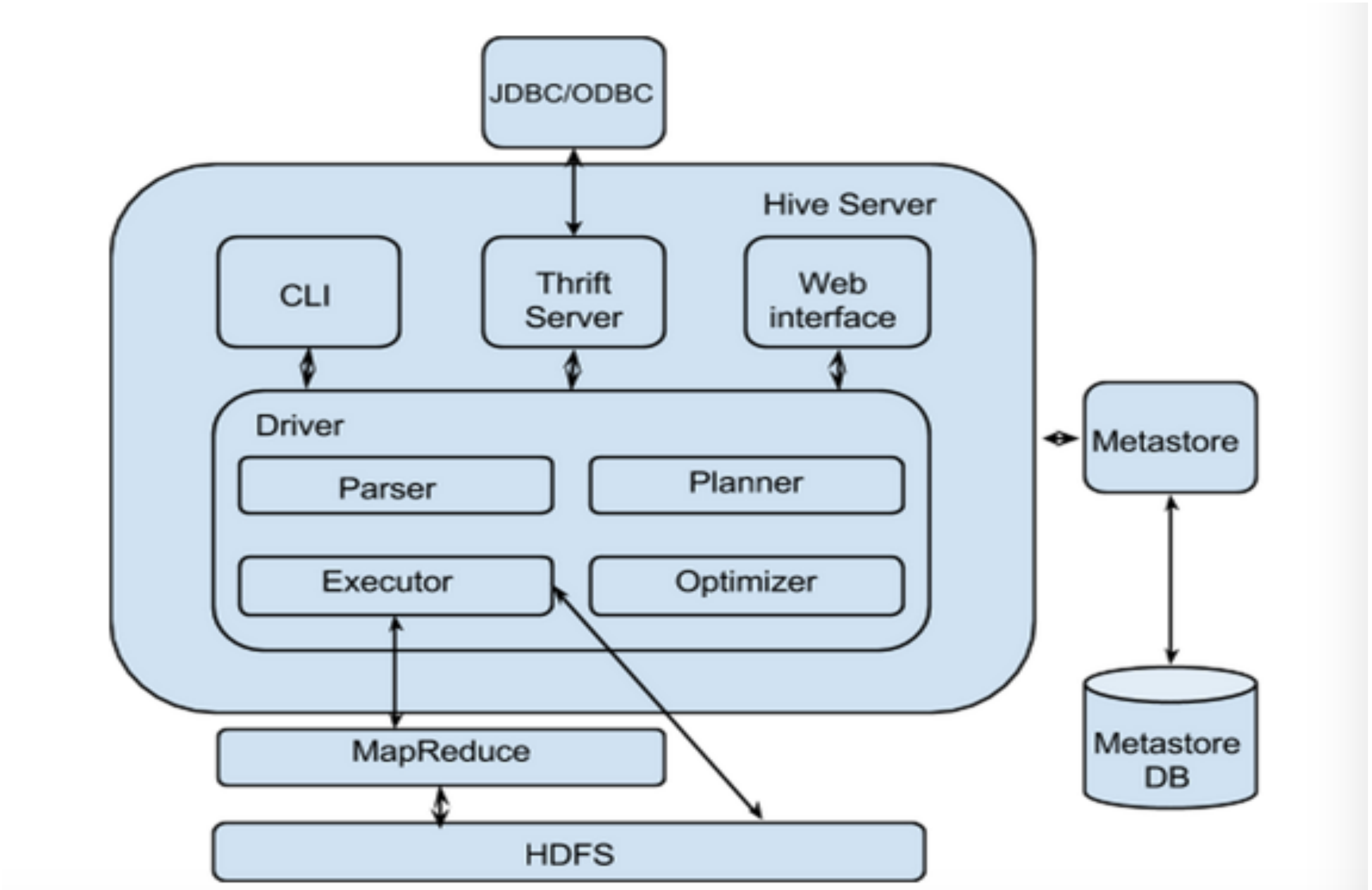


Figure 5. Hive Architecture

Part 1 : RedShift

- Enterprise-class relational database query and management system

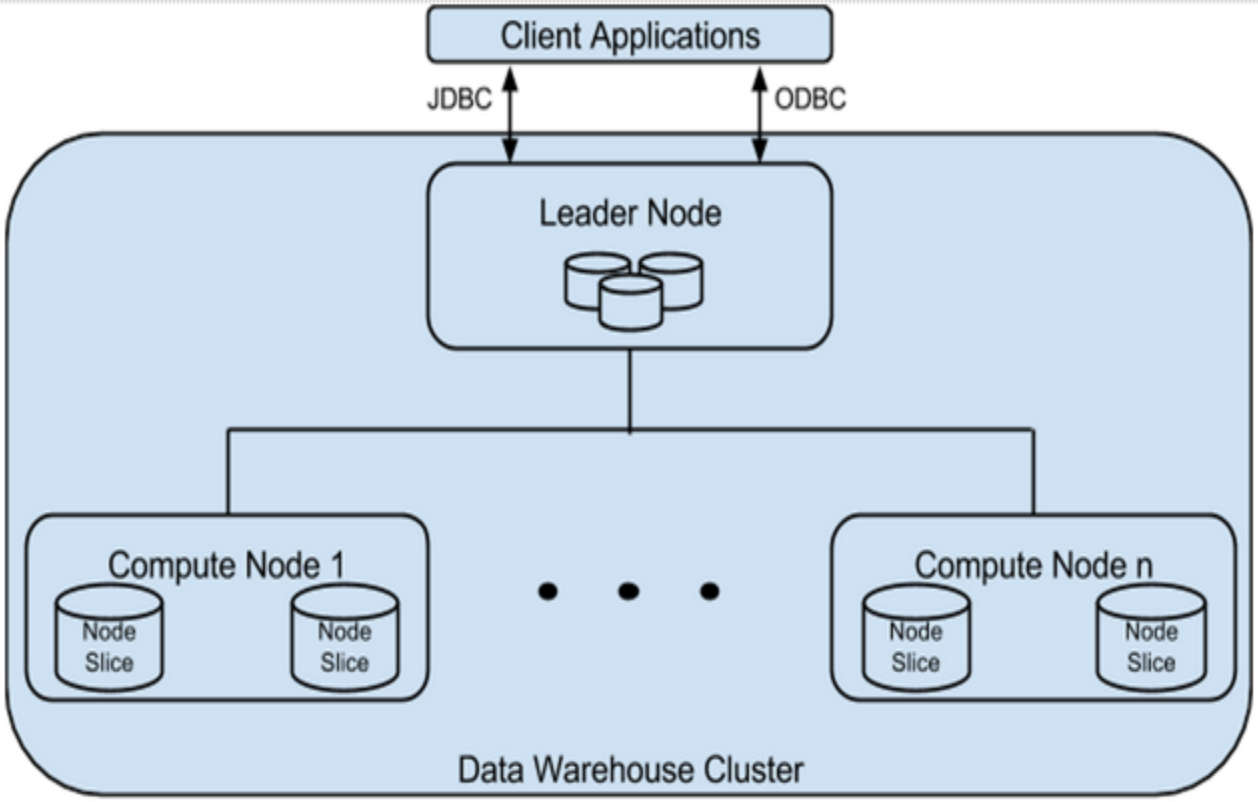
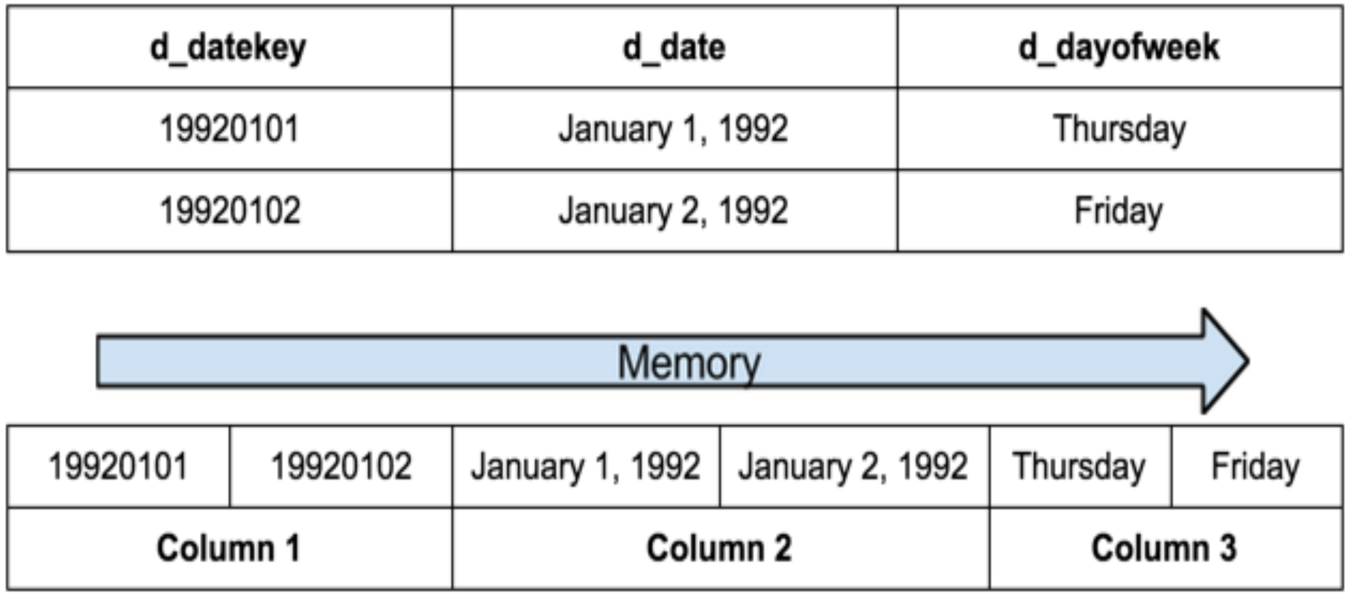


Figure 7: Redshift Data Warehouse architecture

Part 1 : RedShift

- Columnar storage



d_datekey	d_date	d_dayofweek
19920101	January 1, 1992	Thursday
19920102	January 2, 1992	Friday

19920101	19920102	January 1, 1992	January 2, 1992	Thursday	Friday
Column 1		**Column 2**		**Column 3**	

Figure 8: Block level storage of columns in column store

P3.4: What you need to do

Part 1: Comparison

- Provision a MySQL instance with the star schema benchmark data-set and benchmark the performance of query1 in ssb.
- Provision a Hive (EMR) Cluster. Create an external table in Hive that reads from S3 and benchmark the performance of query1 in ssb.
- Provision a Redshift cluster, load ssb data-set to structured (relational) tables and benchmark queries 1, 2 and 3 in ssb.

