# Performance Analysis of Big Data Analytics on Lustre and HDFS File Systems

Rekha Singhal, Chetan Phalak

TCS Innovation Lab- Performance Engineering, India
{rekha.singhal, chetan1.phalak}@tcs.com

**Abstract.** — Big data technology is widely used for large volume data analysis. Wide acceptance of open source Hadoop platform encourages its use for real time analytics as well; this requires high performance from the system. Moreover, most of the High Performance Computing (HPC) applications may use data analytics as well to improve its execution time by reducing the number of simulation cycles. HDFS is the traditional file system used with Hadoop while Lustre is one of the file system popularly used in HPC systems. Does the same HPC setup be used for data analytics as well? – This paper addresses this question by comparing the performance of Hive SQL and Map-Reduce job executed on Lustre and HDFS file systems. The systems are evaluated for Financial, Telecom and Insurance applications on the Intel HPDA clusters. The results are presented in the paper which shows that application performance on Lustre is at least twice better than on HDFS. The paper also discuss the impact of horizontal and vertical scaling of cluster on performance of application deployed on Lustre and HDFS file systems.

## 1 Introduction

The performance benefits of parallel processing technology have led the migration of existing relational database based analytic applications to big data technologies such as Hadoop and Hive. This migration brings in additional challenges to catch up performance of parallel RDBMS using parallelism for data processing in commodity based nodes' cluster- this raises the need to replace the traditional file systems such as Hadoop Distributed File System (HDFS) with parallel file system such as Lustre, (widely used for HPC). Moreover, convergence of HPC with Big data motivates further to have unified file system to avoid data transfer across different subsystems during HPC simulations. In this paper, we present a benchmark process with the results of evaluating performance on HDFS and Lustre file systems for executing Hadoop Map-Reduce as well as Hive SQL based analytic applications.

Hadoop distributed file system (HDFS) as shown in Fig 1a, distributes data across different commodity nodes with their local storage in a cluster and replicate data across different set of nodes for availability. Each compute node in the cluster will have some part of the whole data sets on its local storage. Lustre is a parallel file system accessed like a centralized system from the cluster compute nodes as shown in Fig 1b. Lustre file system has set of Object Storage Servers (OSS) each having one or more Object Storage Target (OST) as logical disk for storage. It exploits IO parallelism to speed up data access from multiple OSTs simultaneously. There is no data replication, however, HA architecture of Lustre file system ensures fault tolerance.

We have considered Financial (FSI), Telecom and Insurance domain workloads for the file systems comparisons. The native Hadoop system is integrated with HDFS file system. For evaluation of Hadoop on Lustre file system, we have used Hive/Hadoop extensions such as Hadoop Adapter for Lustre (HAL) developed by Intel, while comparing the performance against the (HDFS). The performance metric used for evaluation is SQL query elapsed response time which is the time difference between the submission of the Hadoop job or the Hive query to the cluster and getting the result back. We first presents the results of Map-Reduce(MR) job performance comparison on Lustre and HDFS with Hadoop, A Hive SQL query is translated into a set of Map-Reduce jobs, so we extended the evaluation to more complex jobs generated through Hive SQL. We observed a Map-Reduce job performance is atleast 3 times better on Lustre than on HDFS and is twice better for Hive SQL. The file systems are also compared for concurrent executions of the Map-Reduce jobs and Hive SQLs. The benchmark process highlights the workload choices, data generation, Hive, Hadoop, HDFS and Lustre parameter tuning, performance monitoring, and collection of the measurements and post facto analysis of the experiments.


(a) HDFS (b) Lustre

Figure 1 Architecture (a) Hadoop+HDFS system, (b) Hadoop+Lustre File system

The paper is organized as follows. Section 2 discusses the work done in benchmarking Lustre and HDFS file systems. Section 3 talks about the benchmarking process of the big data systems with both the file systems. Section 4 introduces difference in MR processing on HDFS and Lustre file systems. Section 5 presents the experimental setup and, results and analysis are discussed in Section 6. Finally the paper is concluded in Section 7.

