Outline

- Abstract
- Reference Paper Summary
- Research Questions
- Dataset
- Architectures
- Experiments and Results
- Non Functional Requirements
- Conclusion
- Future Scope
- References

Abstract

- For the research implementation, we have found a large dataset having various columns. Based on the big data, we tried to execute a query for a dataset using two different methods and compared the time constraint for the same. First, we executed the query on PostgreSQL and another one using the Hadoop cluster and map-reduce technique.

- Presently, seeing the current trend, cloud technology is being used in order to manage and process the data and it works efficiently than the typical offline or manual based systems.

- As a part of the result, we tried to calculate the speedup factor which shows that distributed execution is much faster than linear execution. Along with time, many factors such as cost, scalability, reliability, etc need to be considered.

Reference Paper Summary

- In the reference paper, they have discussed that enterprises have employee strength nearly about 42000, out of which more than 7000 employees are from overseas. So they were facing a situation in which the company's parent branch did not have a feasible tool to collect data from the child branches.

- An example of a manual 3 level leave approval system and related issues shown in the paper.

- Data flow between local system, cloud and Hadoop ecosystem.

- Massive data processing is conducted by the MapReduce framework in cloud computing data center.

Research Questions

- RQ1 : How e-HR system is more efficient than the manual HR system?
- RQ2 : Why Distributed database is better than local database for large amount of data?
- RQ3 : What is the impact of parallel processing on the execution of complex queries for Big data?

Dataset

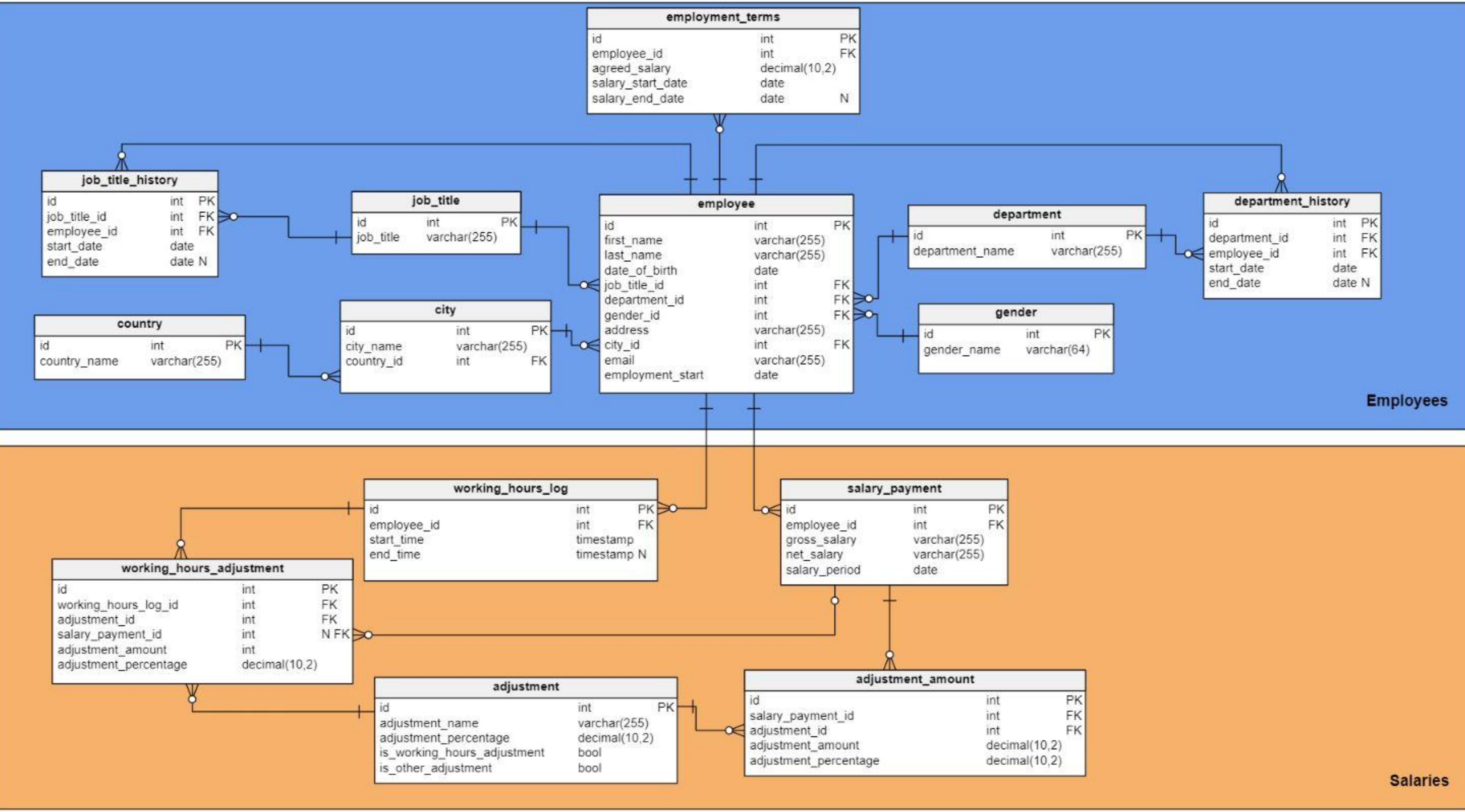
Not large enough dataset present for HR department.

So we have used Rate.csv from the **Health Insurance Marketplace** dataset which consist of around 12800000 rows and 24 columns (size : approximate 1.8GB).

Columns include BusinessYear, IssuerId, Age, VersionNumber, etc.

The csv file is uploaded to hive table as well as pgadmin for database table creation.

Operation	Time taken by PostgreSQL	Time taken by MapReduce
Loading the data from CSV	68.67 seconds	2.4 seconds



employment_terms		
id	int	PK
employee_id	int	FK
agreed_salary	decimal(10,2)	
salary_start_date	date	
salary_end_date	date	N

job_title_history		
id	int	PK
job_title_id	int	FK
employee_id	int	FK
start_date	date	
end_date	date	N

job_title		
id	int	PK
job_title	varchar(255)	

employee		
id	int	PK
first_name	varchar(255)	
last_name	varchar(255)	
date_of_birth	date	
job_title_id	int	FK
department_id	int	FK
gender_id	int	FK
address	varchar(255)	
city_id	int	FK
email	varchar(255)	
employment_start	date	

department		
id	int	PK
department_name	varchar(255)	

department_history		
id	int	PK
department_id	int	FK
employee_id	int	FK
start_date	date	
end_date	date	N

gender		
id	int	PK
gender_name	varchar(64)	

country		
id	int	PK
country_name	varchar(255)	

city		
id	int	PK
city_name	varchar(255)	
country_id	int	FK

Employees

working_hours_log		
id	int	PK
employee_id	int	FK
start_time	timestamp	
end_time	timestamp	N

salary_payment		
id	int	PK
employee_id	int	FK
gross_salary	varchar(255)	
net_salary	varchar(255)	
salary_period	date	

working_hours_adjustment		
id	int	PK
working_hours_log_id	int	FK
adjustment_id	int	FK
salary_payment_id	int	N FK
adjustment_amount	int	
adjustment_percentage	decimal(10,2)	

