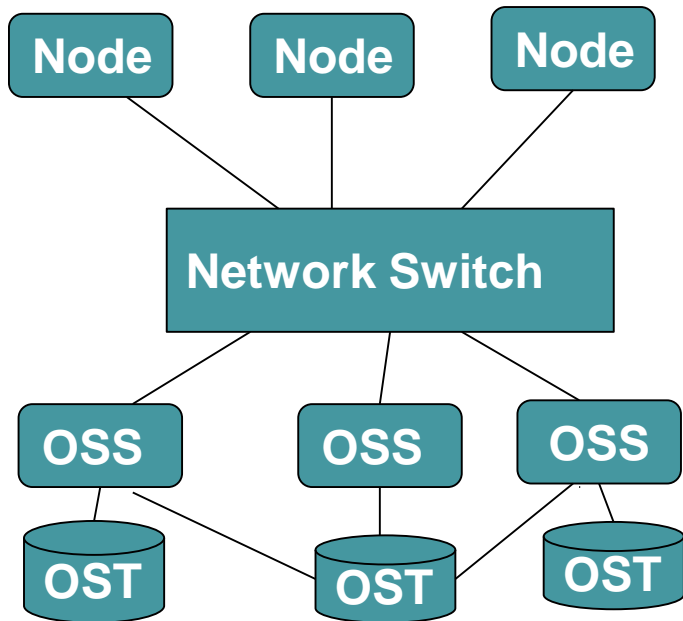# Performance Comparison of SQL based Big Data Analytics with Lustre and HDFS file systems
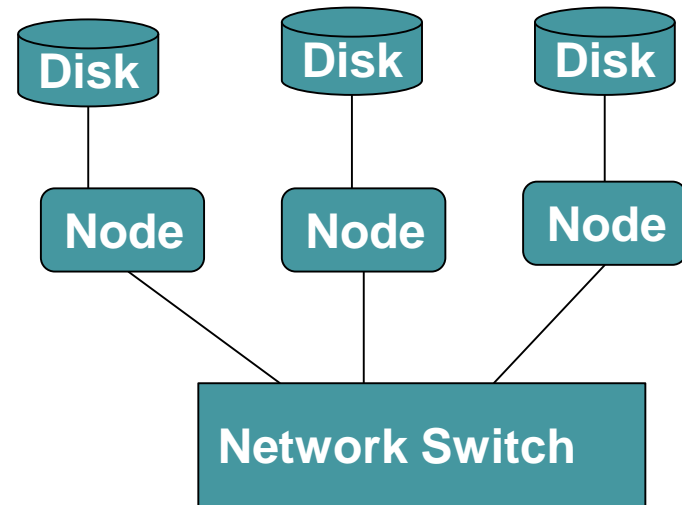
Rekha Singhal and Gabriele Pacciucci
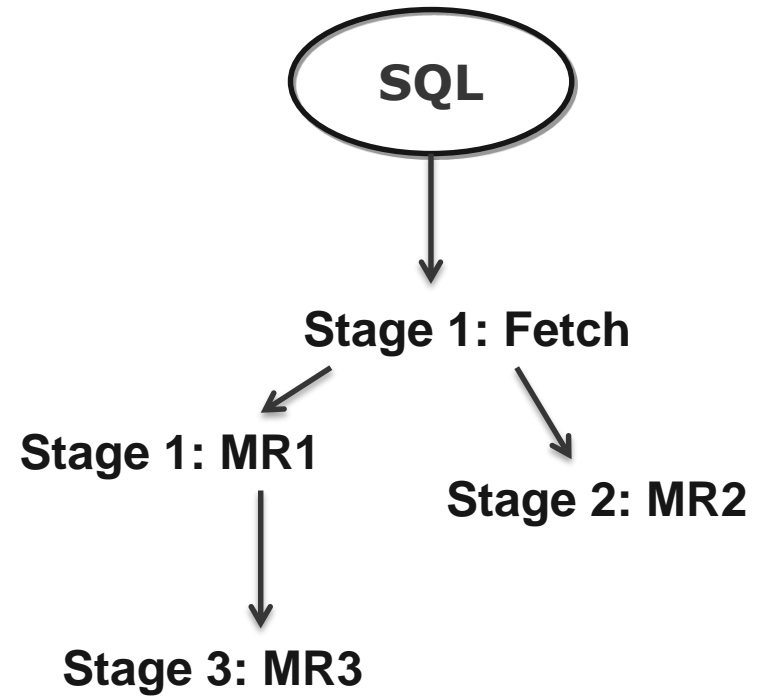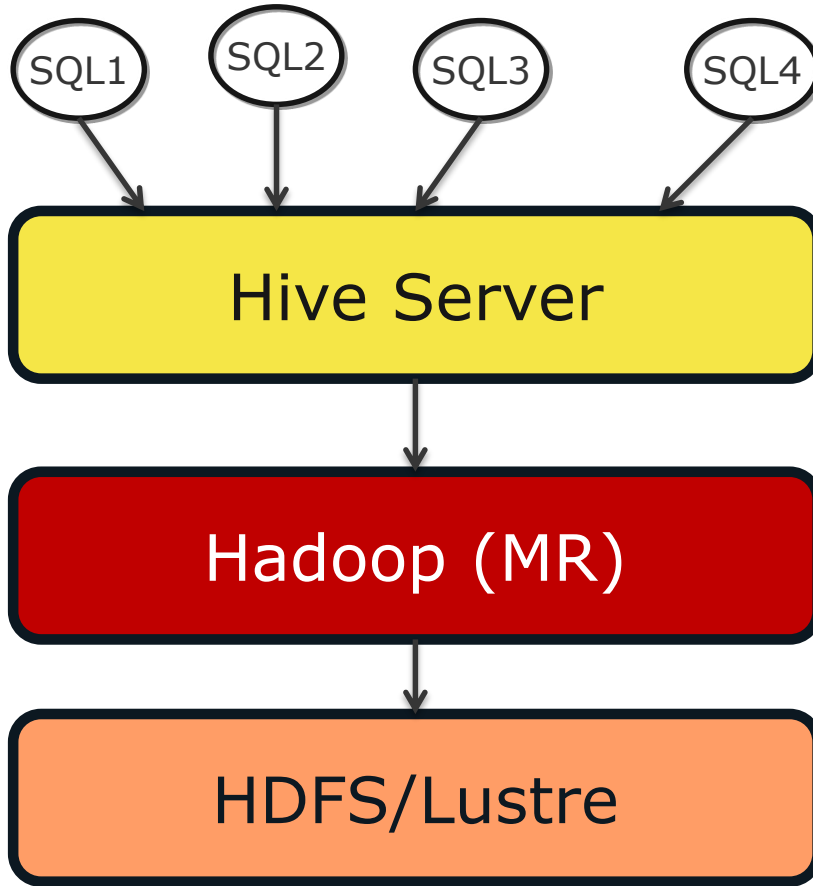
Lustre File System