Part 2: Optimization

- Optimize the table structure in Redshift using sort keys and dist keys in order to improve the performance of the ssb queries by leveraging the parallel execution and columnar compression in distributed columnar stores.

Part 2 : Optimize RedShift

Sort Keys:

- While creating a table, you can specify one or more columns as the sort key.
- Redshift stores your data on disk in sorted order according to the sort key.
- How your data is sorted has an important effect on
  - disk I/O, columnar compression, and query performance.

Dist Keys:

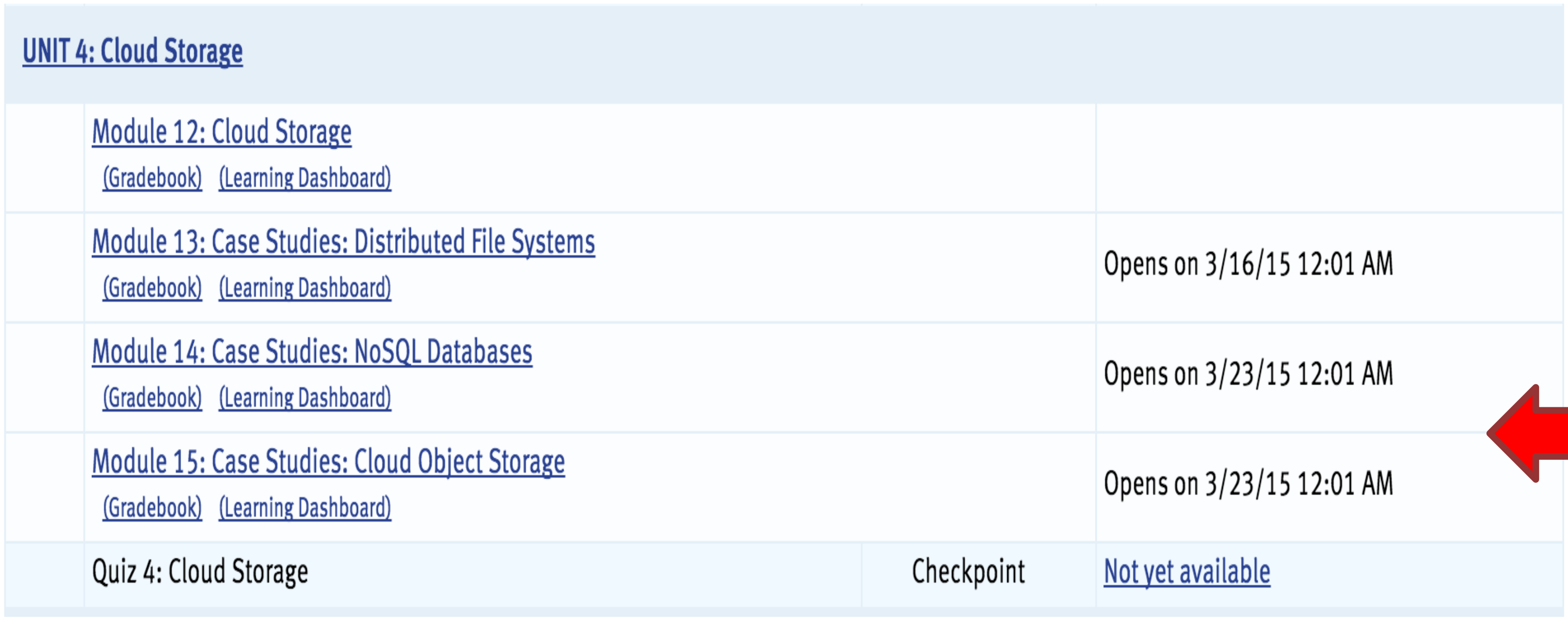
- When loading data into a table, Redshift distributes the rows of the table to each of the node slices according to the table's distribution style
- You should assign distribution styles to achieve these goals.
  - Collocate the rows from joining tables.
  - When the rows for joining columns are on the same slices, less data needs to be moved during query execution.
  - Distribute data evenly among the slices in a cluster.
  - If data is distributed evenly, the workload can be allocated evenly to all the slices.

P3.4: Possible hurdles

1. Do not run multiple Runner processes from the same instance at the same time.
   - This will report wrong results to the autograder.

2. Make sure the security groups on the backend and runner instance allow traffic on the ports on which the database listens.
   - MySQL:3306
   - Hive: 10000
   - RedShift: 5439

Unit 4 Module 14 & 15



UNIT 4: Cloud Storage		
Module 12: Cloud Storage (Gradebook) (Learning Dashboard)		
Module 13: Case Studies: Distributed File Systems (Gradebook) (Learning Dashboard)		Opens on 3/16/15 12:01 AM
Module 14: Case Studies: NoSQL Databases (Gradebook) (Learning Dashboard)		Opens on 3/23/15 12:01 AM
Module 15: Case Studies: Cloud Object Storage (Gradebook) (Learning Dashboard)		Opens on 3/23/15 12:01 AM
Quiz 4: Cloud Storage	Checkpoint	Not yet available

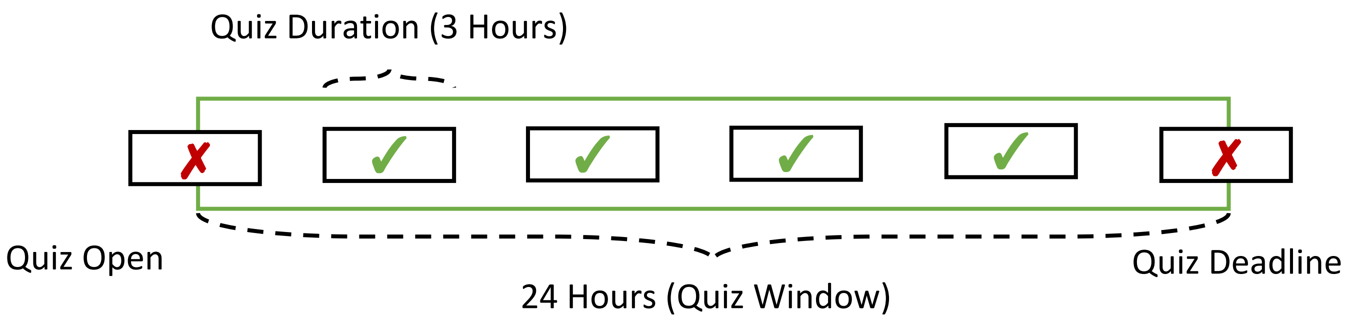
NoSQL Databases

- **Apache HBase**
  - An open-source version of Google's BigTable distributed storage system.
  - Both systems are distributed, scalable, high-performance, versioned databases.
- **MongoDB**
  - A document store that stores documents in collections.
- **Apache Cassandra**
  - A fully distributed, structured key-value storage system, which uses multiple design aspects of BigTable and Dynamo.
- **Amazon's DynamoDB**
  - DynamoDB is a managed NoSQL service provided by AWS.

Cloud Object Storage

- **Amazon S3**
  - Amazon Simple Storage Service (S3) is an on durable and scalable object storage service of by Amazon Web Services.
- **Openstack Swift**
  - OpenStack Swift is an open-source object storage system for public or private clouds.
- **Ceph object gateway**
  - Ceph object gateway is an access layer over the RADOS distributed object store.This offers both S3 and SWIFT compatible interfaces into RADOS.

Quiz 4



- Quiz 4 will be open for 24 hours, Friday, Mar 27
  - Quiz 4 becomes available on **Mar 27, 00:01 AM EST.**
  - Deadline for submission is **Mar 27, 11:59 PM EST.**
  - Once open, you have **180 min** to complete the quiz.
  - Late submissions are NOT accepted.
  - You may not start the quiz after the deadline has passed.
  - **Maintain your own timer from when you start the quiz.**
  - **Click submit before deadline passes. No Exceptions!**

Location	**Silicon Valley**	**Pittsburgh**	**Rwanda**	**Adelaide**
Open	Mar 26, 09:01 PM	Mar 27, 00:01 AM	Mar 27 06:01 AM	Mar 27 02:31 PM
Deadline	Mar 27, 08:59 PM	Mar 27, 11:59 PM	Mar 28 05:59 AM	Mar 28 02:29 PM

Quiz 4

- 5% of your Overall Grade
- You only have 1 attempt
- You can save your Quiz answers
  - Highly recommended
  - Save prompt every 15 minutes
- What can I expect from the Quiz?
  - Questions similar to the activities in the Units
  - multiple choice, fill-in-the-blanks, numeric questions, ...
- Feedback for Quiz 4 is released after the deadline passes

Upcoming Deadlines

- Modules 14 & 15
- Quiz 4
  - Due: 11:59PM ET Mar 27th (Friday)
- P3.4
  - Due: 11:59PM ET Mar 29th (Sunday)
- 15619Project Phase2
  - Due: 16:59PM ET Apr 1st (Wednesday)