## 2 Related Work

In prior art, Sun Microsystem, Xyratex and Intel have done comparisons of HDFS with Lustre file systems for MR jobs. Sun had benchmarked the system for BigMapOutput and Wordcount applications. They had kept same bill of materials for both, Lustre and HDFS systems. They had observed Lustre performance worse than HDFS for different sizes of data for these applications [1]. The Lustre file system took almost double the time in some cases. Xyratex [2] and Intel [3] earlier evaluations have targeted the MR implementation in HPC environments. The compute nodes are kept same in both the set up. In HDFS set up, compute nodes with local disks are used and for Lustre, separate cluster of OSS and OST is integrated with the compute nodes cluster keeping the same capacity and data transfer from hard disks in both the setup. The evaluations were done for TestDFSIO and Terasort benchmarks. Xyratex observed that using fast interconnect between compute cluster and Lustre file system, Lustre setup significantly outperformed the HDFS. The standard 1 GigE network yielded, on average equivalent results for both Lustre and HDFS, because of the network speed acting as the bottleneck. Intel earlier evaluations showed Lustre being more than 100% faster than HDFS on TestDFSIO benchmarks, and also TeraSort shows 10-15% better performance with Lustre.

In literature, [4] has suggested optimizations in Lustre for improving MR performance in HPC environment. They observed for large number of compute nodes, Lustre faces IO bottleneck which could degrade its performance vis a vis HDFS.
This paper presents performance comparison of Lustre and HDFS for MR and Hive query workloads. The benchmark takes either Hive SQL analytic queries or their corresponding map and reduces functions as application which is executed on Hadoop system. The benchmark evaluated the system for job execution time and uses Markov reward models to evaluate performability (performance in the presence of failures and repairs) of the system with HDFS and Lustre file systems.

## 3 Benchmarking Process for Big Data Systems

Benchmark process of any big data system shall require right tools for data and workload generation and scripts for running workloads with collection of system and job level logs. Since the system is mostly available for short duration, the benchmark has to be very efficient and collects sufficient performance numbers for post facto analysis. The features of various components of benchmarking process are shared as follows.

- Data generation & Loading – For comparing performance of different systems, the data set should be same. Either data generator should have same random seed so that it generates same data set on multiple runs – data generator may generate data directly on HDFS, or data may be generated on one file system (Lustre) and loaded to HDFS from there. A file on Lustre can be directly used by Hadoop without any loading time, however significant time will be spent in loading in HDFS. Data generator should have capability of generating data incrementally for increase in data size to evaluate the system for different data sizes. Data sizes should be large enough to saturate the underlying file systems to get their best and worst performance scenarios.
- Workload generation – Different types of SQL workloads in terms of number of IO accesses and size of IO accesses, may need to be generated to build worst and best cases dependent on the file system architecture. Additionally, workload shall capture both map side and reduce side joins in SQL for extensive evaluation of the systems. Workload size (i.e. concurrency) evaluates the effect of caching and concurrent IO accesses performances in the file systems.
- Hive/Hadoop configurations- The parameters may be tuned for best performance from underlying file systems.
- Performance Data Collection – Hive and MR job logs shall be collected to get job level details, along with system utilizations for all nodes in the cluster, such as CPU, Disk, Memory and Network.
- Performance Analysis – Identify the bottleneck in performance for systems under considerations by co-relating the job and system level performance data.

## 4 Data Processing on HDFS and Lustre File System

Hadoop is a parallel processing framework used for processing Map-Reduce (MR) jobs. Hive SQL is executed as set of MR jobs on Hadoop as shown in Fig 2. There may be concurrent MR jobs executing on the system. A join query may result in writing large intermediate data which may be replicated by replication factor and may lead to large execution time Hive may create its metastore on Namenode and data files mapped as tables may be stored on file system (HDFS or Lustre). In a traditional Hadoop setup, the fault tolerance to the tables is implemented using HDFS data replication factor.

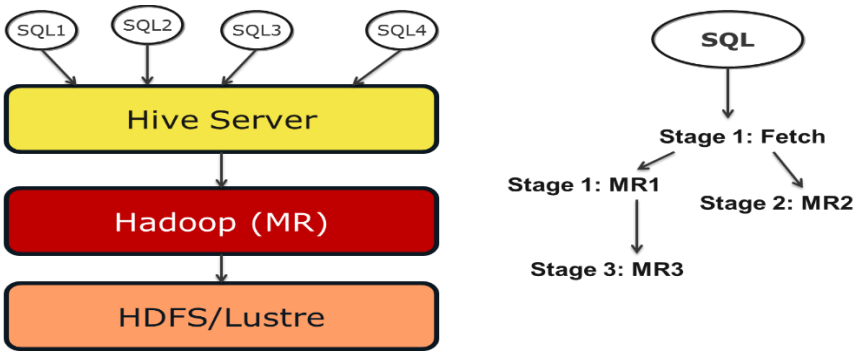



Figure 2: Architecture of Job Execution on HDFS/Lustre File Systems

Hadoop can be configured to access/store data on Lustre file system using Lustre adapters ,such as HAL from Intel, which connects MR jobs/ Hiive SQL MR jobs to access data to/from Lustre file system. All MR jobs reads/write data from Lustre so write and shuffle phases of MR jobs are optimized to reduce MR job execution time. The connectivity of the compute nodes to Lustre shall be large enough to support simultaneous data traffic from all compute nodes to exploit benefits of Lustre file system. In Lustre setup, each compute node does not have local data storage. All data whether Hive metadata, intermediate data and output data etc. is written in Lustre file system only. Intel's Hadoop Adapter over Lustre ( HAL) connects the compute node cluster to the Lustre file system.