- **Parallel File System**
- **No data replication**
- **No local storage**
- **Widely used for HPC applications**

Hadoop Dist. File System

- **Distributed File System**
- **Data replication**
- **Local storage**
- **Widely used for MR applications**

Hive + Hadoop Architecture

Hive+Hadoop

- Open source SQL on MapReduce framework for data-intensive computing

- Hive translates SQL into stages of MR jobs

- A MR job – two functions: Map and Reduce

- Map: Transforms input into a list of key value pairs
  - Map(D) → List[Ki , Vi]

- Reduce: Given a key and all associated values, produces result in the form of a list of values
  - Reduce(Ki , List[Vi]) → List[Vo]

- Parallelism hidden by framework
  - Highly scalable: can be applied to large datasets (Big Data) and run on commodity clusters

- Comes with its own user-space distributed file system (HDFS) based on the local storage of cluster nodes

# MR Processing in Intel® EE for Lustre* and HDFS
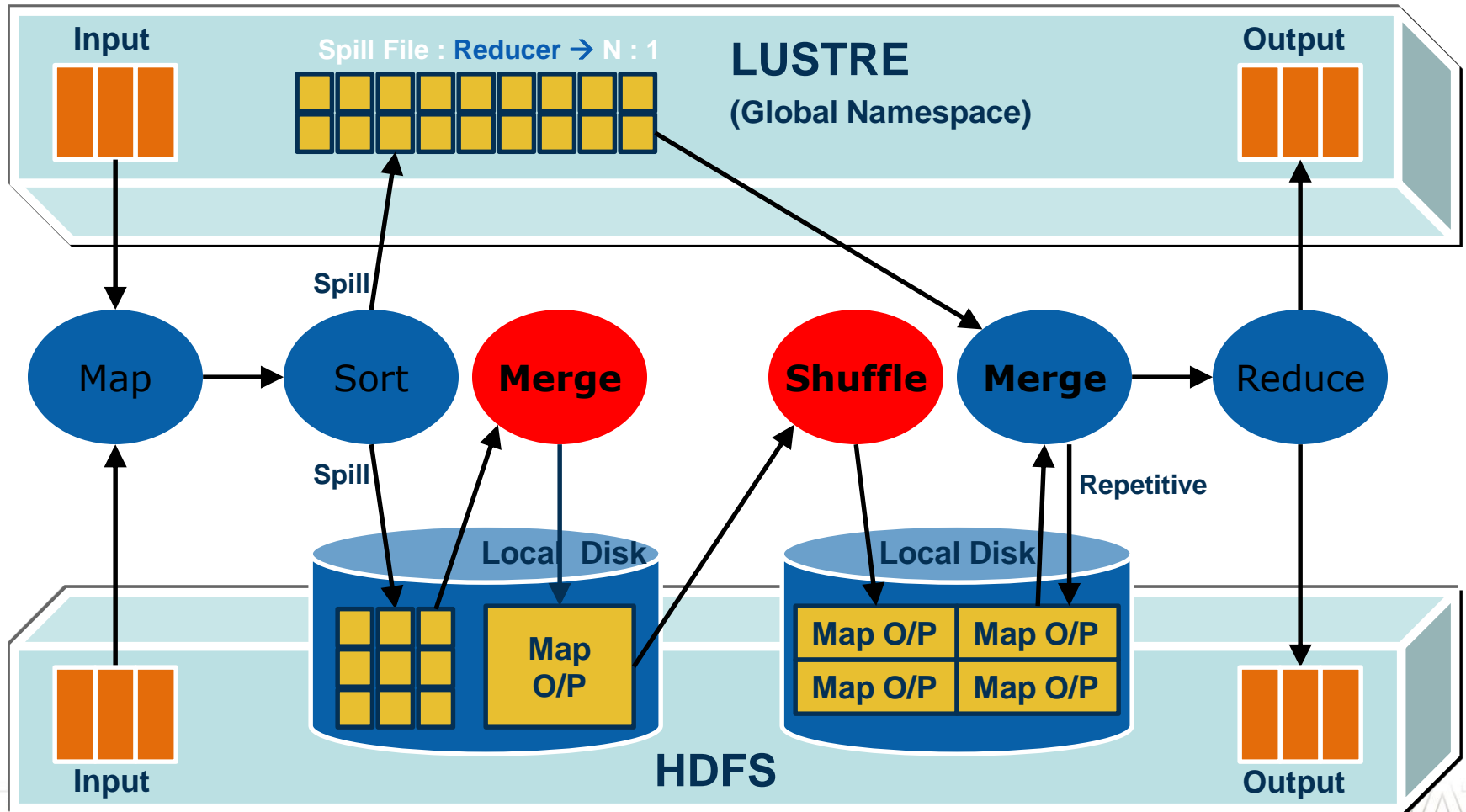
Motivation

❑ Could HPC and Analytic Computations co-exist?

  ▪ required to reduce simulations for HPC applications

❑ Need to evaluate use of alternative file systems for Big Data Analytic applications

  ▪ HDFS is an expensive distributed file system

Using Intel® Enterprise Edition for Lustre* software with Hadoop

HADOOP 'ADAPTER' FOR LUSTRE

Hadoop over Intel EE for Lustre* Implementation

- Hadoop uses pluggable extensions to work with different file system types

- Lustre is POSIX compliant:
  - Use Hadoop's built-in LocalFileSystem class
  - Uses native file system support in Java

- Extend and override default behavior: LustreFileSystem
  - Defines new URL scheme for Lustre – lustre:///
  - Controls Lustre striping info
  - Resolves absolute paths to user-defined directory
  - Leaves room for future enhancements

- Allow Hadoop to find it in config files

org.apache.hadoop.fs

FileSystem

RawLocalFileSystem

LustreFileSystem
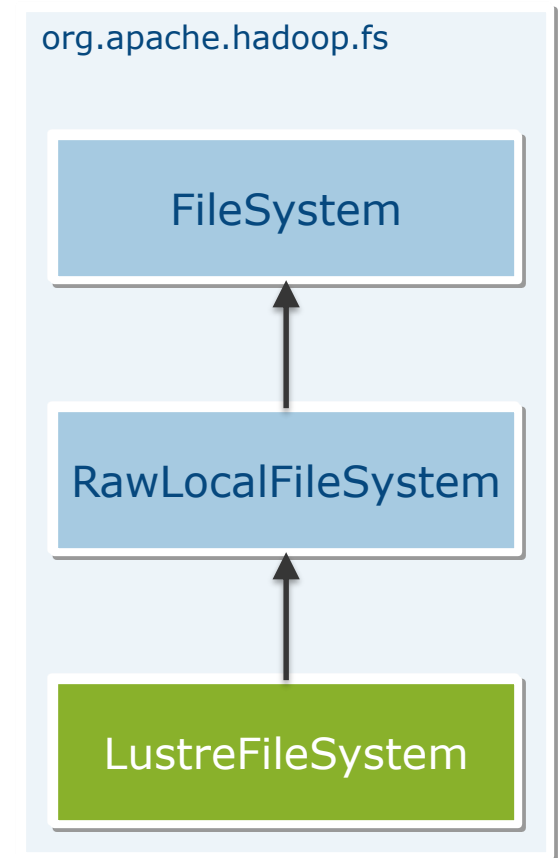
(intel)