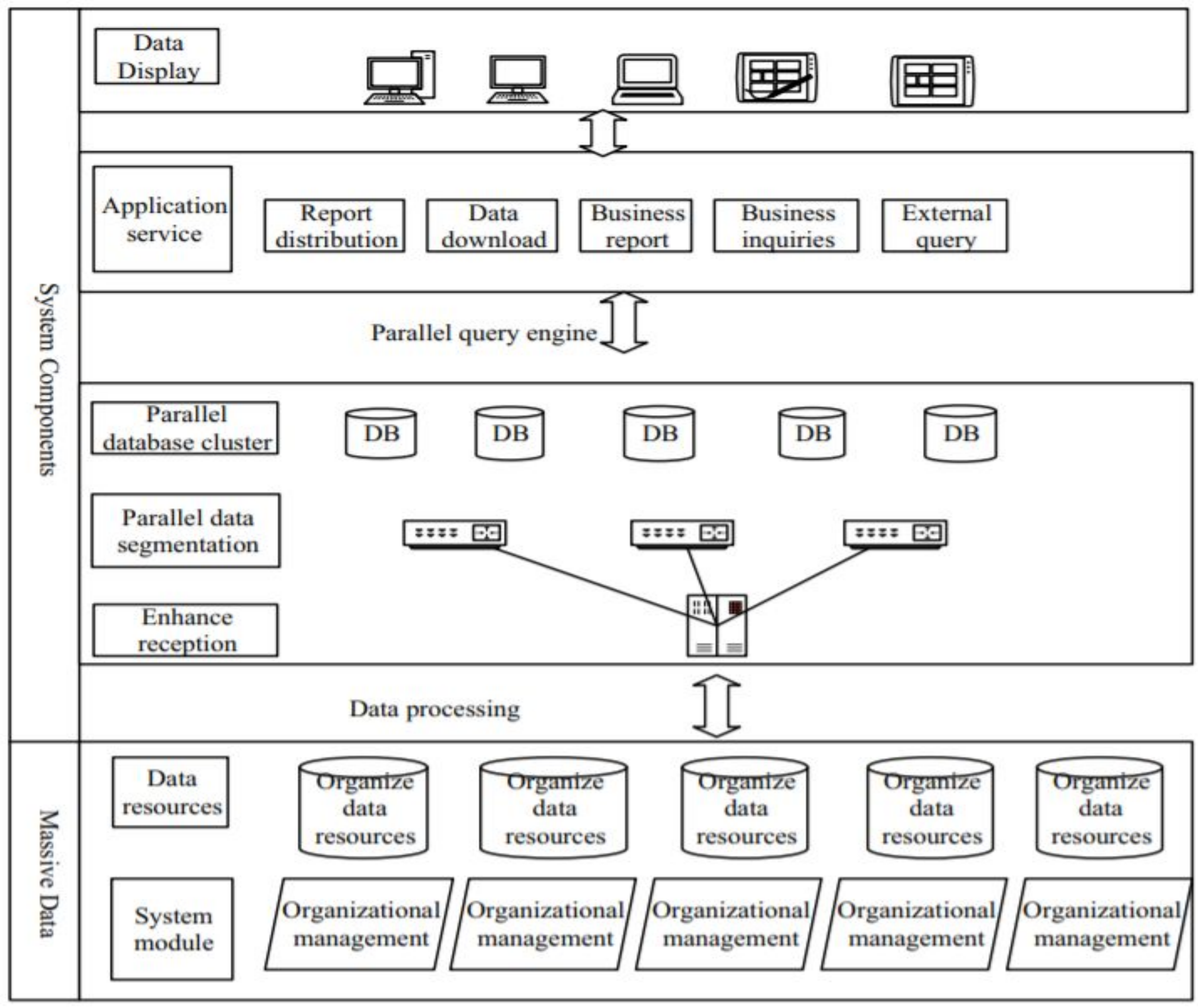
adjustment		
id	int	PK
adjustment_name	varchar(255)	
adjustment_percentage	decimal(10,2)	
is_working_hours_adjustment	bool	
is_other_adjustment	bool	

adjustment_amount		
id	int	PK
salary_payment_id	int	FK
adjustment_id	int	FK
adjustment_amount	decimal(10,2)	
adjustment_percentage	decimal(10,2)	