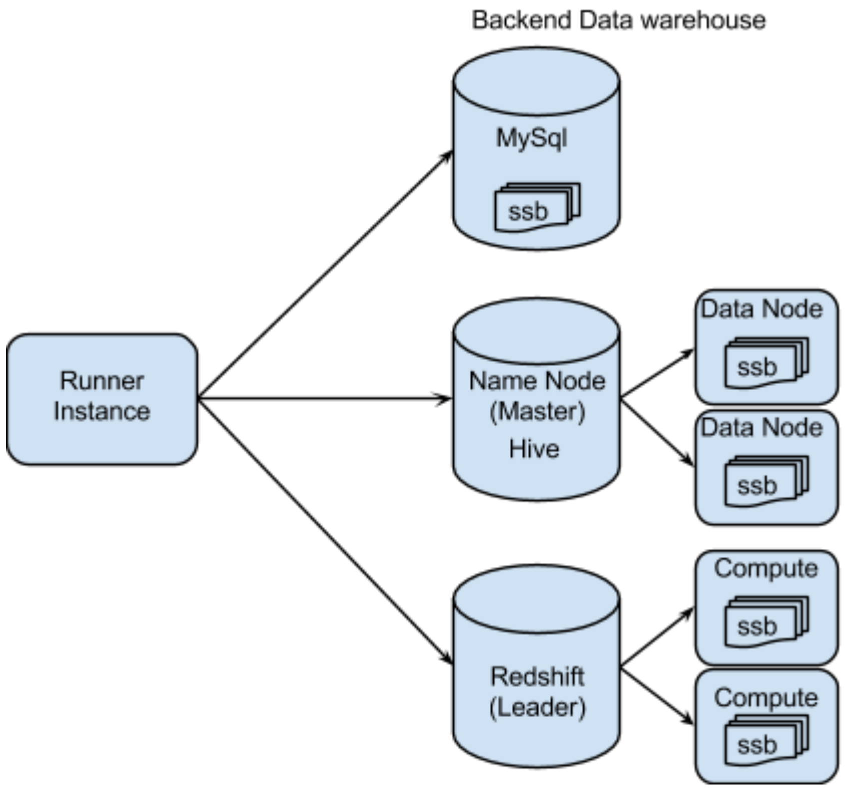
Project 3.4

Demo

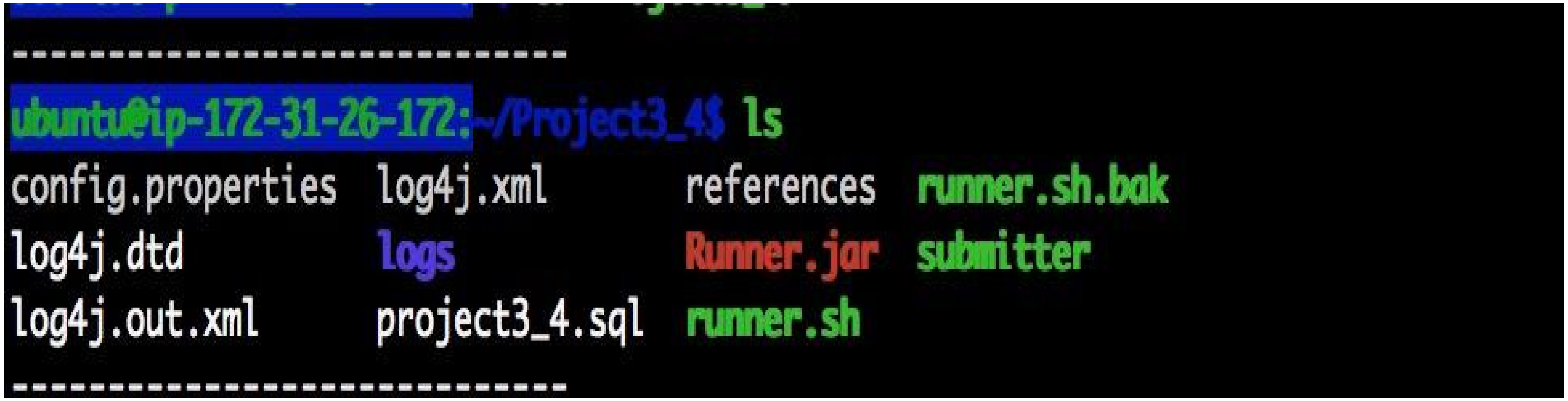
Overview

Test query benchmark on three different backend systems:

1. MySQL,
2. Hive, and
3. Amazon RedShift



Runner (Frontend)

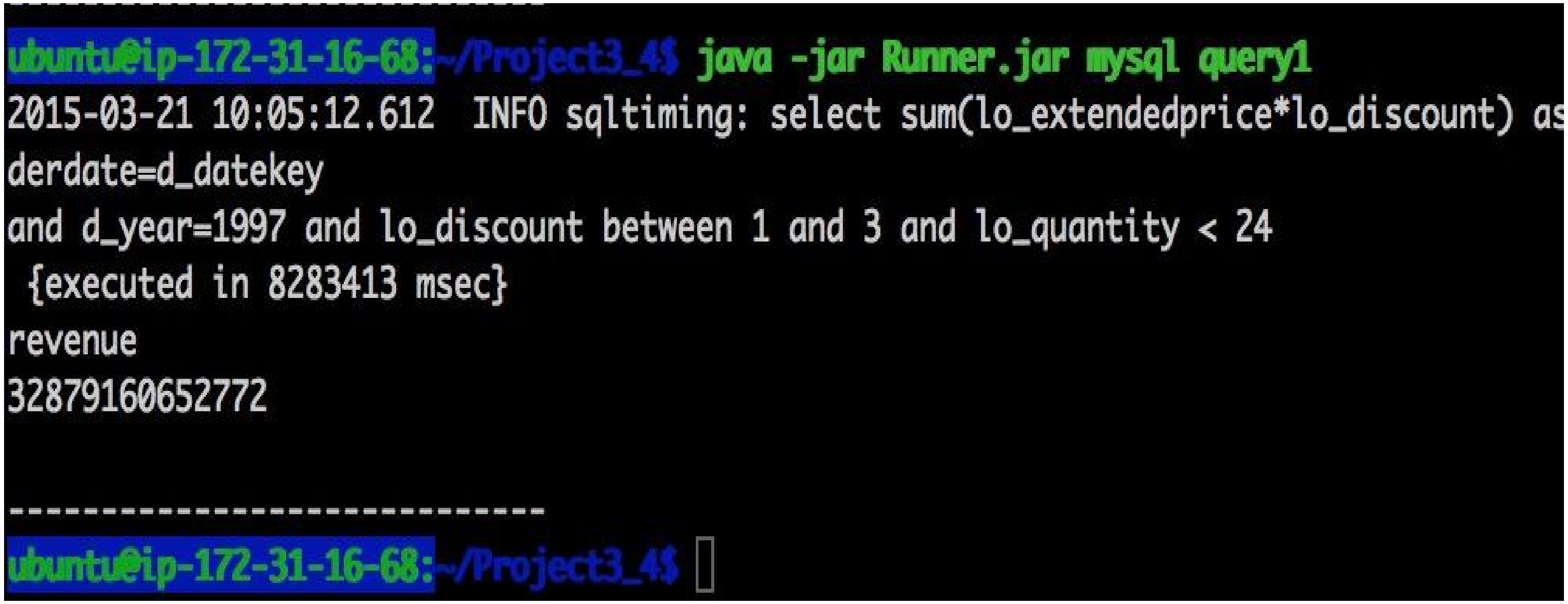


```
------------------------------
ubuntu@ip-172-31-26-172:~/Project3_4$ ls
config.properties  log4j.xml        references  runner.sh.bak
log4j.dtd          logs             Runner.jar  submitter
log4j.out.xml      project3_4.sql   runner.sh
------------------------------
```

java -jar Runner.jar <DBType> <Query>

Set up DNS for backend instances and username/password of RedShift

Task 1: MySQL

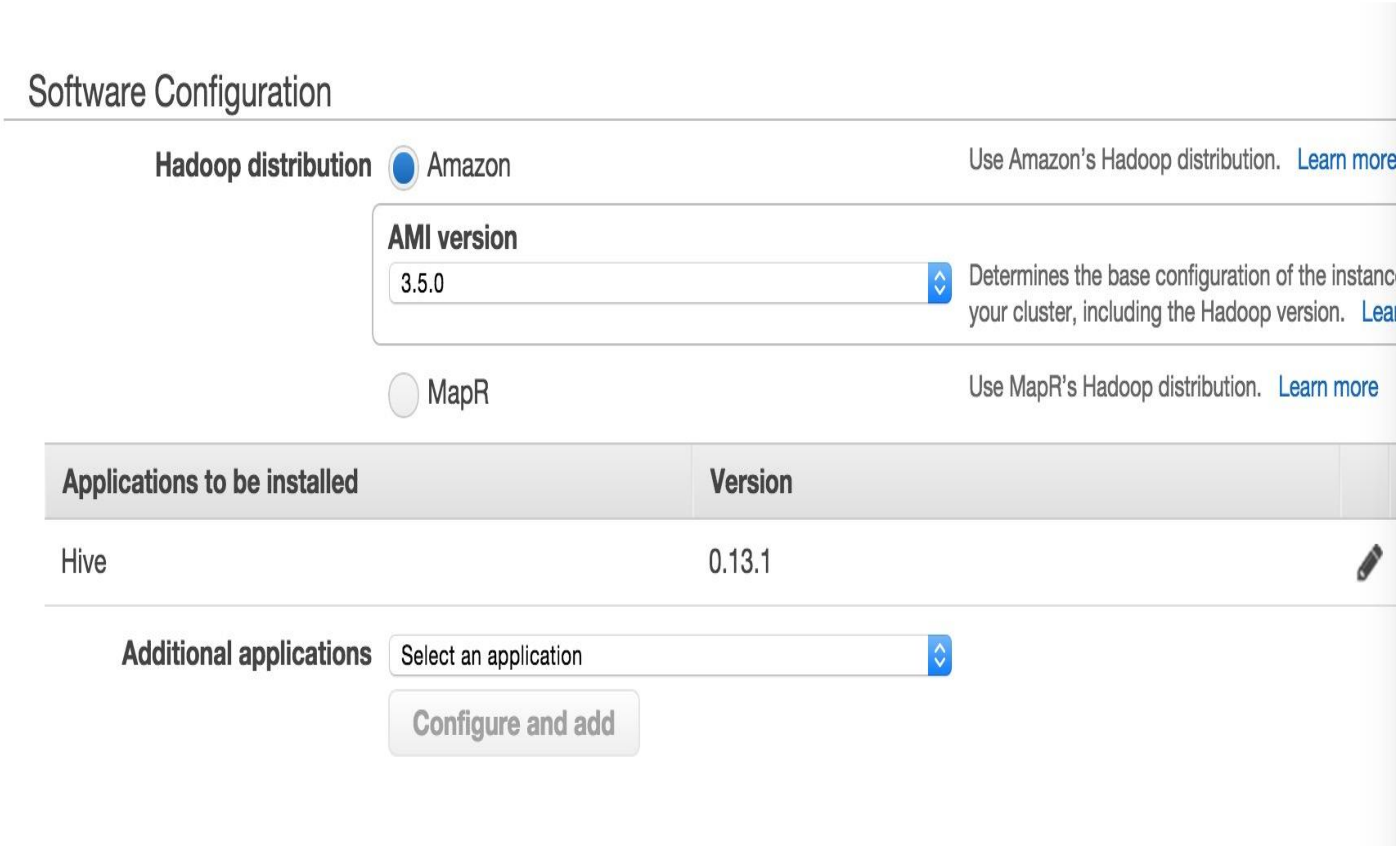


```
------------------------------
ubuntu@ip-172-31-16-68:~/Project3_4$ java -jar Runner.jar mysql query1
2015-03-21 10:05:12.612  INFO sqltiming: select sum(lo_extendedprice*lo_discount) as
derdate=d_datekey
and d_year=1997 and lo_discount between 1 and 3 and lo_quantity < 24
 {executed in 8283413 msec}
revenue
32879160652772


------------------------------
ubuntu@ip-172-31-16-68:~/Project3_4$
```