Problem Definition

Performance comparison of LUSTRE and HDFS for SQL Analytic queries of **FSI, Insurance and Telecom** workload

on16 nodes HDDP cluster hosted in the Intel BigData Lab in Swindon (UK) and Intel® Enterprise Edition for Lustre* software
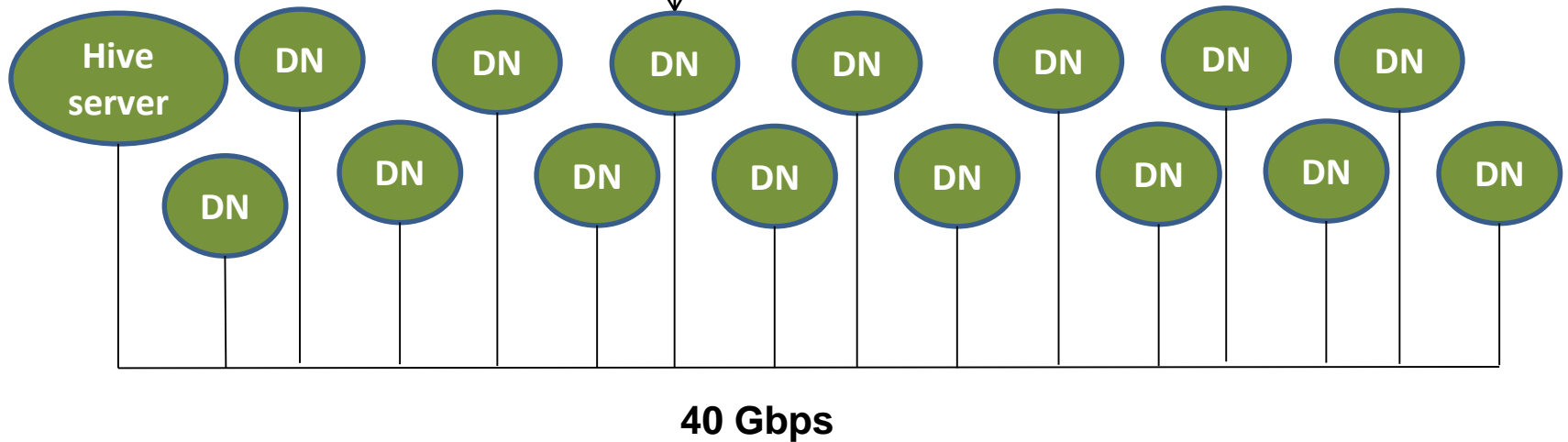
Performance metric : SQL Query Average Execution Time

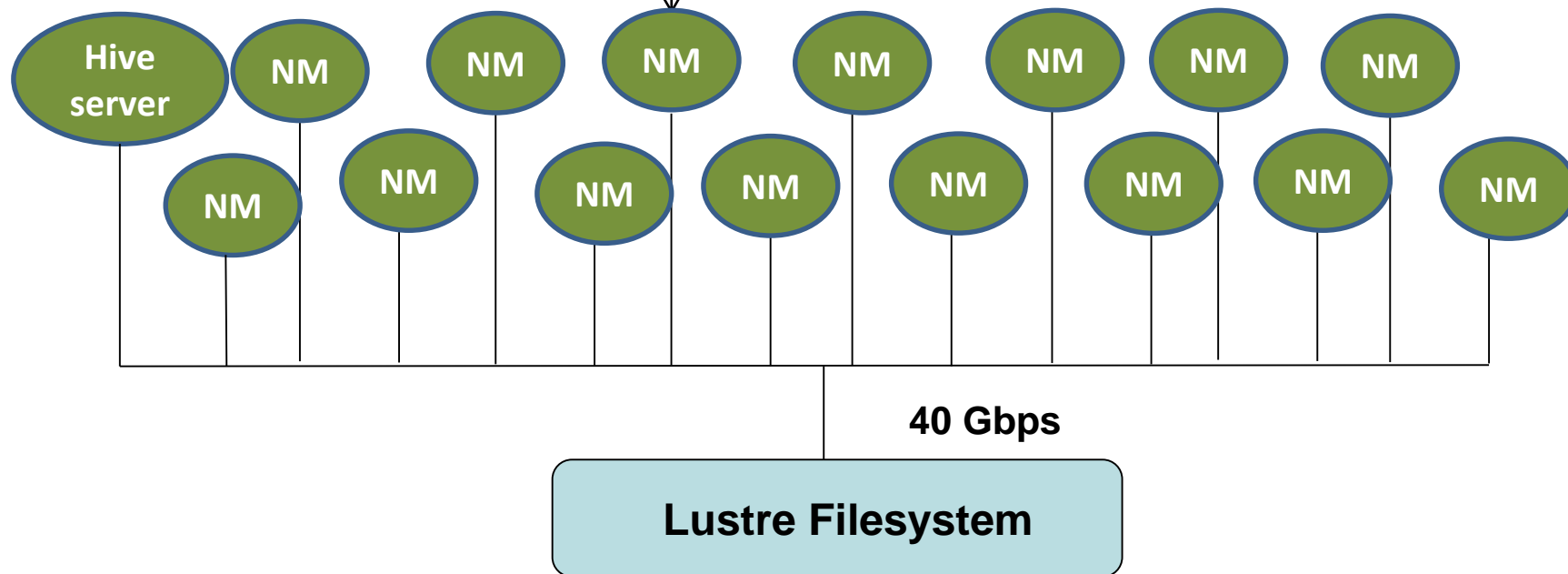# EXPERIMENTAL SETUP

Hive+Hadoop+ HDFS Setup

Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz,
320GB cluster RAM, 1 TB SATA 7200 RPM