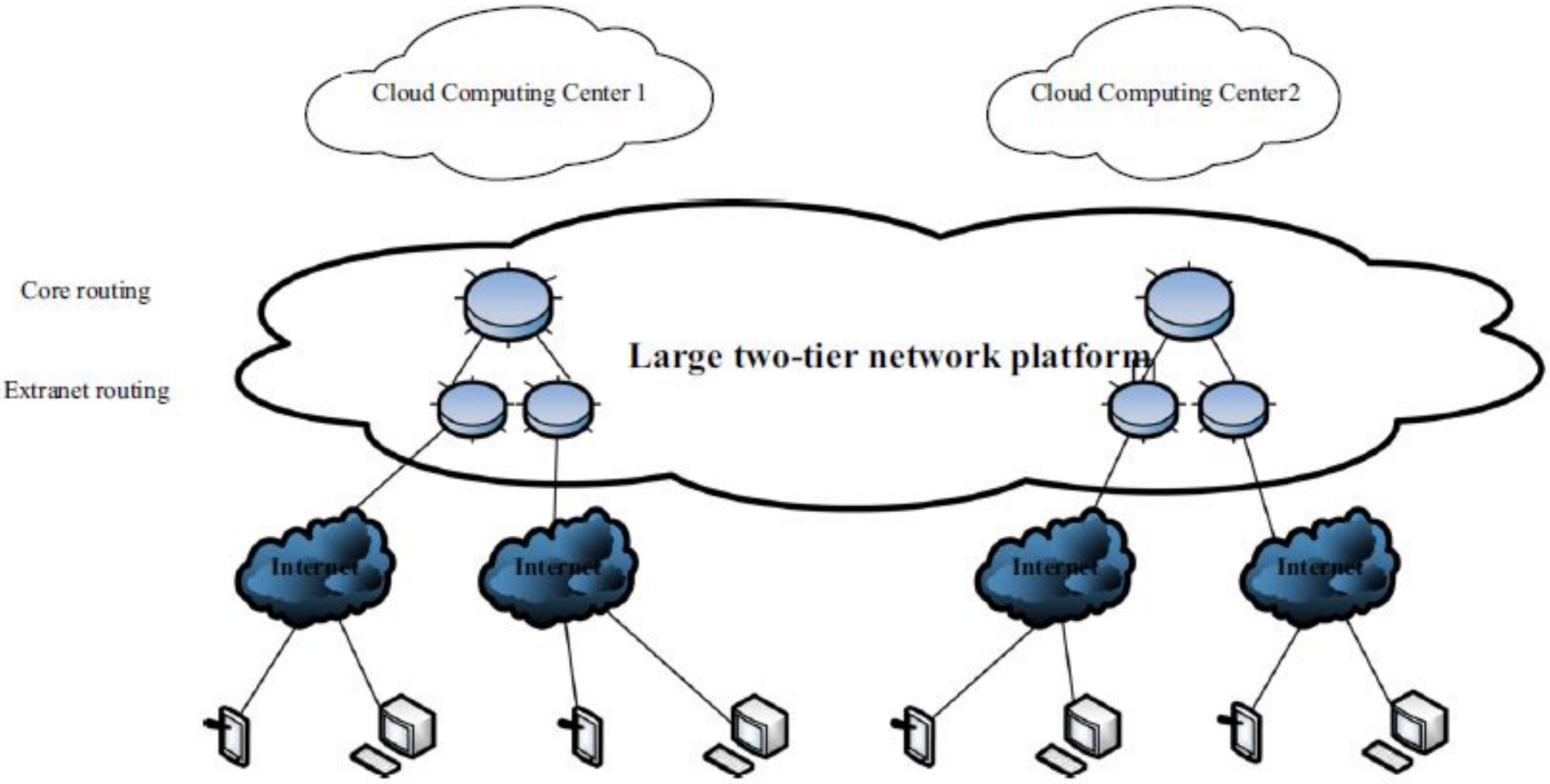
Salaries

HRMS System Architecture

Full architecture with backend and frontend



Cloud Two Tier Network Platform Architecture

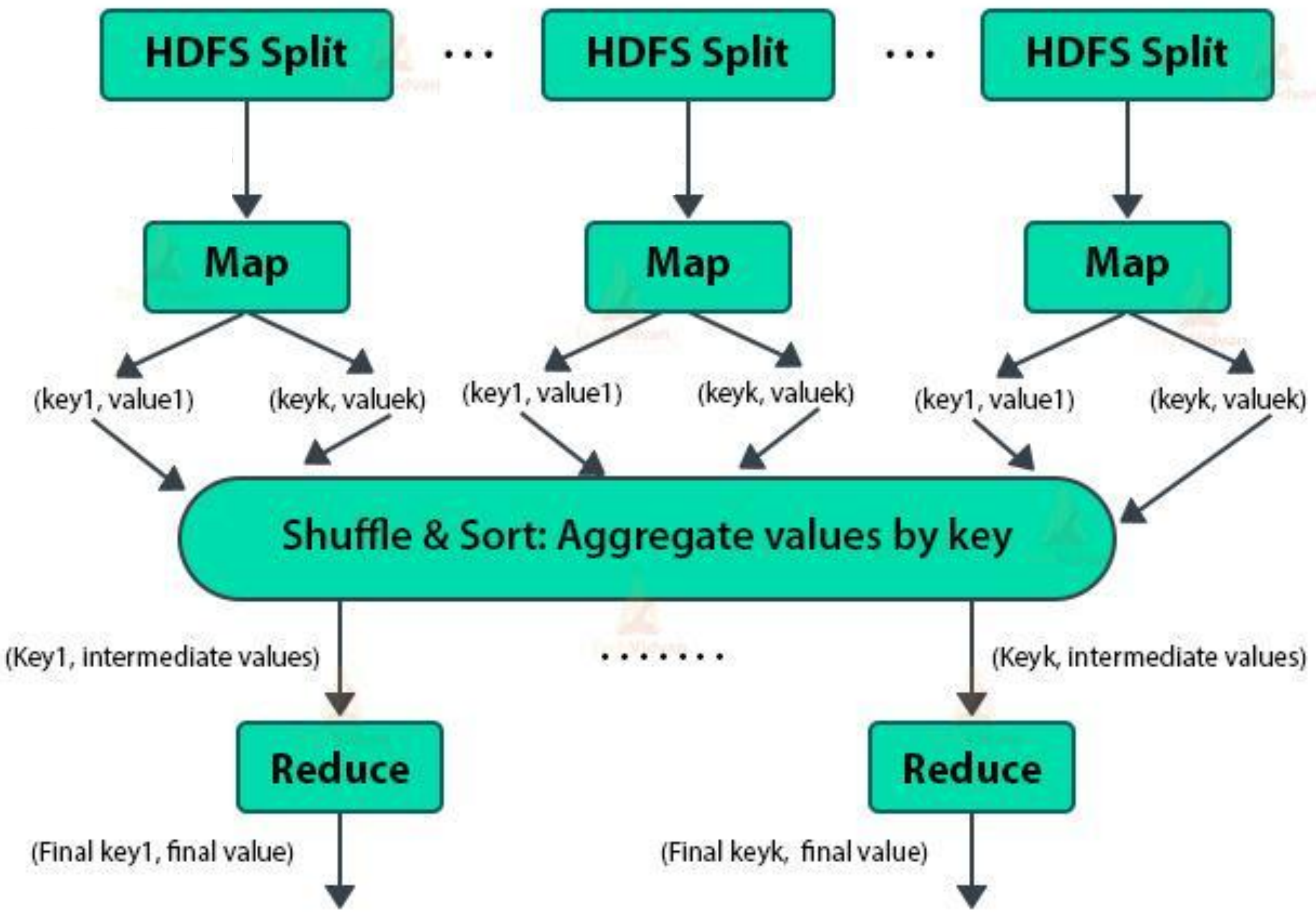


Hadoop Cluster

Data nodes as well as task nodes are connected to each other