Please be patient…

Query runs for ~2 hours

Task 2: Hive



Software Configuration

Hadoop distribution (● Amazon) Use Amazon's Hadoop distribution. Learn more

AMI version

3.5.0

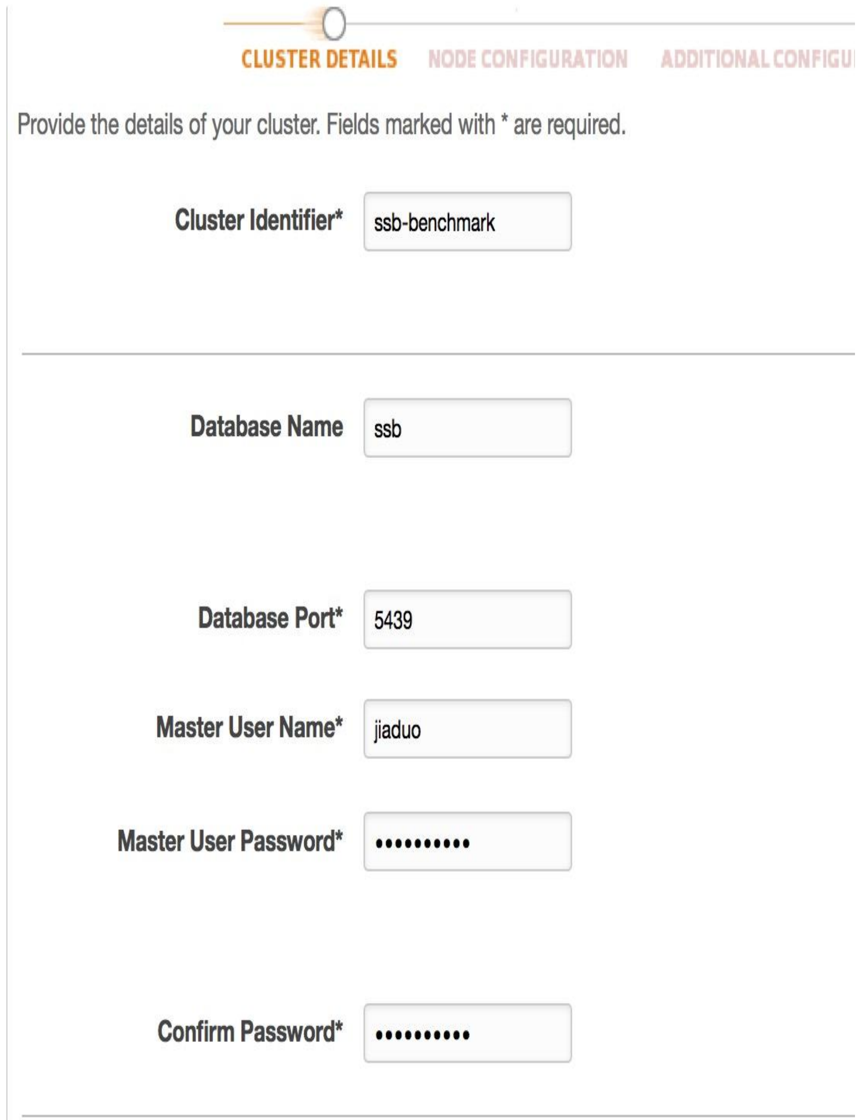
Determines the base configuration of the instanc
your cluster, including the Hadoop version. Lea

○ MapR Use MapR's Hadoop distribution. Learn more

Applications to be installed	Version
Hive	0.13.1

Additional applications: Select an application

Task 3: Amazon Redshift



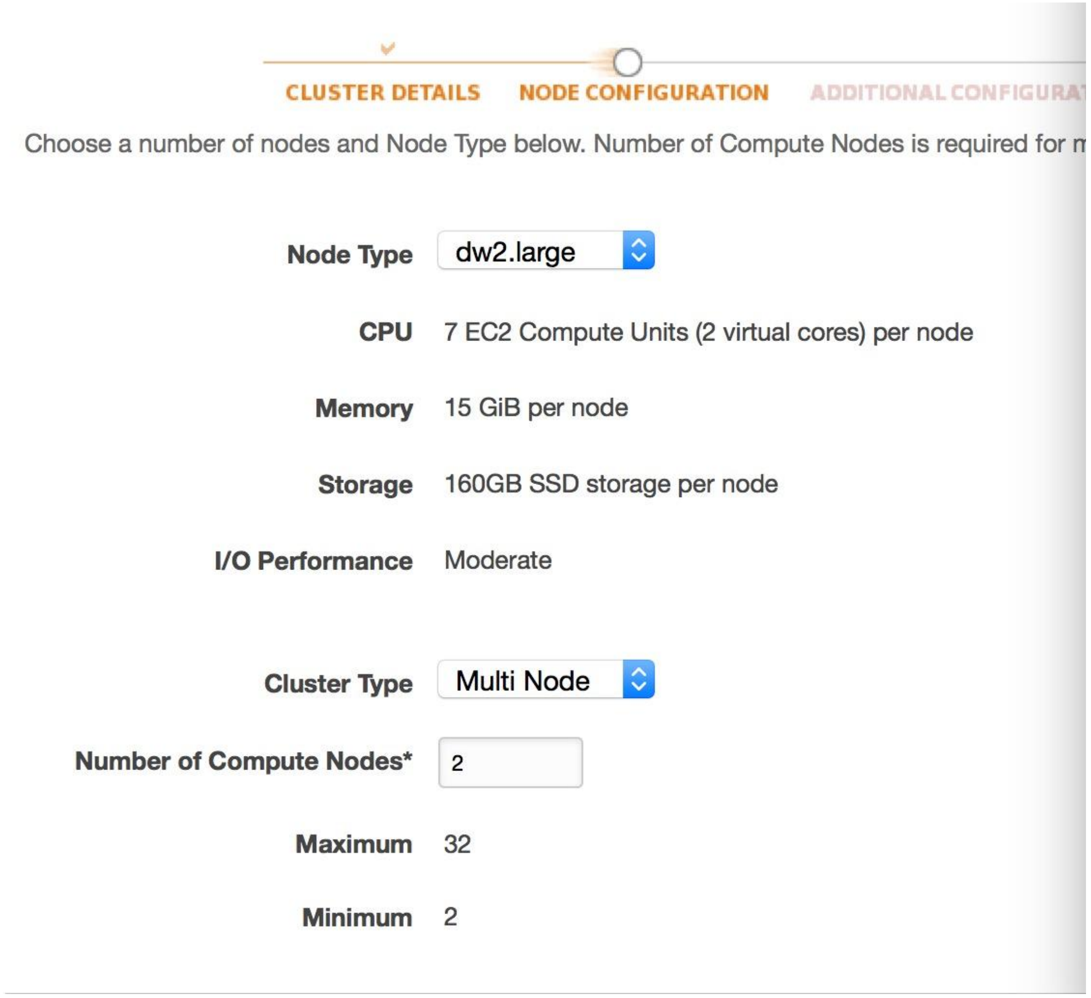
Provide the details of your cluster. Fields marked with * are required.

Cluster Identifier*	ssb-benchmark
Database Name	ssb
Database Port*	5439
Master User Name*	jiaduo
Master User Password*	••••••••••
Confirm Password*	••••••••••

Please make sure to add your <u>username</u> and and <u>password</u> in config.properties

Redshift

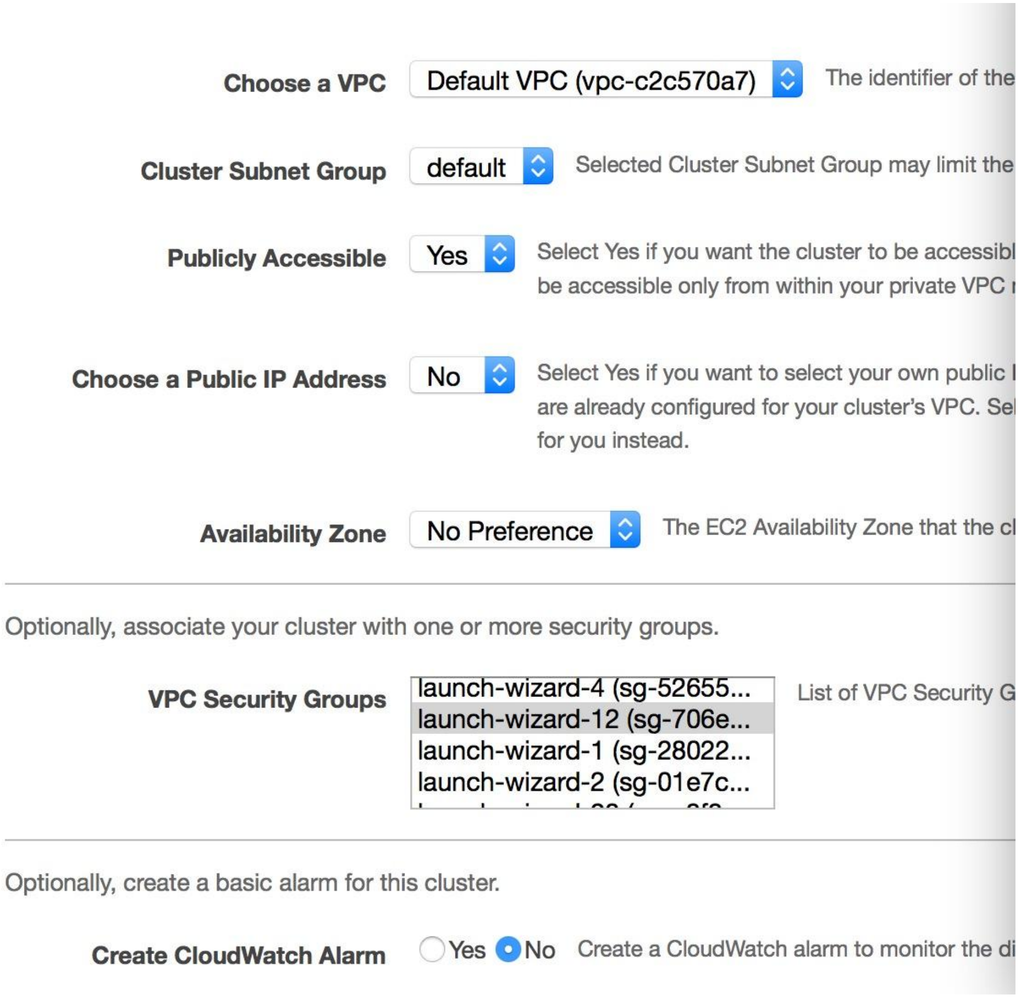
Configuration



Choose a number of nodes and Node Type below. Number of Compute Nodes is required for m

Node Type	dw2.large
CPU	7 EC2 Compute Units (2 virtual cores) per node
Memory	15 GiB per node
Storage	160GB SSD storage per node
I/O Performance	Moderate
Cluster Type	Multi Node
Number of Compute Nodes*	2
Maximum	32
Minimum	2

Set the security groups to allow traffic on port 5439



Choose a VPC	Default VPC (vpc-c2c570a7)	The identifier of the
Cluster Subnet Group	default	Selected Cluster Subnet Group may limit the
Publicly Accessible	Yes	Select Yes if you want the cluster to be accessibl be accessible only from within your private VPC
Choose a Public IP Address	No	Select Yes if you want to select your own public are already configured for your cluster's VPC. Se for you instead.
Availability Zone	No Preference	The EC2 Availability Zone that the c

Optionally, associate your cluster with one or more security groups.