40 Gbps

Redhat 6.5, CDH 5.2, Hive 0.13

Hive+ Hadoop+Lustre Setup

Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz,
320GB cluster RAM, 1 TB SATA 7200 RPM
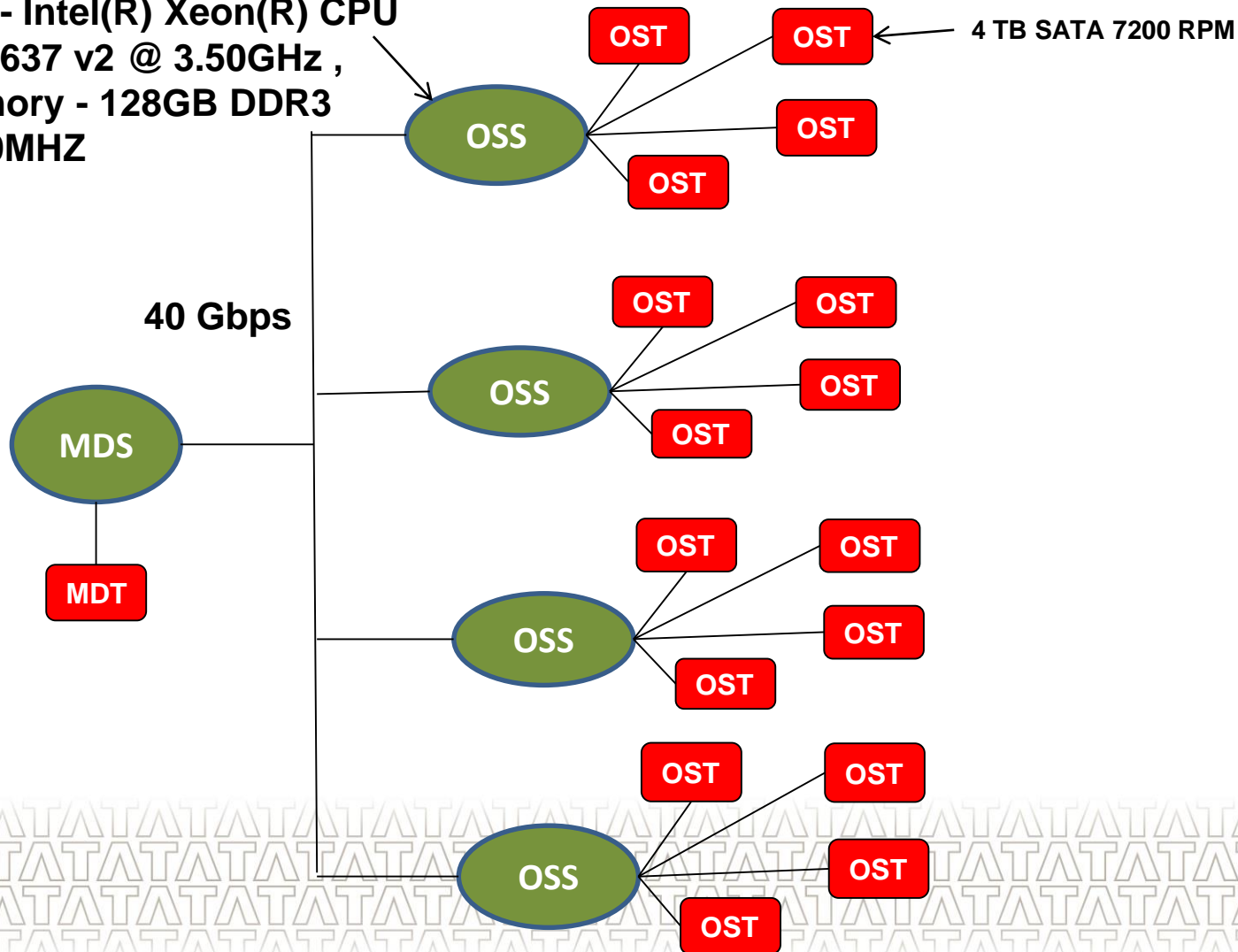

40 Gbps

Lustre Filesystem

Redhat 6.5, Hive 0.13, CDH 5.2, Intel® Enterprise Edition for Lustre* software 2.2, HAL 3.1
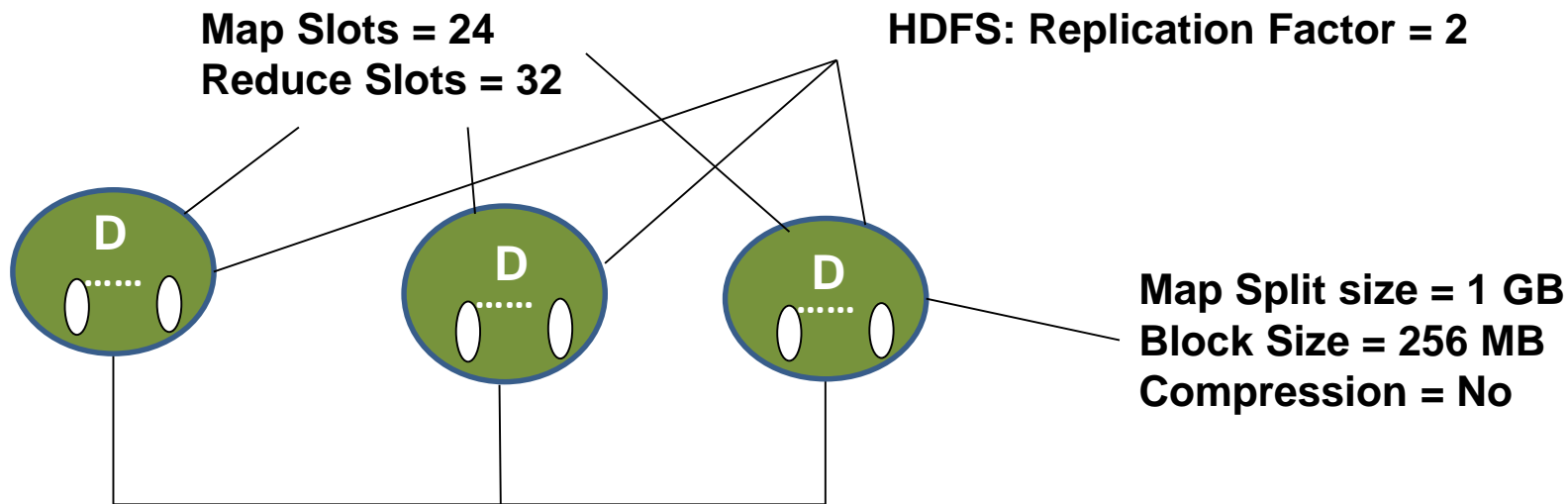
165 TB of usable cluster storage