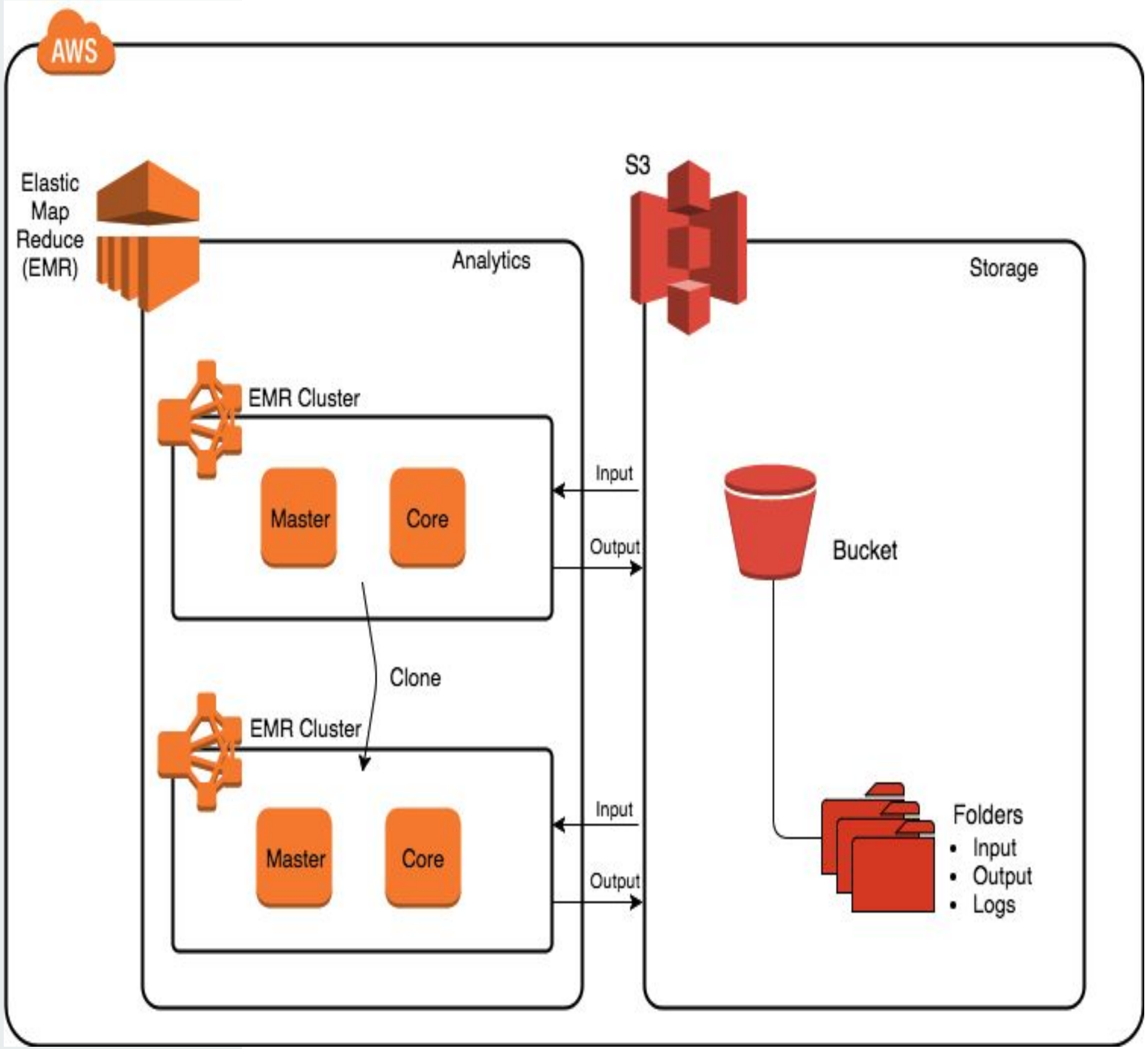
Cluster configuration :
1 master node and 7 data nodes

Hadoop MapReduce

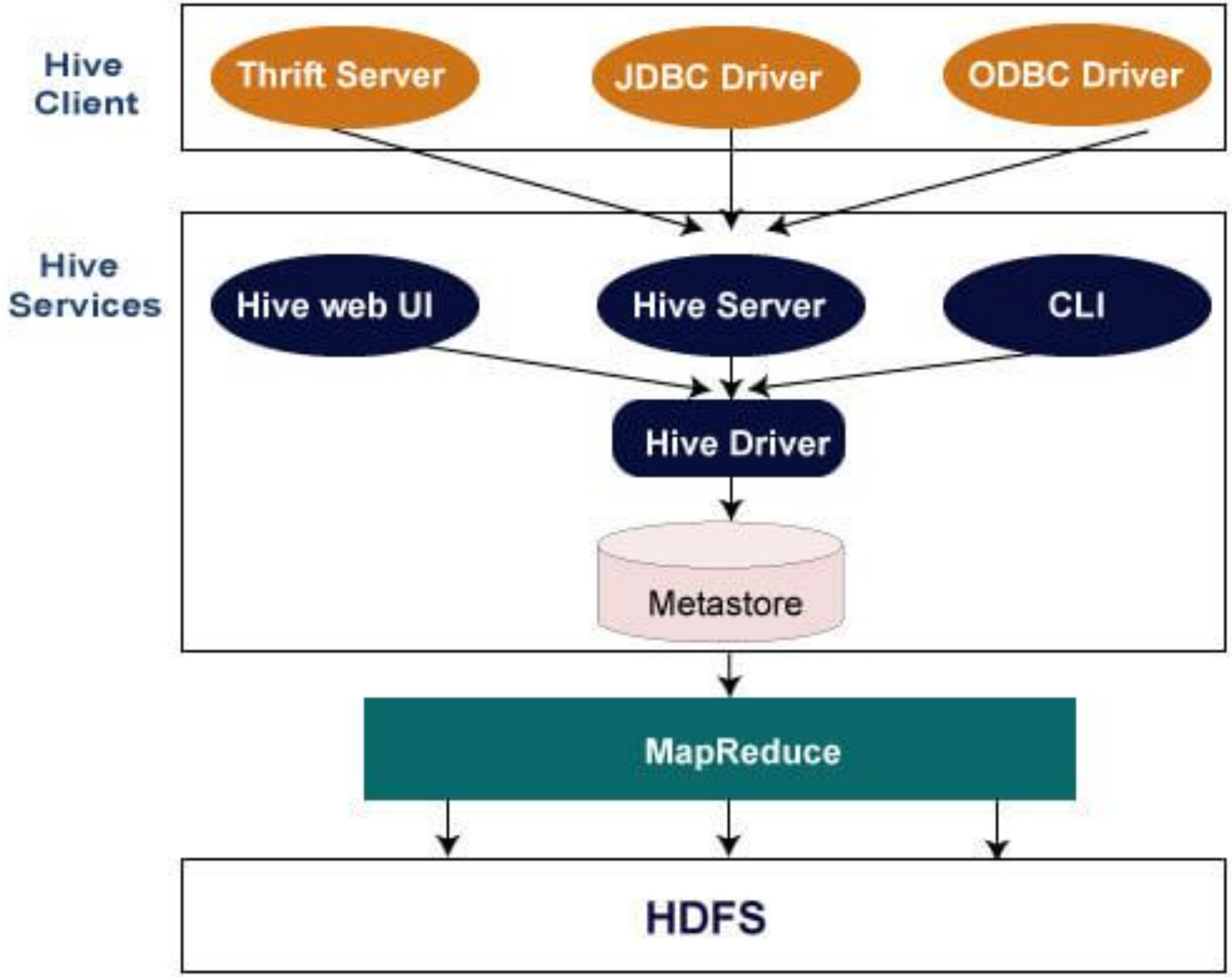


EMR Architecture

- **Data cannot be directly inserted into Hive, but can be inserted using S3 service.**
- **All config files of EMR are stored in S3 buckets.**