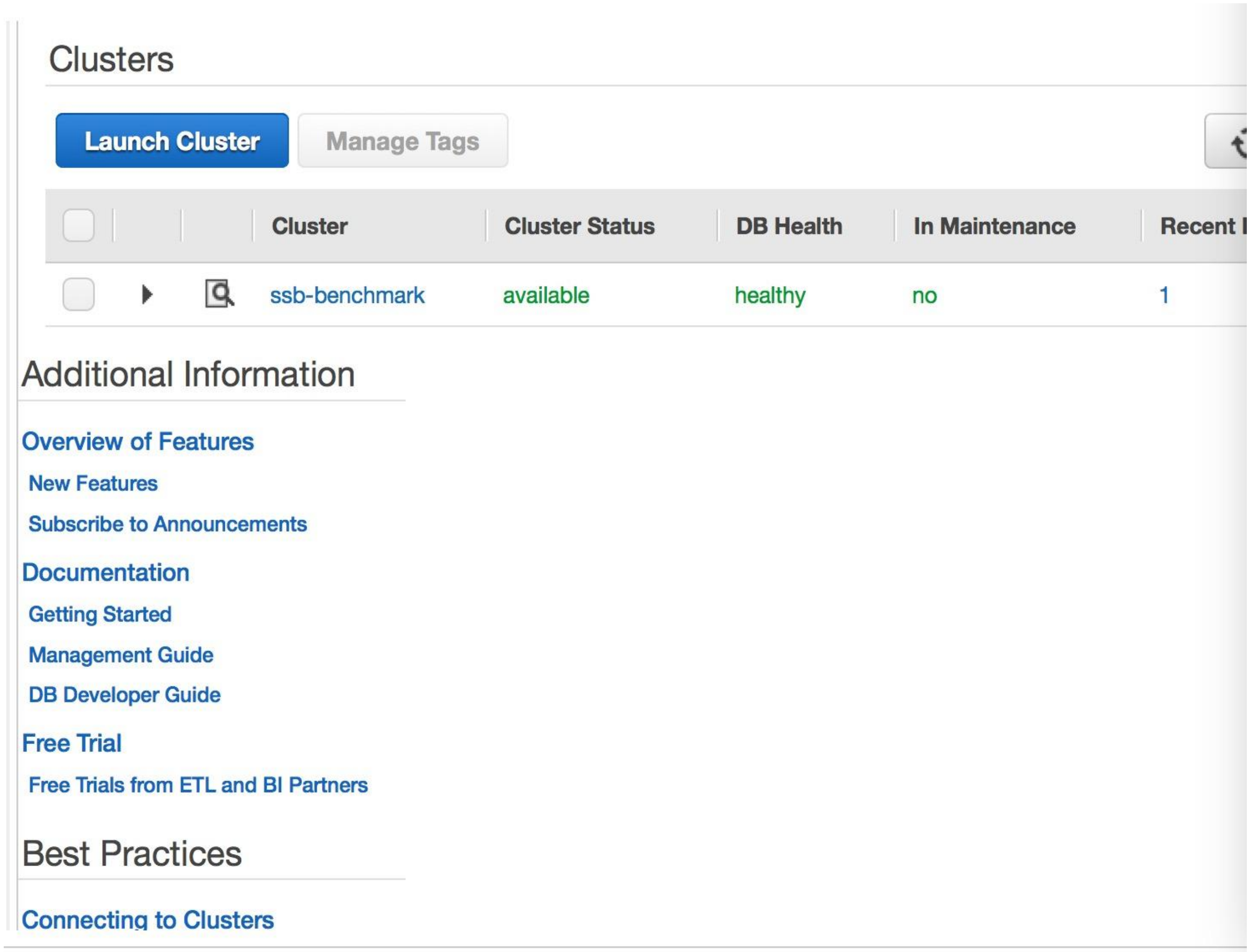
VPC Security Groups
- launch-wizard-4 (sg-52655...
- launch-wizard-12 (sg-706e...
- launch-wizard-1 (sg-28022...
- launch-wizard-2 (sg-01e7c...

List of VPC Security G

Optionally, create a basic alarm for this cluster.

Create CloudWatch Alarm: Yes [✓ No] Create a CloudWatch alarm to monitor the di

Redshift Status Page



Clusters

			Cluster	Cluster Status	DB Health	In Maintenance	Recent I
			ssb-benchmark	available	healthy	no	1