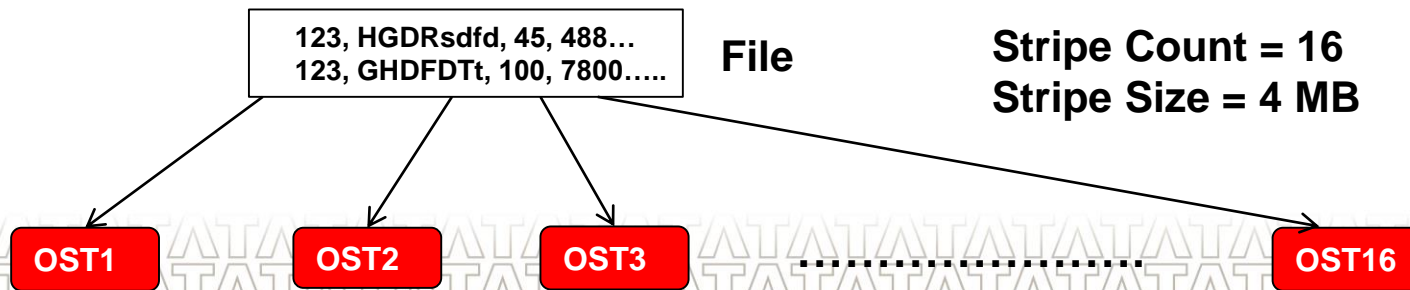# Intel® Enterprise Edition for Lustre* software 2.2 Setup

CPU- Intel(R) Xeon(R) CPU E5-2637 v2 @ 3.50GHz , Memory - 128GB DDR3 1600MHZ

OST OST **4 TB SATA 7200 RPM**

OST

OSS

OST

OST

OST OST

40 Gbps

OST

OSS

OST

MDS

OST OST

MDT

OST

OSS

OST

OST OST

OST

OSS

OST

Parameters Configuration (Hadoop)

Map Slots = 24
Reduce Slots = 32

HDFS: Replication Factor = 2

D

D

D

Map Split size = 1 GB
Block Size = 256 MB
Compression = No

Intel® EE for Lustre

123, HGDRsdfd, 45, 488…
123, GHDFDTt, 100, 7800…..

File

Stripe Count = 16
Stripe Size = 4 MB

OST1   **OST2**   **OST3**   ……………………   **OST16**

Parameters Configuration (Hive)

Parameters	1T	2T	4T
input.filei.minsize	4294967296	8589934592	17179869184
task.io.sort.factor #streams to merge	50	60	80
mapreduce.task.io.sort.mb	1024	1024	1024

Workloads

FSI Workload

- **Single Table**
- **Two SQL queries**

Telecom Workload

- **Two Tables - Call fact details & Date dimension**
- **Two SQL queries - single Map join**

Insurance Workload

- **Four Tables – Vehicle, Customer, Policy & Policy Details**
- **Two SQL queries - having 3 level joins (map as well reduce)**

Example Workload – Consolidate Audit Trail

(Part of FINRA)

Database File (Single table, 12 columns)

Order-id, issue_symbol, orf_order_id, orf_order_received_ts , routes_share_quantity, route_price,...

072900, FSGWTE, HFRWDF, 1410931788223, 100, 39.626602,
072900, VCSETH, BCXFNBJ, 1410758988282, 100, 32.642002,
072900, FRQSXF, BVCDSEY, 1410758988284, 100, 33.626502,
072900, OURSCV, MKVDERT, 1410931788223, 100, 78.825609,
072900, VXERGY, KDWRXV, 1410931788285, 100, 19.526312,

Query

Concurrency =1,2,8

Query

Size : 100GB, 500GB, 1TB, 2TB, 4TB

Query: Print total amount attributed to a particular share code routing during a date range.
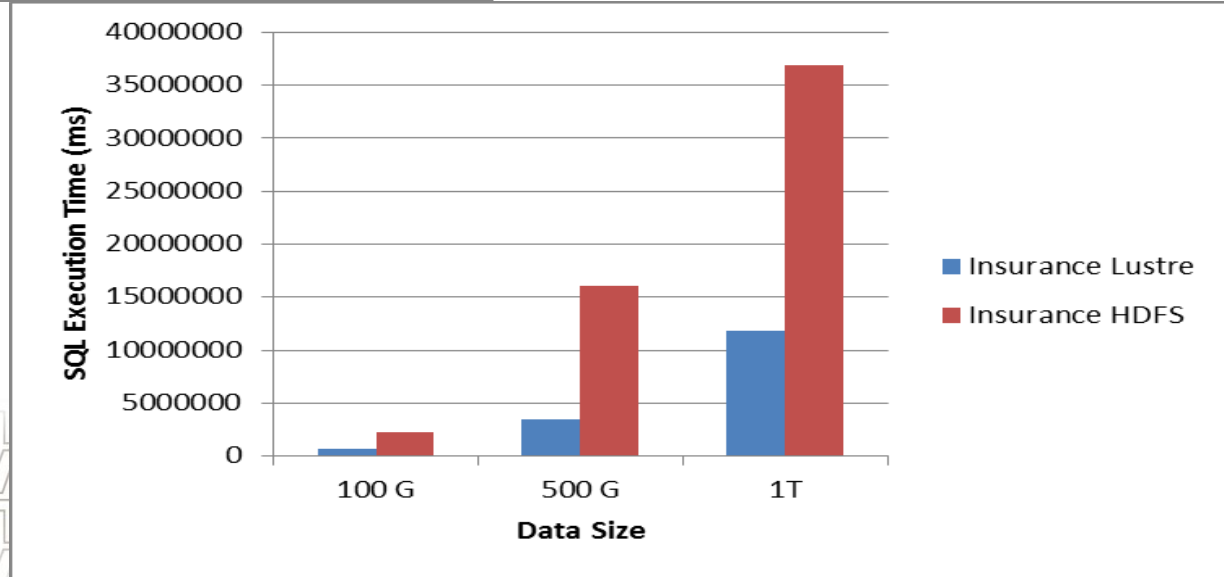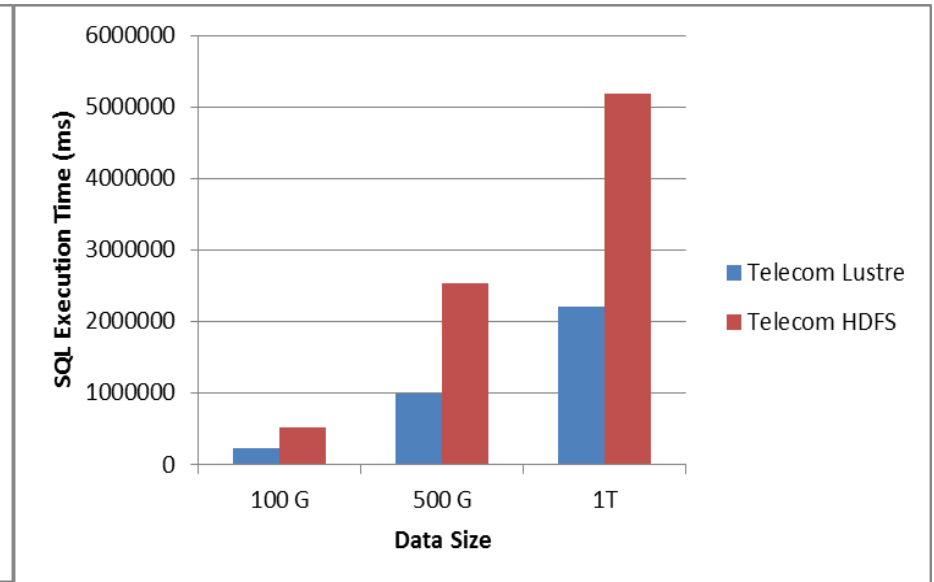
RESULT ANALYSIS