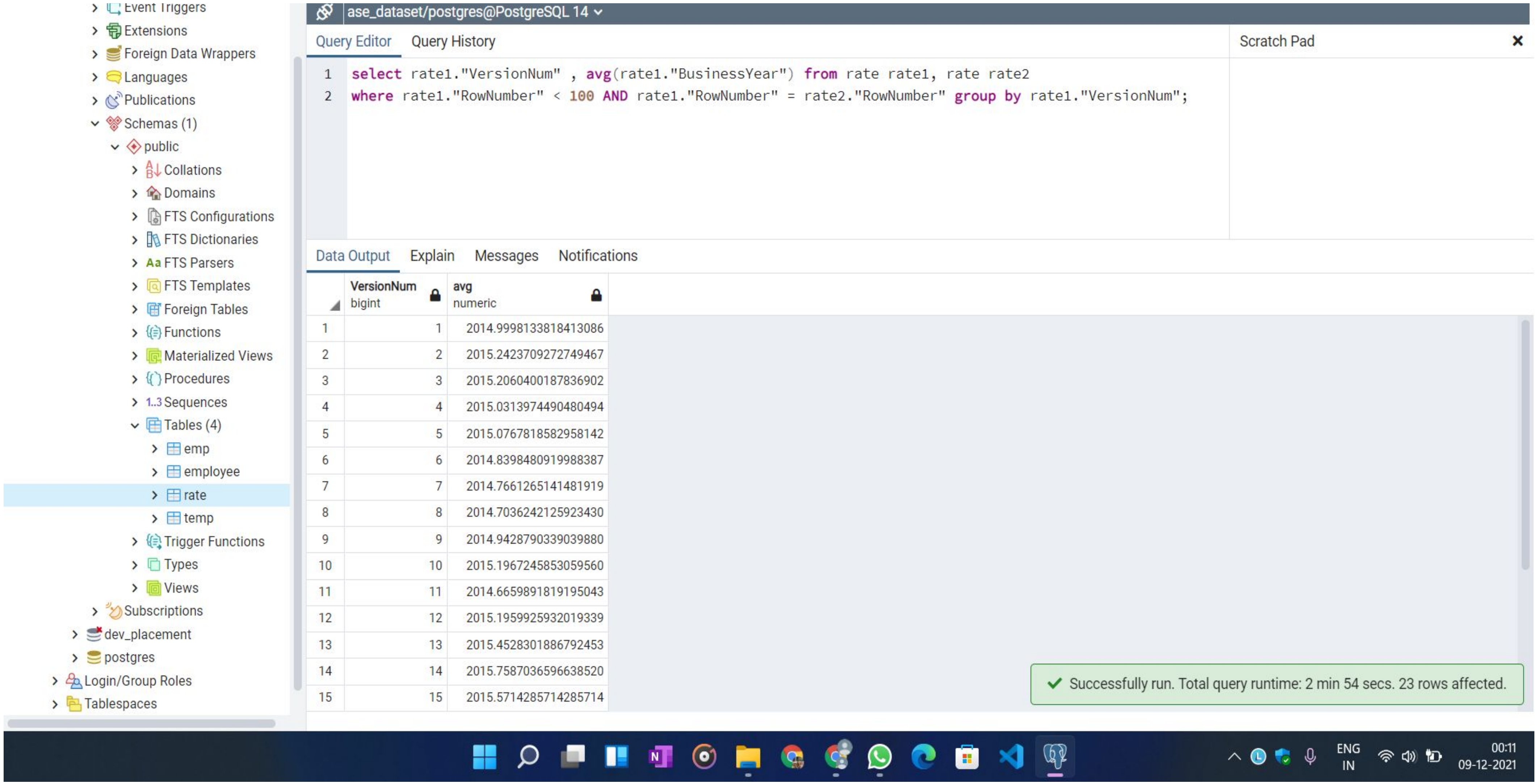


Apache Hive Architecture



Experiments and Results

Linear Execution Example 1

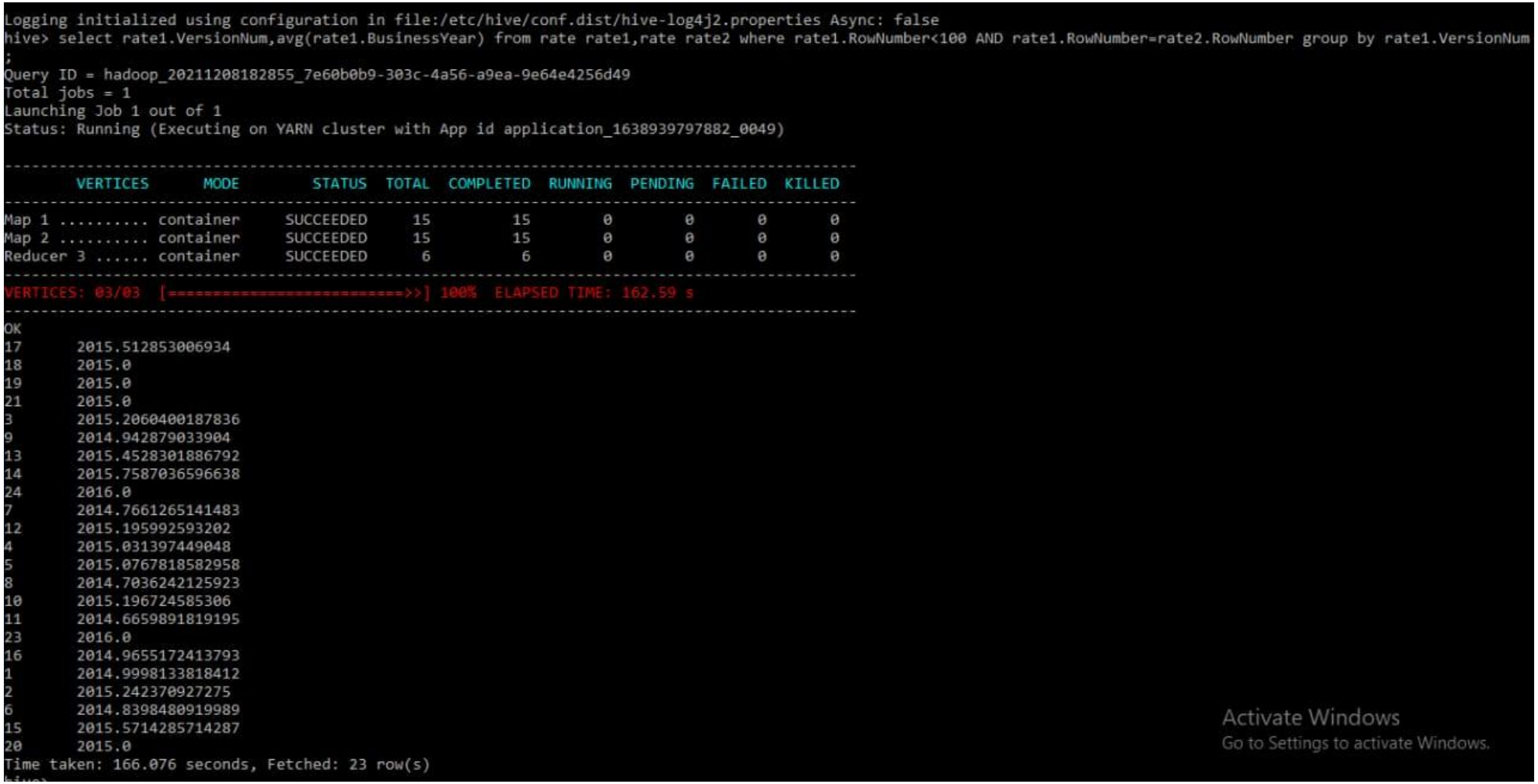


```sql
select rate1."VersionNum" , avg(rate1."BusinessYear") from rate rate1, rate rate2
where rate1."RowNumber" < 100 AND rate1."RowNumber" = rate2."RowNumber" group by rate1."VersionNum";
```