Fig 3 shows the difference in MR processing on Lustre and HDFS file systems. In case of Hadoop+Lustre system, the data is read/written by all maps and reduce tasks in Lustre file system only. Data is read in chunks, each chunk by one map process. Each map process write intermediate data in Lustre file system, whose addresses are passed to the reducer tasks on any node. This eliminates the shuffle time which is quite dominant in case of Hadoop+HDFS system.

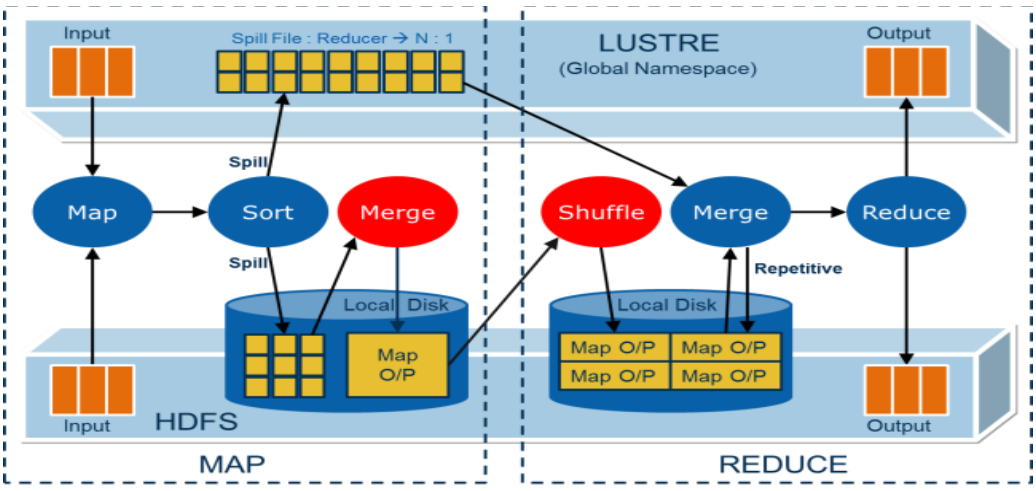
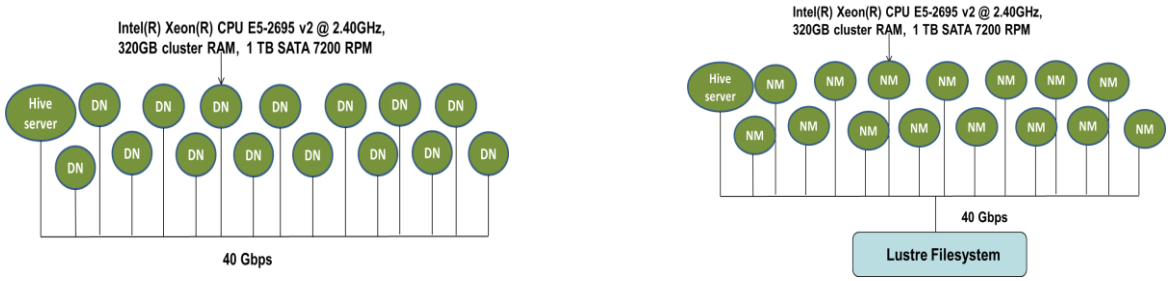



Figure 3: MR processing in HDFS and Lustre file system (source :Intel white paper)