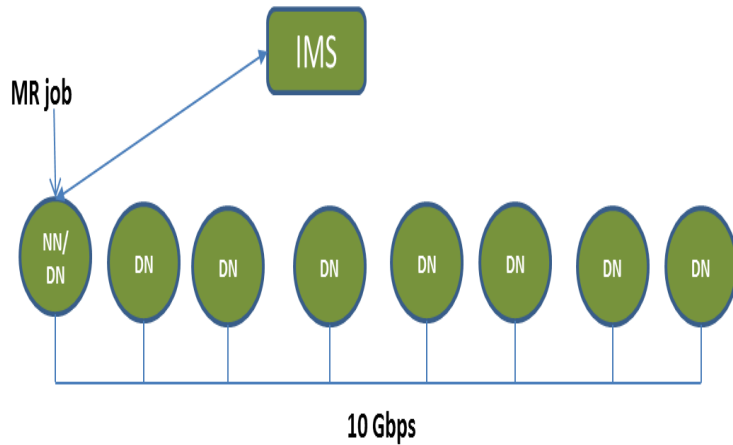# Lustre = 2 * HDFS, data size >>

Concurrency=1
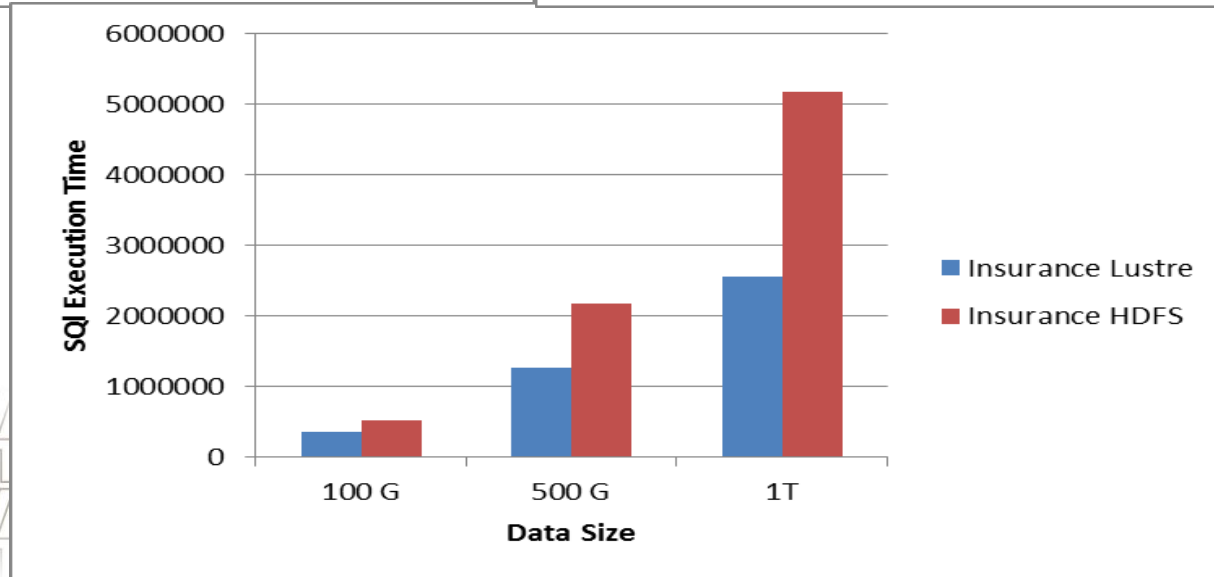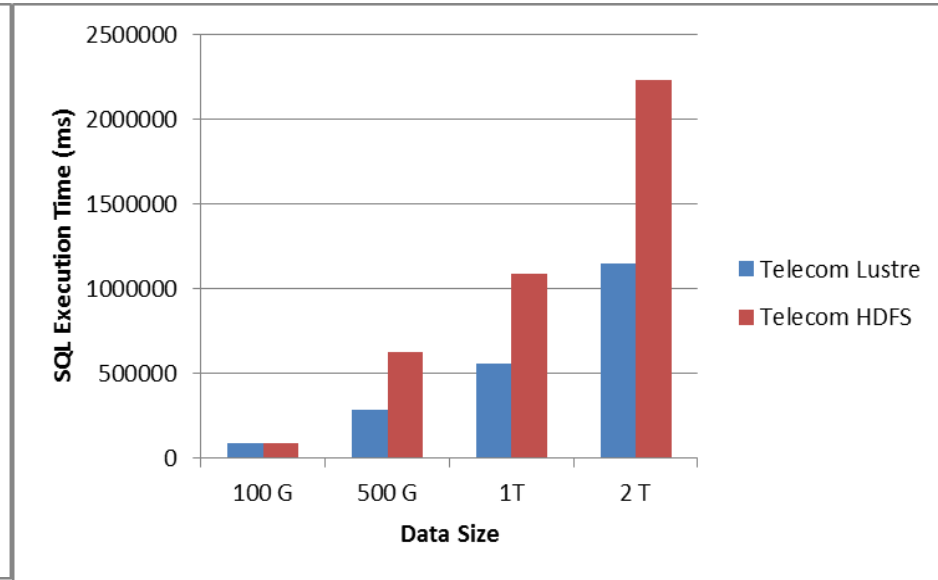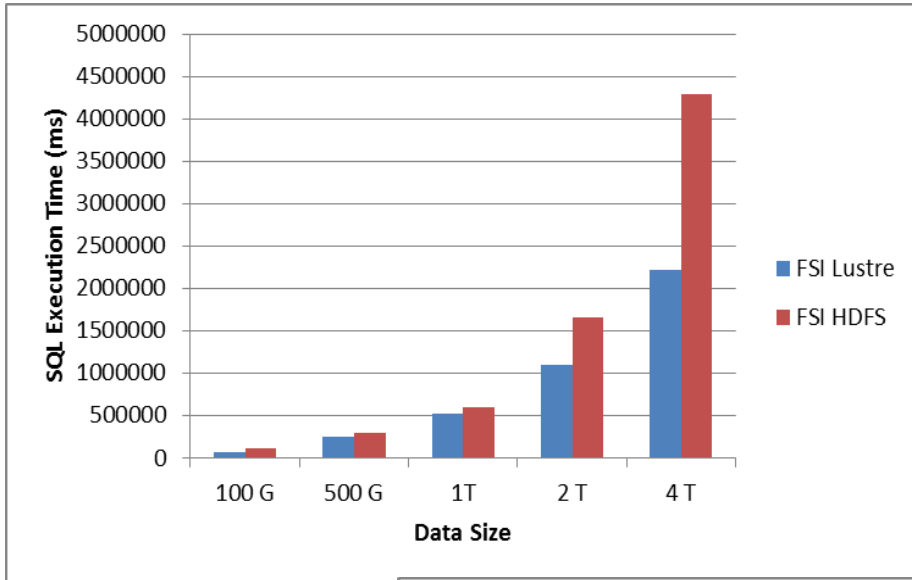
Lustre = 3 * HDFS,   data size >>