	VersionNum bigint	avg numeric
1	1	2014.9998133818413086
2	2	2015.2423709272749467
3	3	2015.2060400187836902
4	4	2015.0313974490480494
5	5	2015.0767818582958142
6	6	2014.8398480919988387
7	7	2014.7661265141481919
8	8	2014.7036242125923430
9	9	2014.9428790339039880
10	10	2015.1967245853059560
11	11	2014.6659891819195043
12	12	2015.1959925932019339
13	13	2015.4528301886792453
14	14	2015.7587036596638520
15	15	2015.5714285714285714

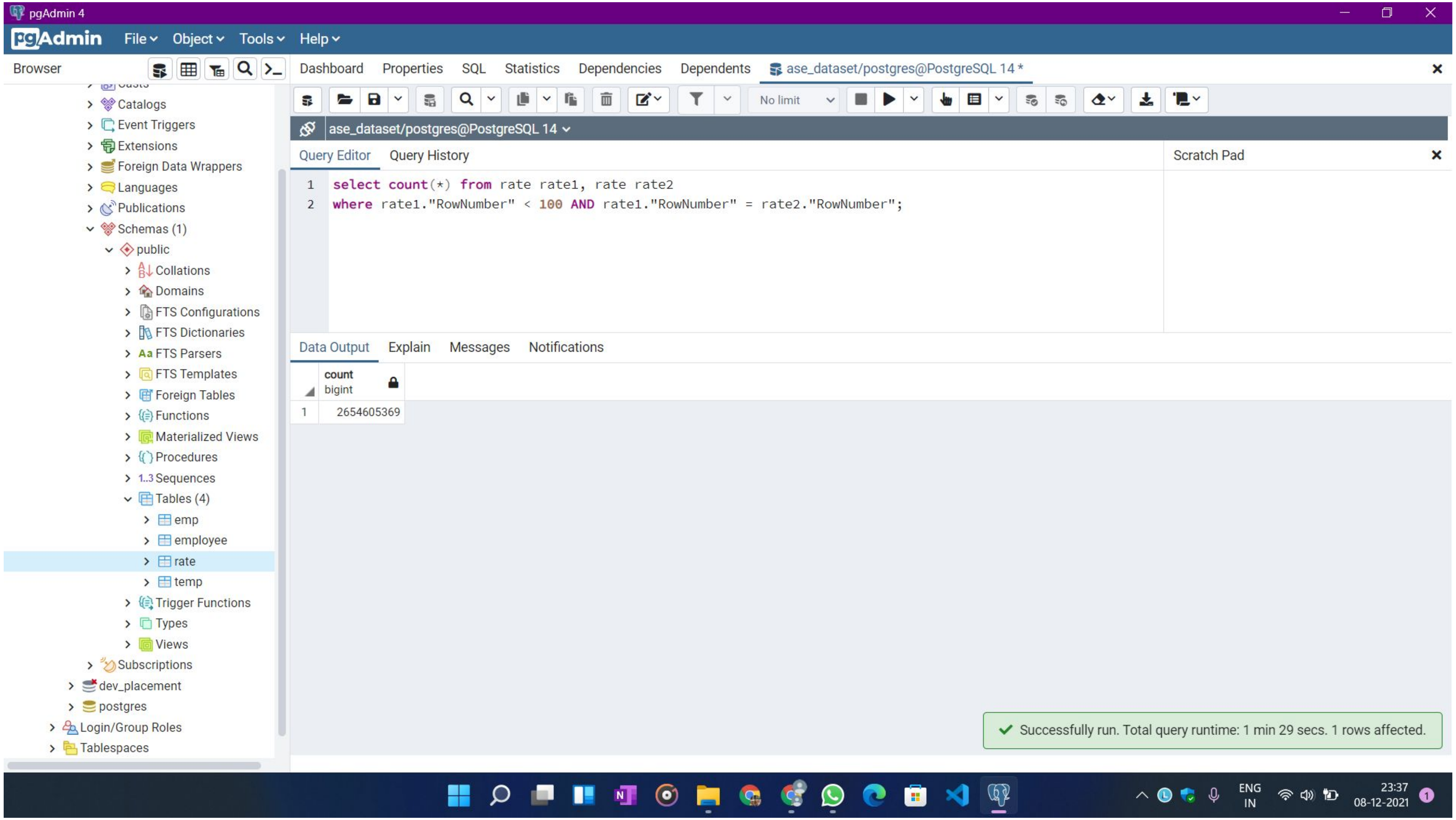
✓ Successfully run. Total query runtime: 2 min 54 secs. 23 rows affected.

Distributed Execution Example 1



```
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> select rate1.VersionNum,avg(rate1.BusinessYear) from rate rate1,rate rate2 where rate1.RowNumber<100 AND rate1.RowNumber=rate2.RowNumber group by rate1.VersionNum
;
Query ID = hadoop_20211208182855_7e60b0b9-303c-4a56-a9ea-9e64e4256d49
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1638939797882_0049)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     15         15        0        0       0       0
Map 2 .......... container     SUCCEEDED     15         15        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      6          6        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 162.59 s
----------------------------------------------------------------------------------------------
OK
17      2015.512853006934
18      2015.0
19      2015.0
21      2015.0
3       2015.2060400187836
9       2014.942879033904
13      2015.4528301886792
14      2015.7587036596638
24      2016.0
7       2014.7661265141483
12      2015.195992593202
4       2015.031397449048
5       2015.0767818582958
8       2014.7036242125923
10      2015.196724585306
11      2014.6659891819195
23      2016.0
16      2014.9655172413793
1       2014.9998133818412
2       2015.242370927275
6       2014.8398480919989
15      2015.5714285714287
20      2015.0
Time taken: 166.076 seconds, Fetched: 23 row(s)
hive>
```