Additional Information

Overview of Features

New Features

Subscribe to Announcements

Documentation

Getting Started

Management Guide

DB Developer Guide

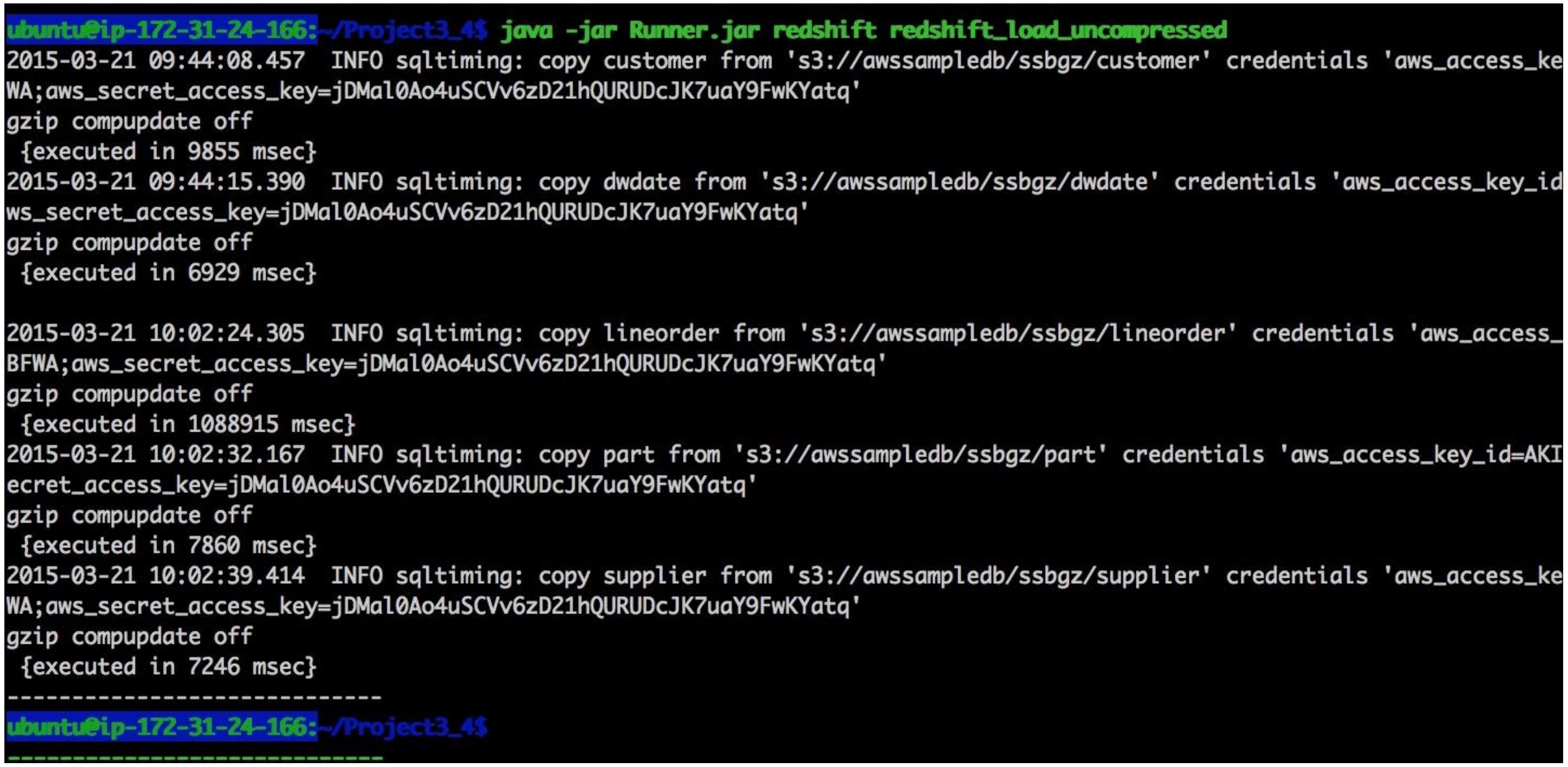
Free Trial

Free Trials from ETL and BI Partners

Best Practices

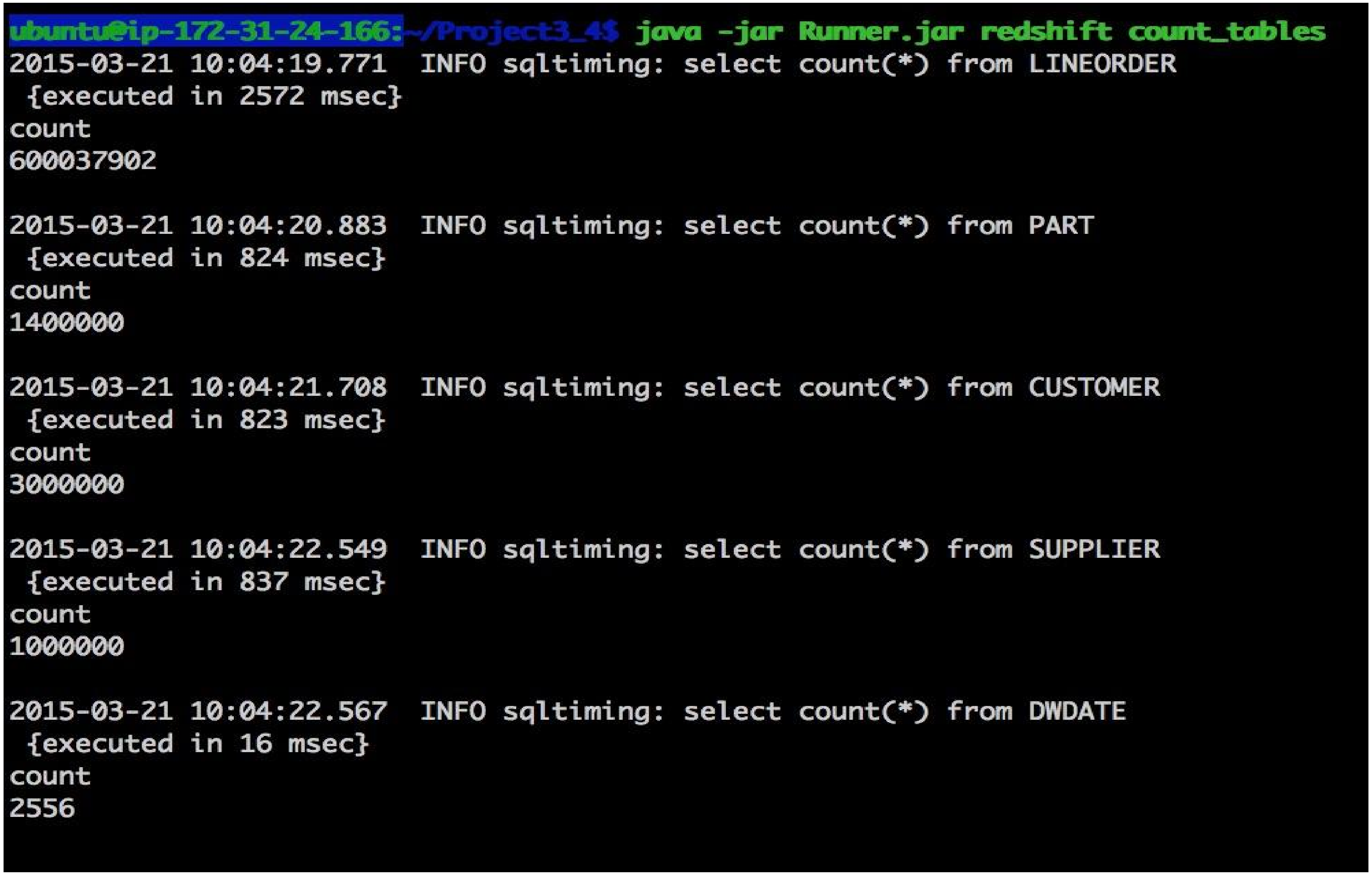
Connecting to Clusters

Load data into Redshift



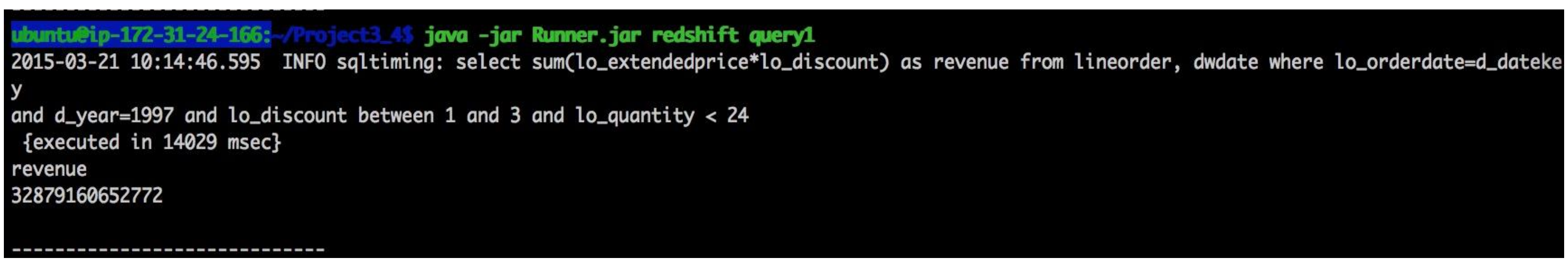
```
ubuntu@ip-172-31-24-166:~/Project3_4$ java -jar Runner.jar redshift redshift_load_uncompressed
2015-03-21 09:44:08.457  INFO sqltiming: copy customer from 's3://awssampledb/ssbgz/customer' credentials 'aws_access_ke
WA;aws_secret_access_key=jDMal0Ao4uSCVv6zD21hQURUDcJK7uaY9FwKYatq'
gzip compupdate off
 {executed in 9855 msec}
2015-03-21 09:44:15.390  INFO sqltiming: copy dwdate from 's3://awssampledb/ssbgz/dwdate' credentials 'aws_access_key_id
ws_secret_access_key=jDMal0Ao4uSCVv6zD21hQURUDcJK7uaY9FwKYatq'
gzip compupdate off
 {executed in 6929 msec}

2015-03-21 10:02:24.305  INFO sqltiming: copy lineorder from 's3://awssampledb/ssbgz/lineorder' credentials 'aws_access_
BFWA;aws_secret_access_key=jDMal0Ao4uSCVv6zD21hQURUDcJK7uaY9FwKYatq'
gzip compupdate off
 {executed in 1088915 msec}
2015-03-21 10:02:32.167  INFO sqltiming: copy part from 's3://awssampledb/ssbgz/part' credentials 'aws_access_key_id=AKI
ecret_access_key=jDMal0Ao4uSCVv6zD21hQURUDcJK7uaY9FwKYatq'
gzip compupdate off
 {executed in 7860 msec}
2015-03-21 10:02:39.414  INFO sqltiming: copy supplier from 's3://awssampledb/ssbgz/supplier' credentials 'aws_access_ke
WA;aws_secret_access_key=jDMal0Ao4uSCVv6zD21hQURUDcJK7uaY9FwKYatq'
gzip compupdate off
 {executed in 7246 msec}
-----------------------------
ubuntu@ip-172-31-24-166:~/Project3_4$
-----------------------------
```

Check rows in tables

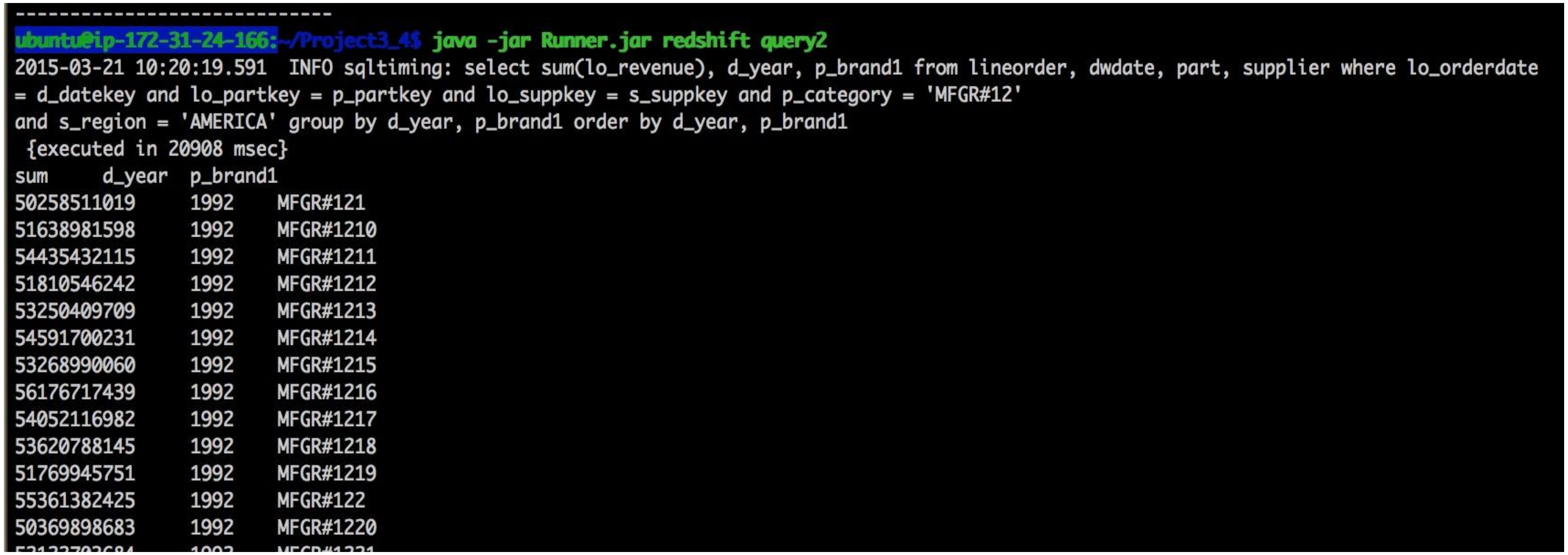


```
ubuntu@ip-172-31-24-166:~/Project3_4$ java -jar Runner.jar redshift count_tables
2015-03-21 10:04:19.771  INFO sqltiming: select count(*) from LINEORDER
 {executed in 2572 msec}
count
600037902

2015-03-21 10:04:20.883  INFO sqltiming: select count(*) from PART
 {executed in 824 msec}
count
1400000

2015-03-21 10:04:21.708  INFO sqltiming: select count(*) from CUSTOMER
 {executed in 823 msec}
count
3000000

2015-03-21 10:04:22.549  INFO sqltiming: select count(*) from SUPPLIER
 {executed in 837 msec}
count
1000000

2015-03-21 10:04:22.567  INFO sqltiming: select count(*) from DWDATE
 {executed in 16 msec}
count
2556
```