Concurrency=8

Hadoop+ HDFS Setup
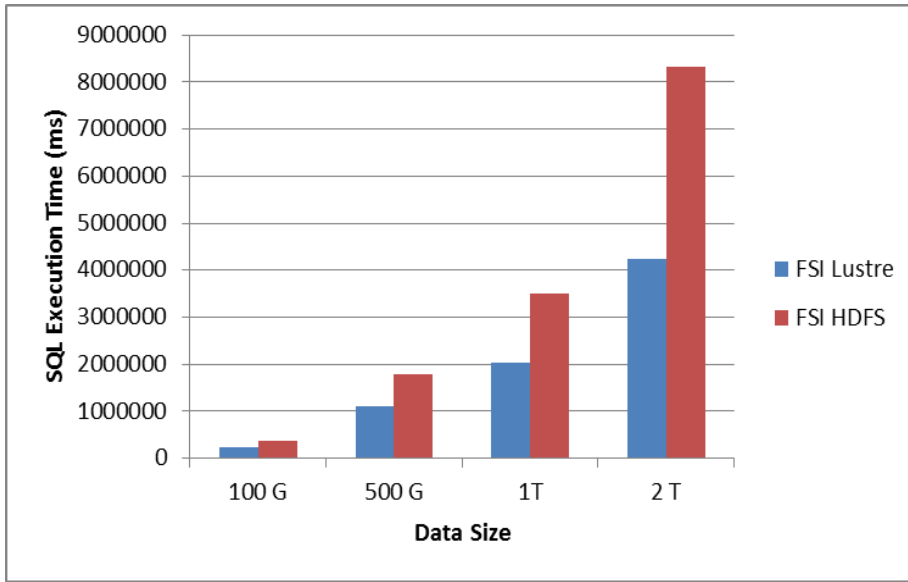
Hadoop+ Intel® EE for Lustre* Setup