Linear Execution Example 2

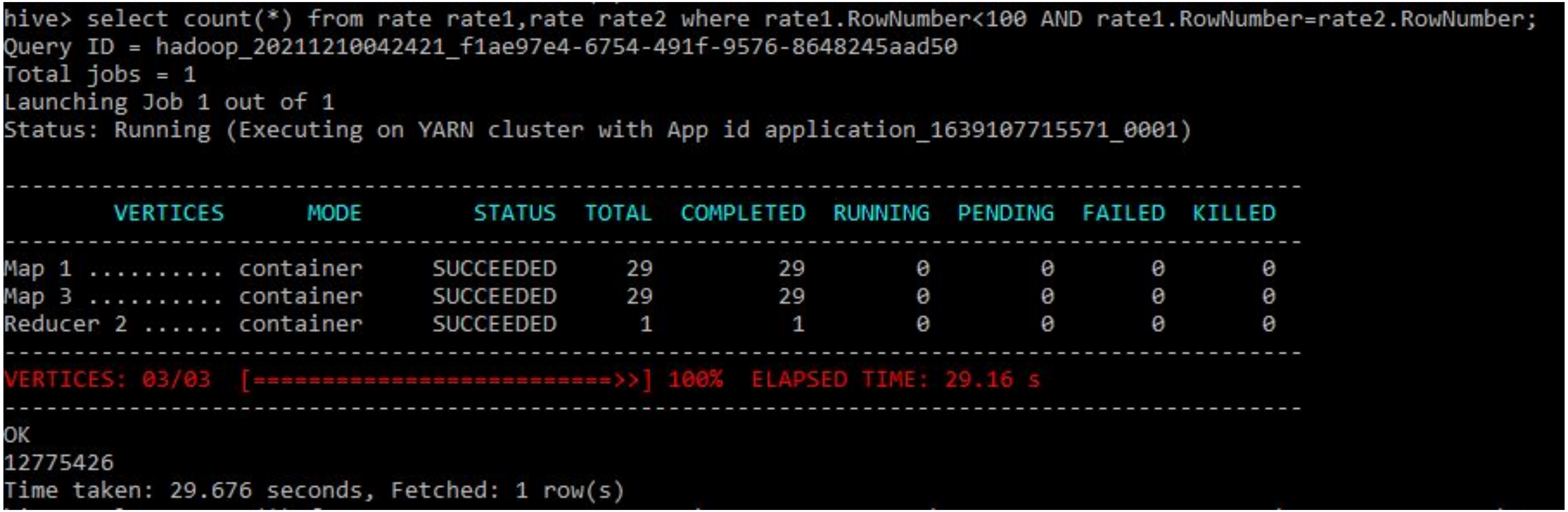


```
select count(*) from rate rate1, rate rate2
where rate1."RowNumber" < 100 AND rate1."RowNumber" = rate2."RowNumber";
```

	count bigint
1	2654605369

✓ Successfully run. Total query runtime: 1 min 29 secs. 1 rows affected.

Distributed Execution Example 2



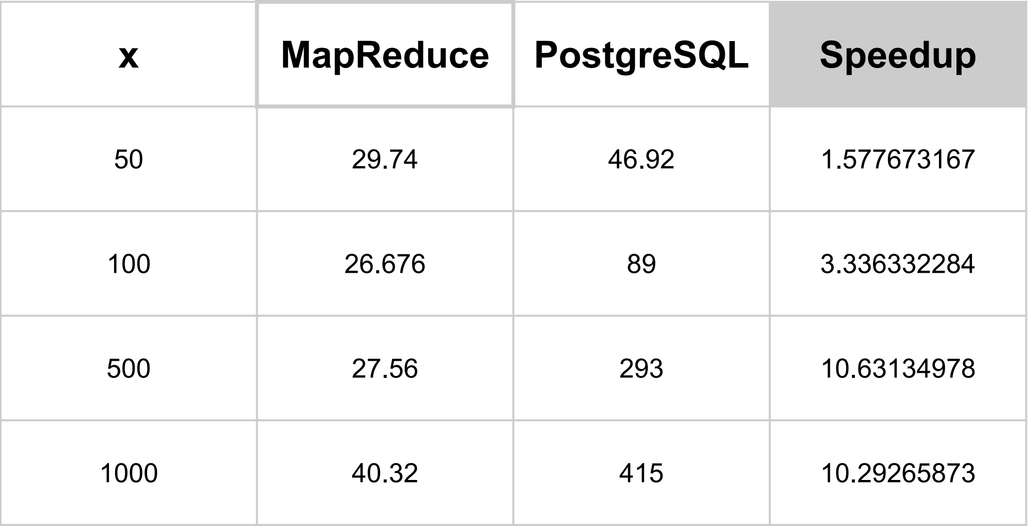
```
hive> select count(*) from rate rate1,rate rate2 where rate1.RowNumber<100 AND rate1.RowNumber=rate2.RowNumber;
Query ID = hadoop_20211210042421_f1ae97e4-6754-491f-9576-8648245aad50
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1639107715571_0001)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     29         29        0        0       0       0
Map 3 .......... container     SUCCEEDED     29         29        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 29.16 s
----------------------------------------------------------------------------------------------
OK
12775426
Time taken: 29.676 seconds, Fetched: 1 row(s)
```

Results

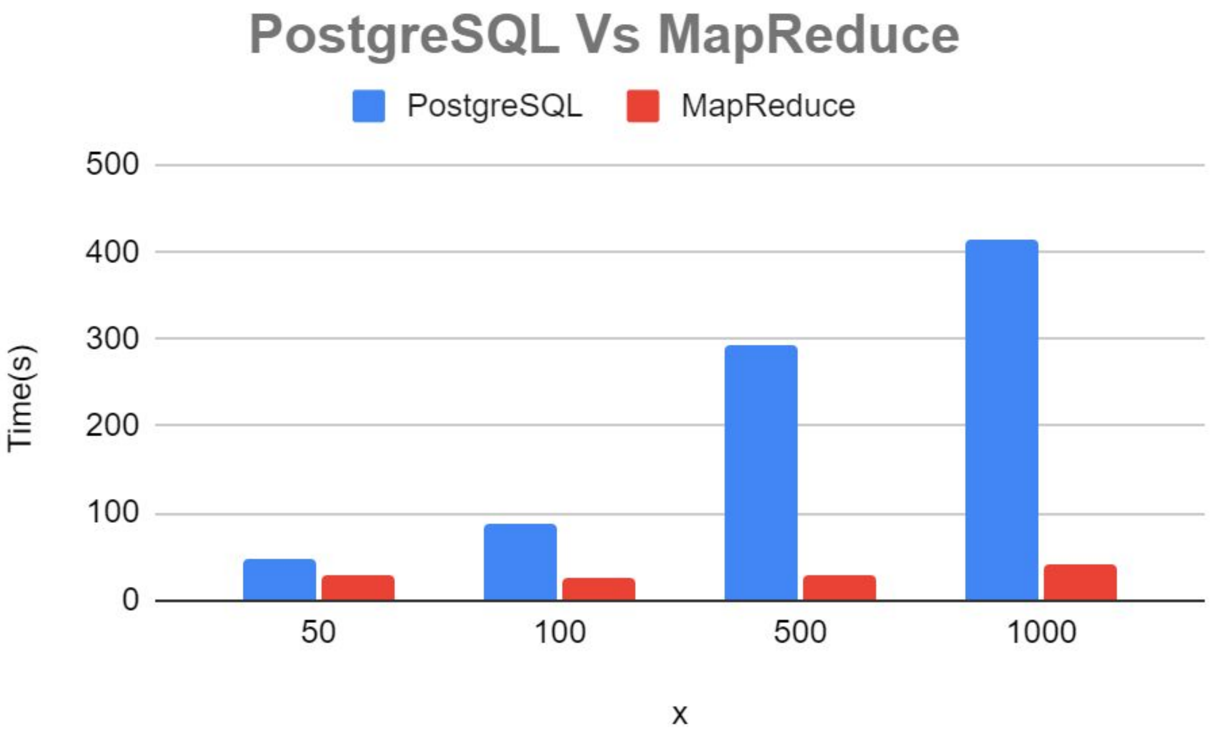
Query	Time taken by PostgreSQL T_{SQL}	Time taken by Map-Reduce $T_{MapReduce}$	Speed Up= T_{SQL} / $T_{MapReduce}$
Load data from csv	68.67s	2.399s	28.62
select rate1.VersionNum, avg(rate1.BusinessYear) from rate rate1, rate rate2 where rate1.RowNumber<100 AND rate1.RowNumber=rate2.RowNumber group by rate1.VersionNum;	174s	28s	6.21
select rate1.VersionNum, avg(rate1.BusinessYear) from rate rate1, rate rate2 where rate1.RowNumber<1000 AND rate1.RowNumber=rate2.RowNumber group by rate1.VersionNum;	803s	30s	26
select count(*) from rate rate1, rate rate2 where rate1.RowNumber<1000 AND rate1.RowNumber=rate2.RowNumber;	415s	40.32s	10.29
select count(*) from rate rate1, rate rate2 where rate1.RowNumber<100 AND rate1.RowNumber=rate2.RowNumber;	89s	59.43s	3