Run each query three times

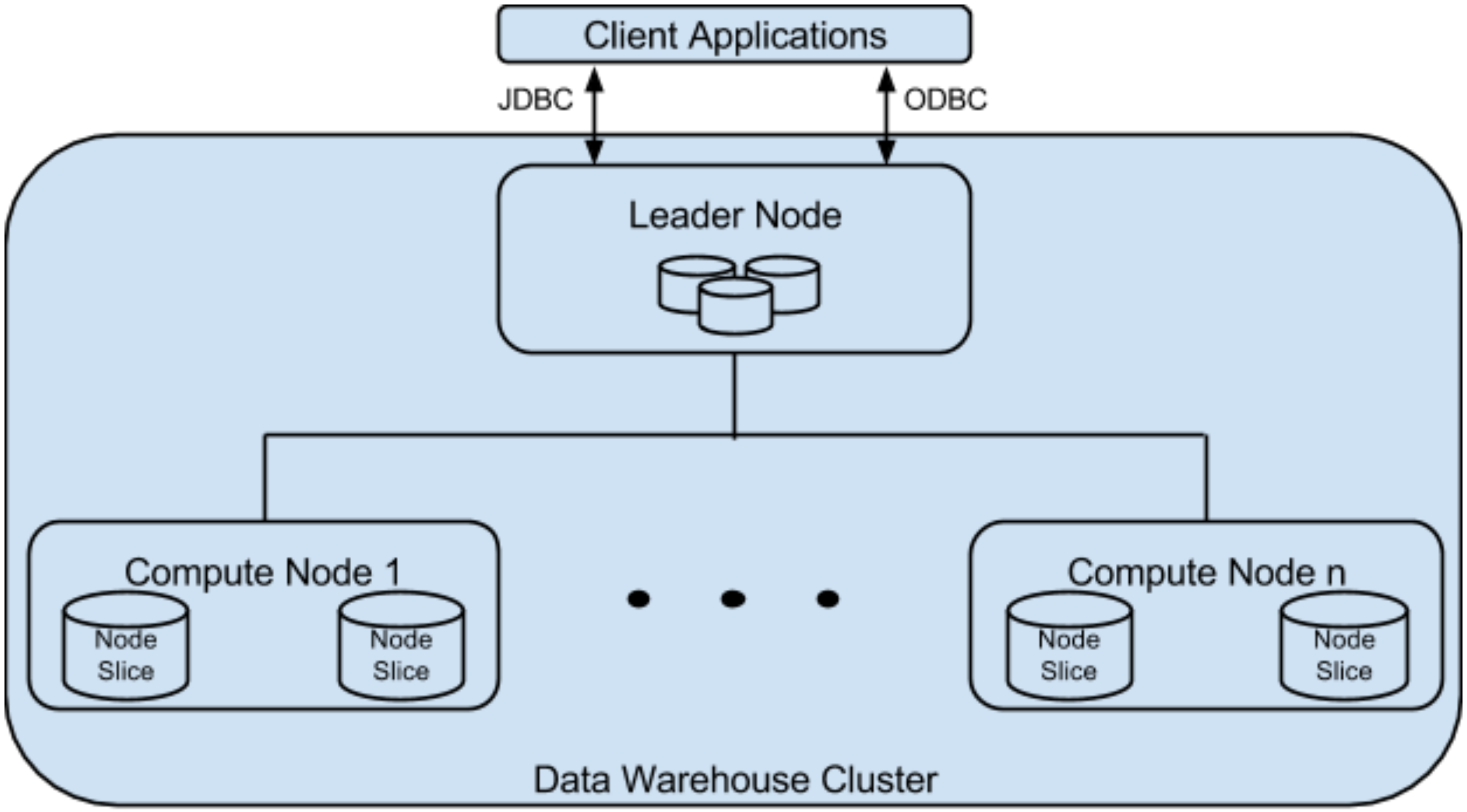


```
ubuntu@ip-172-31-24-166:~/Project3_4$ java -jar Runner.jar redshift query1
2015-03-21 10:14:46.595  INFO sqltiming: select sum(lo_extendedprice*lo_discount) as revenue from lineorder, dwdate where lo_orderdate=d_dateke
y
and d_year=1997 and lo_discount between 1 and 3 and lo_quantity < 24
 {executed in 14029 msec}
revenue
32879160652772

-----------------------------
```



```
-----------------------------
ubuntu@ip-172-31-24-166:~/Project3_4$ java -jar Runner.jar redshift query2
2015-03-21 10:20:19.591  INFO sqltiming: select sum(lo_revenue), d_year, p_brand1 from lineorder, dwdate, part, supplier where lo_orderdate
= d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_category = 'MFGR#12'
and s_region = 'AMERICA' group by d_year, p_brand1 order by d_year, p_brand1
 {executed in 20908 msec}
sum     d_year  p_brand1
50258511019     1992    MFGR#121
51638981598     1992    MFGR#1210
54435432115     1992    MFGR#1211
51810546242     1992    MFGR#1212
53250409709     1992    MFGR#1213
54591700231     1992    MFGR#1214
53268990060     1992    MFGR#1215
56176717439     1992    MFGR#1216
54052116982     1992    MFGR#1217
53620788145     1992    MFGR#1218
51769945751     1992    MFGR#1219
55361382425     1992    MFGR#122
50369898683     1992    MFGR#1220
```

Optimizing query performance



Design considerations

- Workload Balancing

- Communication Cost

Two ways of improvement

- Sort key

  Keys are stored in sorted order for range filtering and compression.

- Dist key

  Distribute keys to every compute node.

Optimized table benchmark

drop_tables

redshift_create_table_optimized

redshift_load_compressed

query1, query2, query3

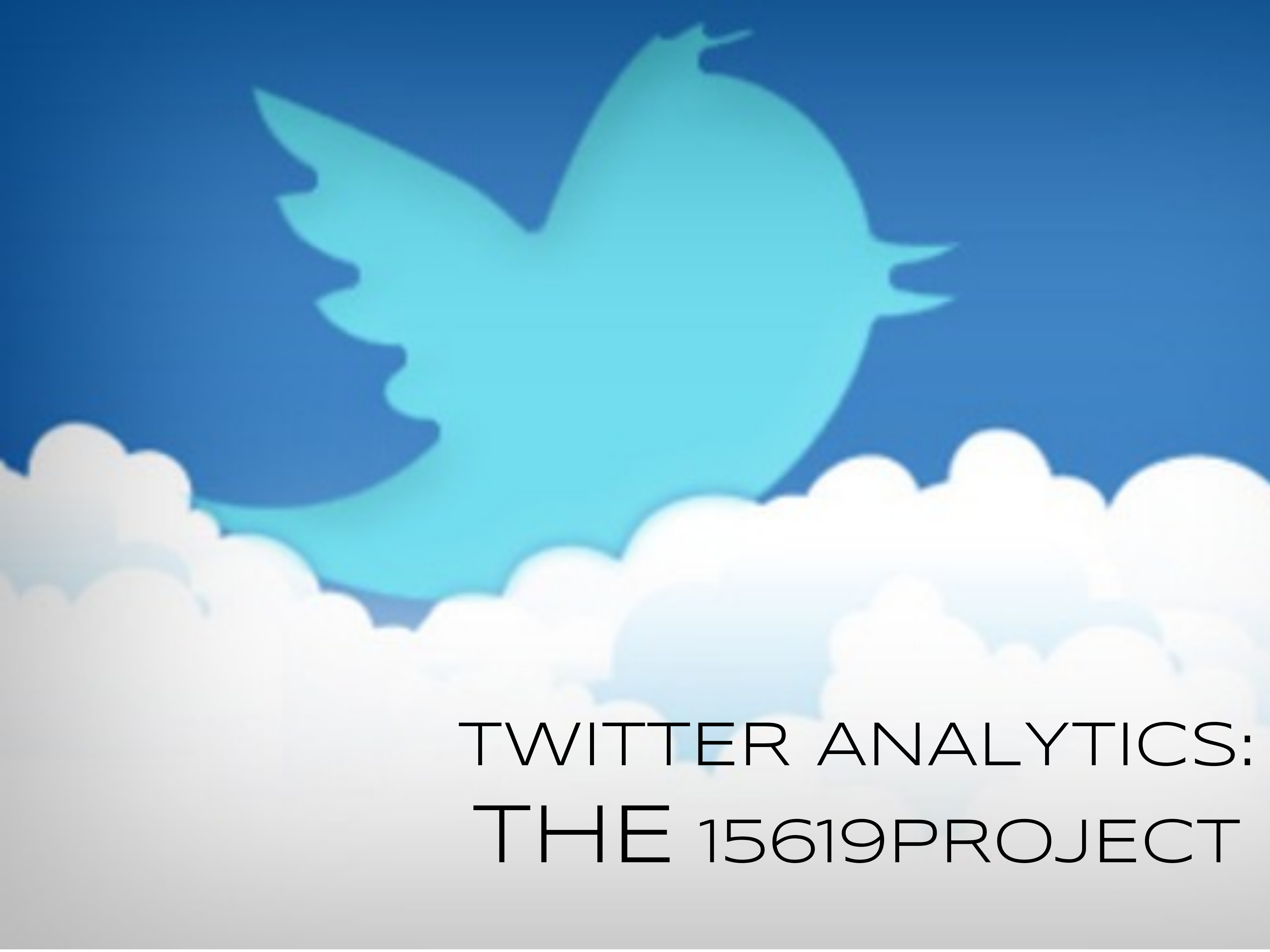
analyze_compression

Don’t forget the quiz for P3.4!

After completing all the tasks:

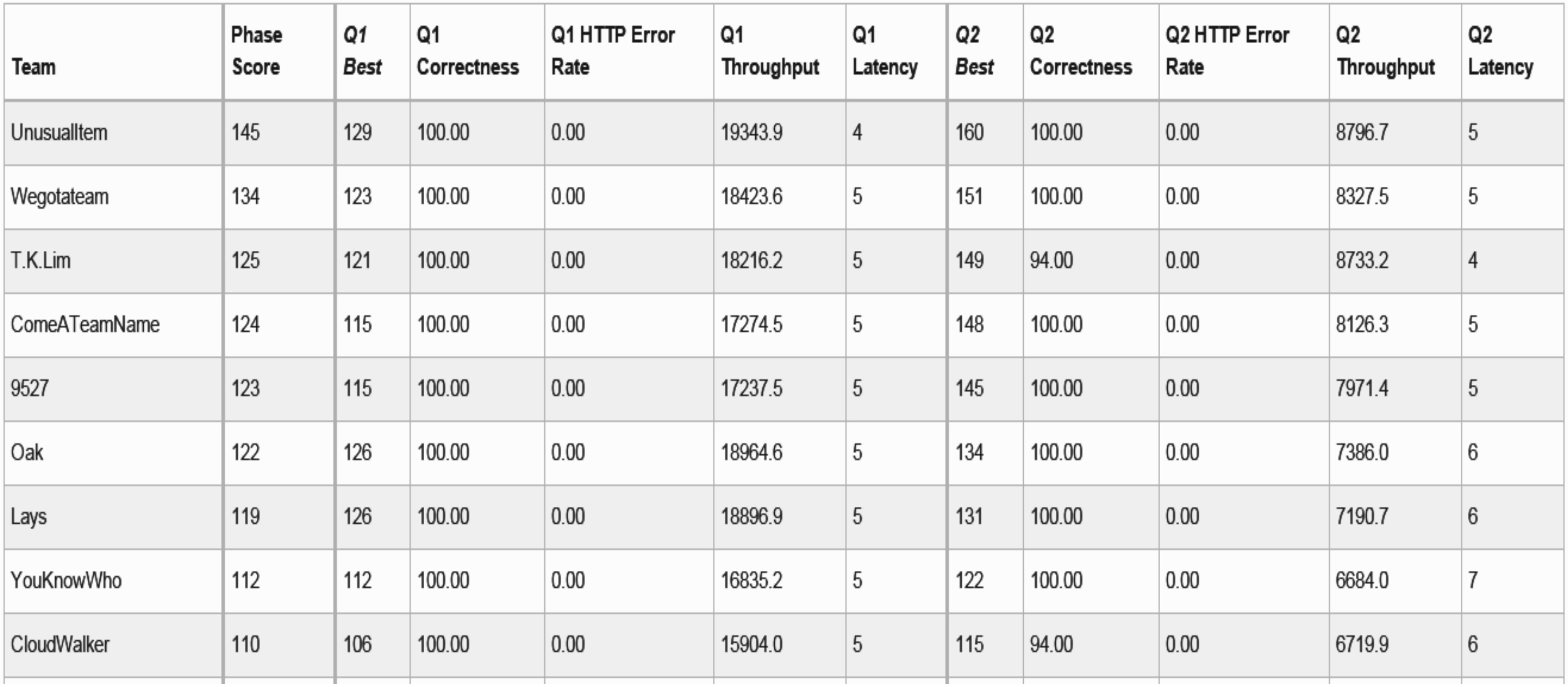
1. Edit runner.sh and answer the questions
2. Edit the references file
3. Submit your answers

Questions?