Total Nodes = Compute Nodes = 16
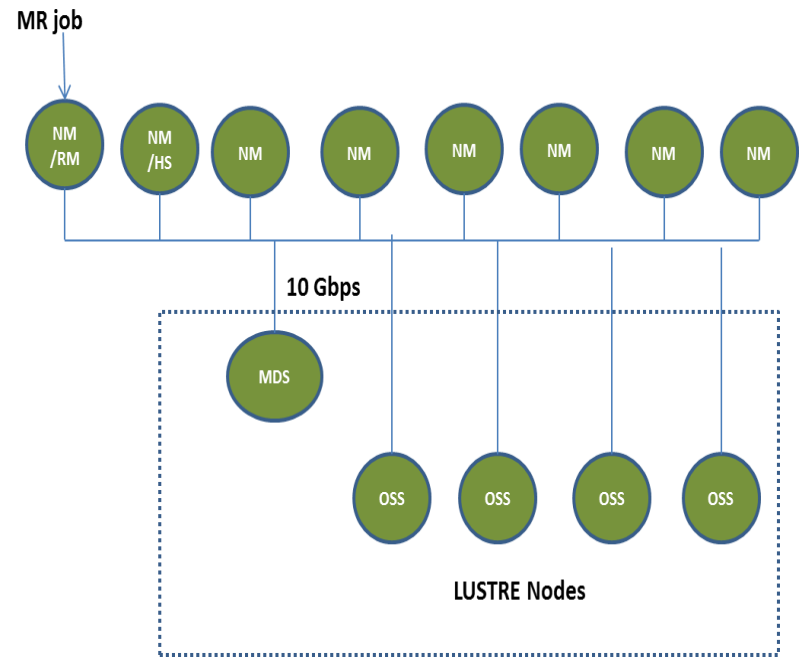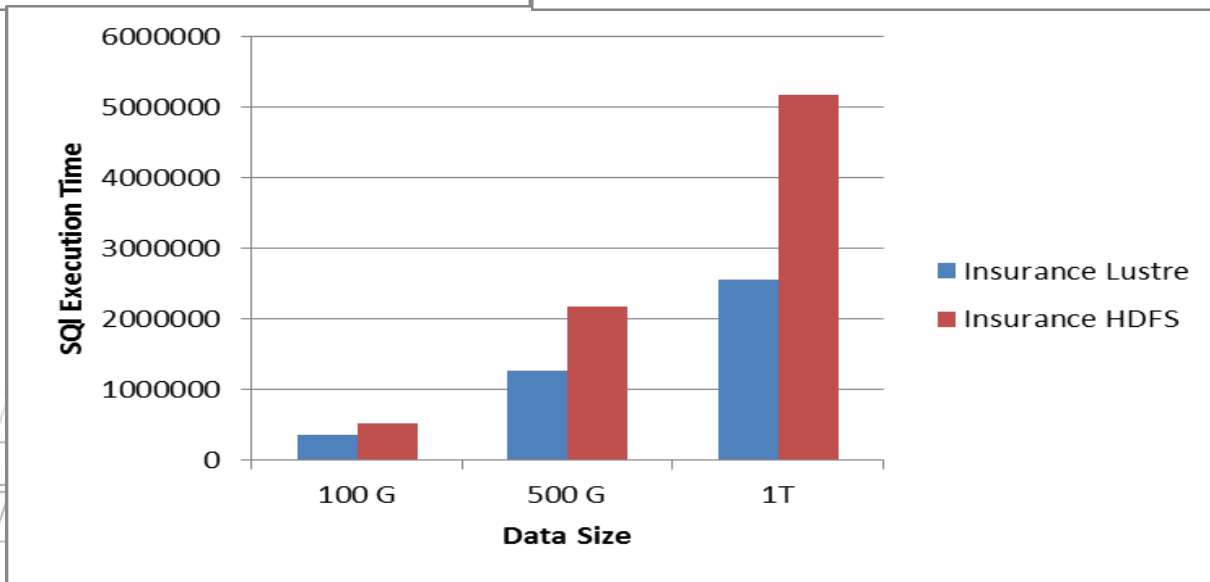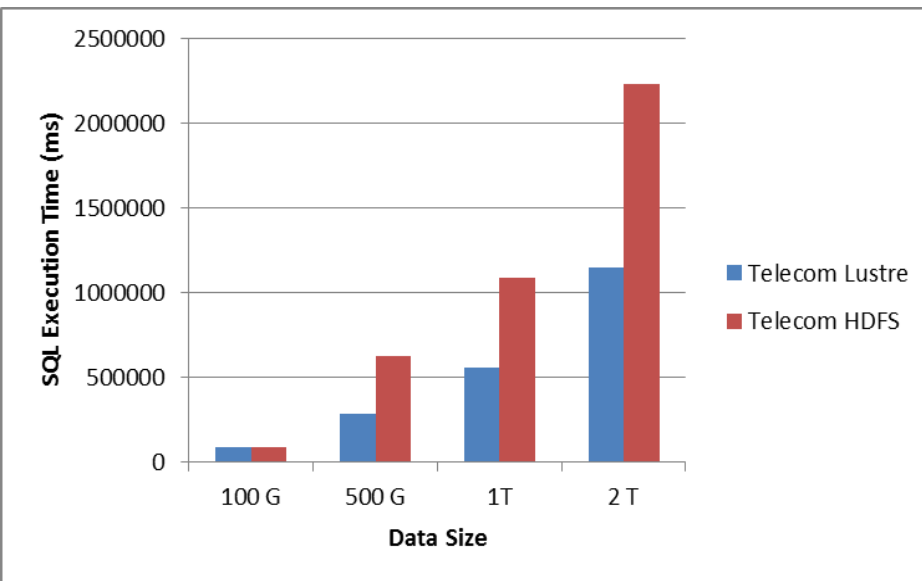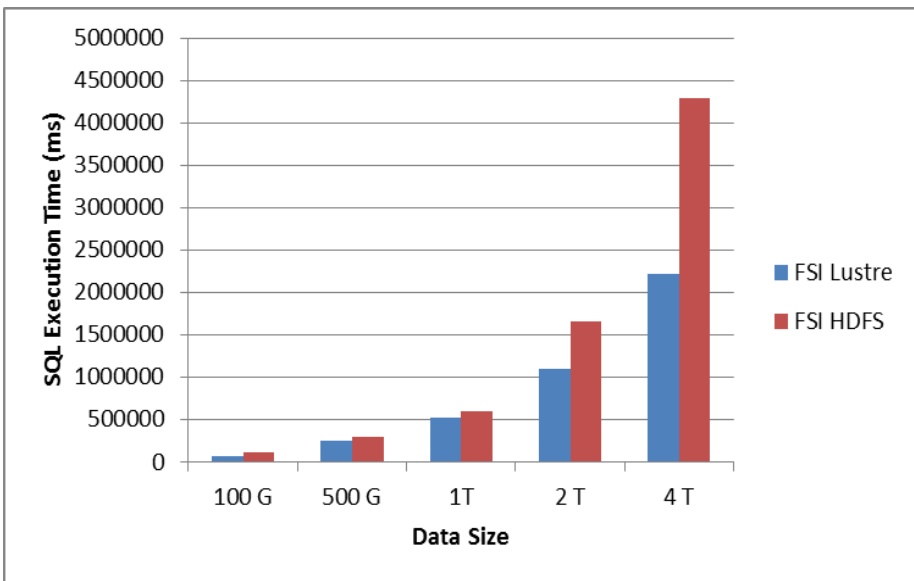
Total Nodes = 16
Compute Nodes = 11
Lustre Nodes = 5

Same BOM –
Lustre.Compute Nodes = 11

Lustre = 2 * HDFS,   data size >>

Conclusion

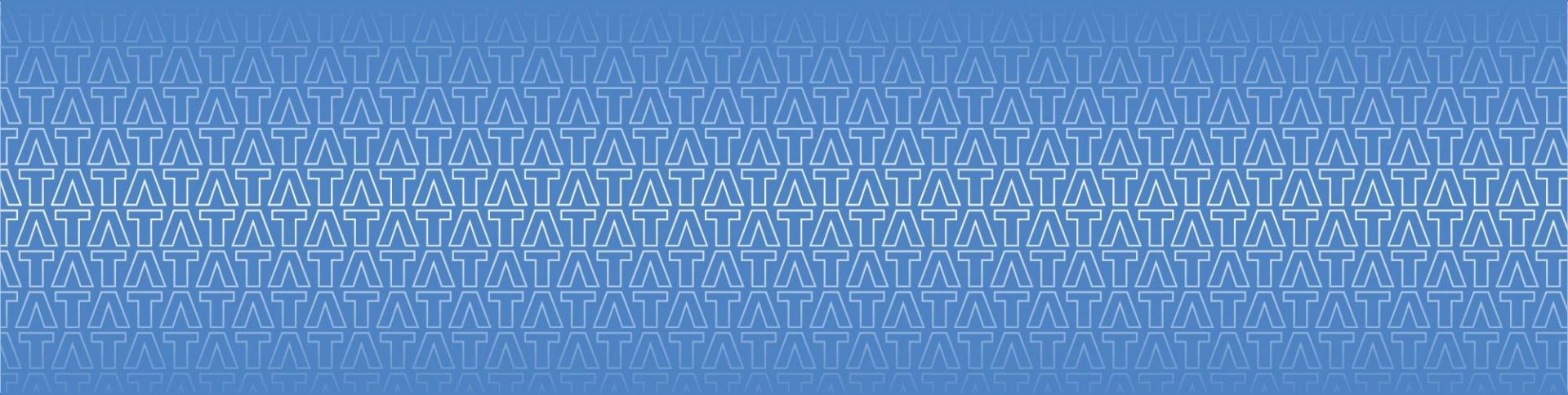❑ Intel® EE for Lustre shows better performance than HDFS for concurrent as well as Join query bound workload

❑ Intel® EE for Lustre = 2 X HDFS for single query

❑ Intel® EE for Lustre = 3 X HDFS for concurrent queries

❑ HDFS: SQL performance is scalable with horizontal scalable cluster

❑ Lustre: SQL performance is scalable with vertical scalability

❑ Future work
  ▪ Impact of large number of compute nodes (i.e. OSSs <<<< Nodes) and scalable Lustre file systems.

Thank You

rekha.singhal@tcs.com