Query 1:

```
select count(*) from rate rate1, rate rate2 where rate1.RowNumber  <  x AND
rate1.RowNumber = rate2.RowNumber;
```



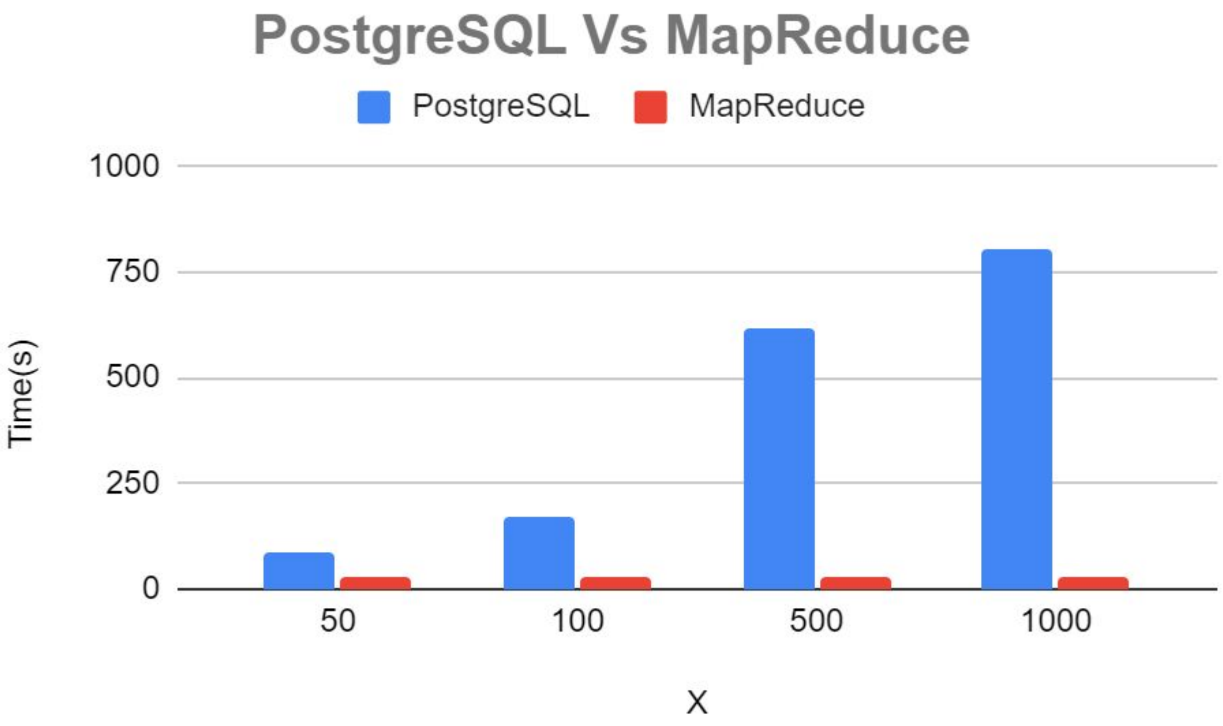
x	MapReduce	PostgreSQL	Speedup
50	29.74	46.92	1.577673167
100	26.676	89	3.336332284
500	27.56	293	10.63134978
1000	40.32	415	10.29265873



Query 2:

select rate1.VersionNum , avg(rate1.BusinessYear) from rate rate1 , rate rate2 where rate1.RowNumber < X AND rate1.RowNumber = rate2.RowNumber group by rate1.VersionNum;

X	MapReduce	PostgreSQL	Speedup
50	26.97	87	3.225806452
100	28	174	6.214285714
500	27.16	617	22.71723122
1000	30	803	26.76666667



Non Functional Requirements

- Usability
- Maintainability
- Scalability
- Availability
- Economical
- User friendly

Conclusion

- RQ1 : How e-HR system is more efficient than manual HR system?

- RQ2 : Why Distributed database is better than local database for large amount of data?

- RQ3 : What is the impact of parallel processing on the execution of complex queries for Big data?

- The proposed model has higher efficacy in dealing with HR core business. It tries to improve the query processing time for the massive data, providing benefits such as cost reduction, user friendliness, full function, and fast speed data.

Future Scope

- Use of Spark Engine
- Partitioning/ Bucketing
- Parquet Files

References

1. Lv, Z., Tan, Z., Wang, Q. *et al.* Cloud Computing Management Platform of Human Resource Based on Mobile Communication Technology. *Wireless Pers Commun* 102, 1293–1306 (2018).
2. Al-Dmour, R. H., Masa'deh, R. E., & Obeidat, B. Y. (2017). Factors influencing the adoption and implementation of HRIS applications: Are they similar? International Journal of Business Innovation and Research, 14(2), 139–167.
3. Gravina, R., Ma, C., Pace, P., Aloi, G., Russo, W., Li, W., & Fortino, G. (2016). Cloud-based activity-aaService cyber–physical framework for human activity monitoring in mobility. Future Generation Computer Systems, 75, 158–171.
4. Pop, F., & Potop-Butucaru, M. (2016). ARMCO: Advanced topics in resource management for ubiquitous cloud computing: An adaptive approach. Future Generation Computer Systems, 54, 79–81.
5. Singh, S., & Chana, I. (2016). QoS-aware autonomic resource management in cloud computing: A systematic review. ACM Computing Surveys (CSUR), 48(3), 42.
6. https://docs.aws.amazon.com/ for AWS documentation
7. Apache hive architecture image
8. EMR architecture image
9. Hadoop MapReduce image
10. HRMS database architecture

Thank You