TWITTER ANALYTICS:
THE 15619PROJECT

Phase 1 Leaderboard



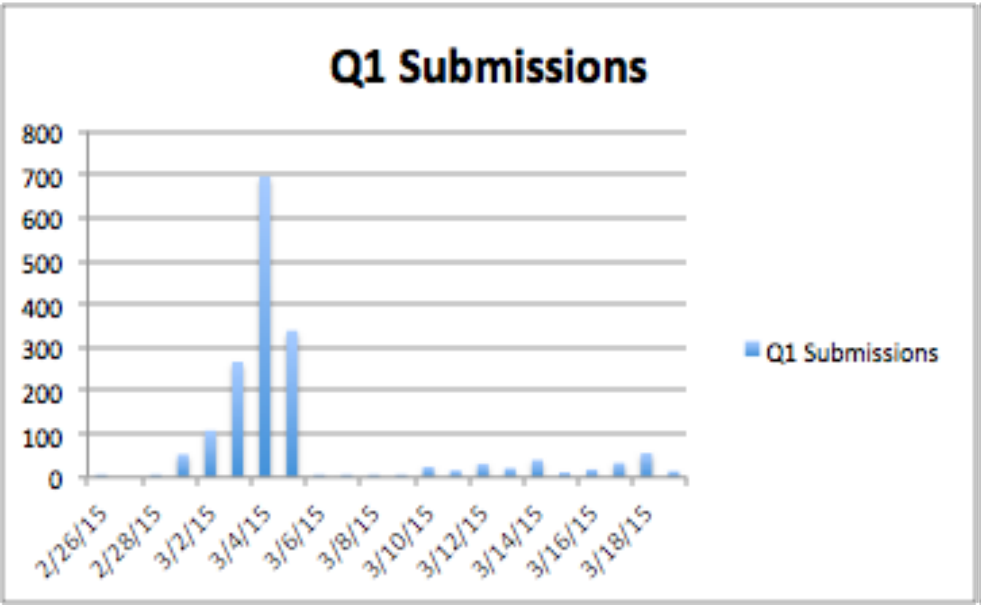
Team	Phase Score	*Q1 Best*	Q1 Correctness	Q1 HTTP Error Rate	Q1 Throughput	Q1 Latency	*Q2 Best*	Q2 Correctness	Q2 HTTP Error Rate	Q2 Throughput	Q2 Latency
UnusualItem	145	129	100.00	0.00	19343.9	4	160	100.00	0.00	8796.7	5
Wegotateam	134	123	100.00	0.00	18423.6	5	151	100.00	0.00	8327.5	5
T.K.Lim	125	121	100.00	0.00	18216.2	5	149	94.00	0.00	8733.2	4
ComeATeamName	124	115	100.00	0.00	17274.5	5	148	100.00	0.00	8126.3	5
9527	123	115	100.00	0.00	17237.5	5	145	100.00	0.00	7971.4	5
Oak	122	126	100.00	0.00	18964.6	5	134	100.00	0.00	7386.0	6
Lays	119	126	100.00	0.00	18896.9	5	131	100.00	0.00	7190.7	6
YouKnowWho	112	112	100.00	0.00	16835.2	5	122	100.00	0.00	6684.0	7
CloudWalker	110	106	100.00	0.00	15904.0	5	115	94.00	0.00	6719.9	6

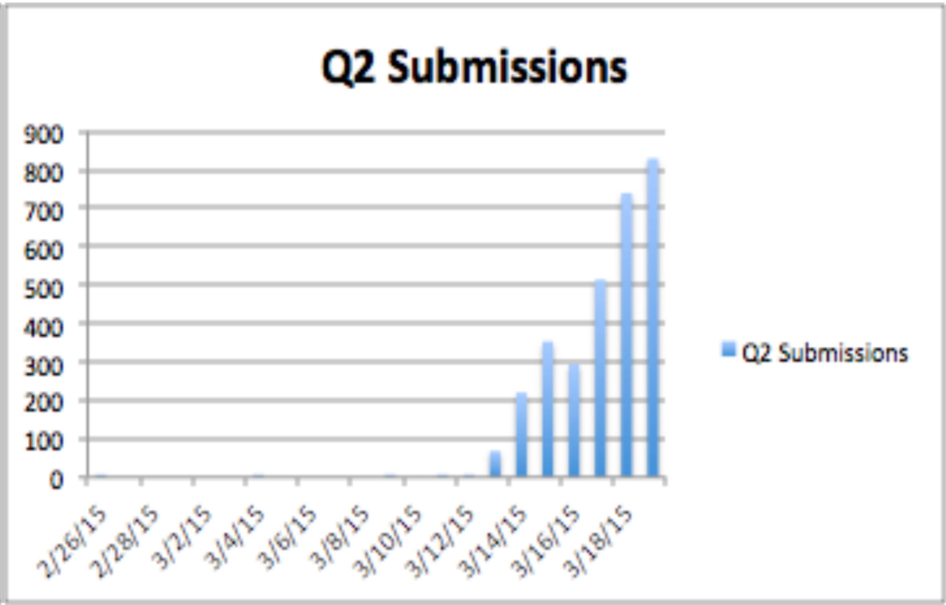
Well done !!!

Congratulations UnusualItem

Phase 1 Statistics

- Total Phase 1 submissions: 3932
  - Q1 1733
  - Q2 2199
- Phase 1 scores
  - Average 54.9
  - Median 36, Max:145 Min:9



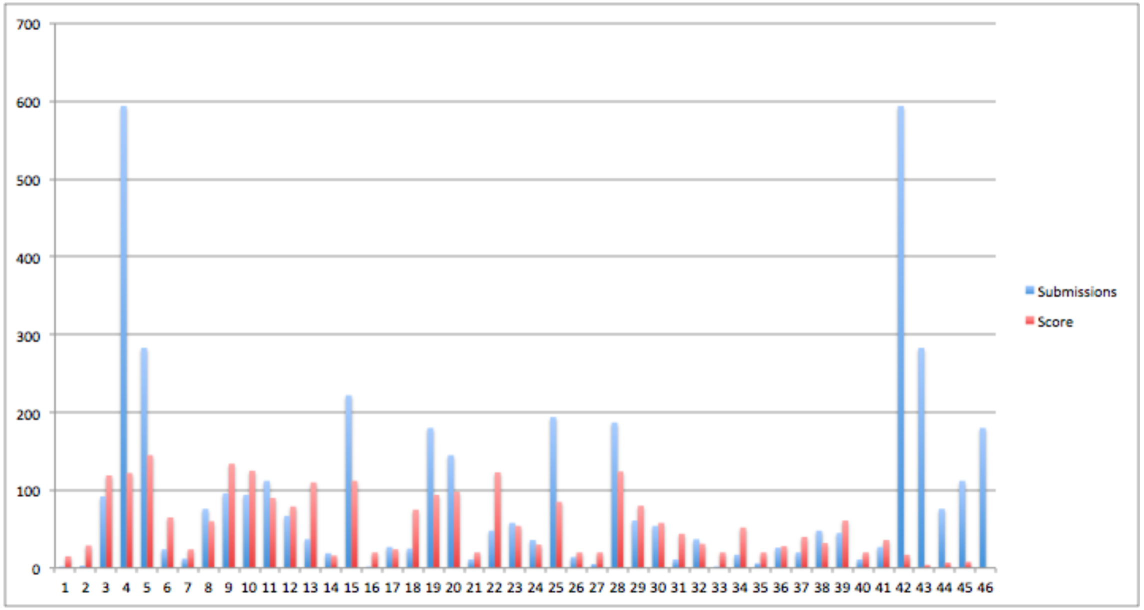


Phase 2

- New Query 3 and Query 4
    - Make sure you read the write up properly.
- Phase2 is Live
    - One team (z2m) has 100% correctness for Q3 and Q4.
- Team Oak

  636 submissions

  till now.



What’s due soon?

- Phase 2 Deadline
    - **Submission by 16.59 ET (Pittsburgh) Wed 4/1**
        - **HBase Live from 6 PM to 9 PM ET**
        - **MySQL Live from 9 PM to midnight ET**
    - Fix Q1 and Q2 if your Phase 1 did not go well
    - New queries Q3 and Q4. Targets 10000 and 6000 rps
    - Heads up: Phase 2 counts for 30% of 15619 grade

- Report at the end of Phase 2
    - Make sure you highlight failures and learning
    - If you didn’t do well, explain why
    - If you did, explain how

What to watch out for in Phase 2...

- Two more queries (Q3 and Q4)
  - More ETL
  - Multiple tables and queries

- **Live Test!!!**
    - **For HBase and MySQL**
    - **Includes Mixed-Load**
    - **No more pre-caching of known requests**

Query 3: Retweet Buddies

Q. What’s a retweet and how do I find it?

Read https://dev.twitter.com/docs/platform-objects/tweets

Query 3: Retweet Buddies

- A retweeted B twice
- B retweeted A once
- C retweeted A once
- A retweeted D once

GET /q3?userid=A

- *,3, B
- +,1, C
- -,1, D

Query 4: Trending Hashtag

- Use the hashtag entity

  GET /q4?hashtag=SamSmith&start=2014-06-23&end=2014-06-24

```
...
481298397299630080,57299114,2014-06-24+04:50:54
...
```

Query 4: Trending Hashtags (how it fits in)

- GET /q2?userid=57299114&tweet_time2014-06-24+04:50:54
- 481298397299630080:0:tapi gak papa deh, doi Taurus juga #SamSmith

Tips for Phase 2

- Avoid Tagging Penalties
- Keep a watch on budget. $60 phase + livetest
- Preparing for the live test
  - You are required to submit two DNS each for MySQL and HBase for the live test
  - Budget limited to $1.75/hr for MySQL and HBase web service separately.
  - Caching known requests will not work
  - Need to have all Qs running at the same time
  - Dont expect testing in sequence.
    - Queries will be mixed.

Thank You

Any Questions?