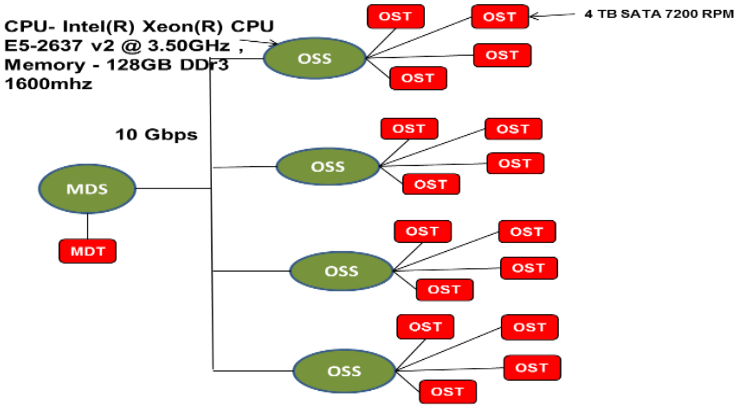
## 5 Experimental Setup

The experiments have been done on Intel Swindon HPDA 16 nodes cluster. Each node is Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz, 320GB cluster RAM and 48 core with 1 TB 7.2K RPM hard disk. The cluster has 27 TB storage with nodes connected through 40 Gigabits network. Each compute node in cluster is loaded with Redhat 6.5, CDH 5.2, Hive 0.13, Intel Enterprise Edition for Lustre 2.2, Hadoop Adapter for Lustre 3.1. The whole cluster is configured one after another for HDFS and Lustre setup for evaluations as shown in Fig 4.

The Lustre setup as shown in Fig 4 (c) consists of one MDS having Intel(R) Xeon(R) CPU E5-2609 v2 @ 2.50GHz X 2, Memory - 64GB DDr3 1600mhz , Disk subsystem - LSI Logic / Symbios Logic MegaRAID SAS 2208, four OSS nodes each with Intel(R) Xeon(R) CPU E5-2637 v2 @ 3.50GHz, 12 core , Memory - 128GB and Four 4TB SATA drives per controller raid 5 configuration per raid set.




(a) Hadoop+HDFS (b) Hadoop+Lustre




(c) Lustre Implementation

Figure 4: Experimental Set up (a) Hadoop + HDFS (b) Hadoop+Lustre (c) Lustre Implementation Details

Benchmarks are executed for different data sizes and concurrency levels for different applications as discussed in the next section. To ensure repeatable results, the benchmarks are executed after flushing OS level caches. Hive/Hadoop system is tuned separately for both the file systems. Appropriate configuration parameters are changed and replicated to all the nodes in the cluster through automated scripts.

Job level and system level monitoring tools are activated before start of each benchmark. Scripts are used to monitor, collect and parse the system level and job level logs for performance analysis. We collect system utilizations (CPU, memory, Disk and Network utilizations) for all nodes in the cluster as well as for Lustre file system

(OSS and MDS) during the workload execution. The performance data of interest is CPU performance numbers(core wise utilizations, IO wait, System busy time, user busy time), IO subsystem details (read bytes/sec, write bytes/sec), Network details (read bytes/sec, transmitted bytes/sec), memory (utilization, page faults)

The SQL execution plan, log file of Hive and various stage jobs' logs generated during execution of a SQL are also collected. Post benchmarking one can co-relate system utilization logs with job logs across different nodes in the cluster. Performance analysis is done for different data sizes and different concurrency levels in both the systems. Horizontal scaling can be examined by analyzing performance across increase in number of nodes in the cluster. While vertical scaling may be evaluated by controlling the number of cores allocated for map and reduce slots in each node in the cluster.

The steps involved in the benchmark exercise are as follows.

1. Choose an application data generator – Financial workload, Telecom workload and Vehicle Insurance workload.
2. Generate data for a given size (100G, 500G, 1T, 2T and 4T) using the data generator
3. Copy the data to HDFS.
4. Prepare the cluster to collect performance monitoring data using in house monitoring tool.
5. Generate the workload (concurrency is configurable- 1, 2 and 8 )
6. On completion of the SQL queries, collect execution log of each job, collect performance monitoring data using TCS tool and aggregate those data at the Name node.
7. Calculate SQL query average execution time and plot CPU, Memory, Disk and Network utilization for the workload execution duration.
8. Repeat from step 5 for different concurrency levels,
9. Repeat from step 2 for different data sizes.
10. Repeat from step 1 for different applications workload.

### 5.1 Workload Details

We have evaluated the system for three applications – Financial (FSI), Telecom and Insurance. We have chosen two queries from each application workload i.e. total 6 SQL analytics queries as shown in Table 1. Financial workload is represented by FINRA Audit trail specifications. The underlying data is organized as single table. The SQL queries in this application do not involve any joins. Telecom workload is represented by daily call details of one of our customer. The data is organized into 2 tables – Call details fact table and Date dimension. The SQL queries of this application involve two table joins- this join is map side join since Data table is very small. Insurance application chosen is our internal implementation of Vehicle Insurance. It has information on policies, policy details, vehicles and customer details organized into 4 tables. The SQL queries involve four table joins with mix of both map side and reduce side joins. The details of workload SQLs are given in Appendix A.

Data is generated as flat files – one file for each table. Each data file is mapped to an external table in Hive. We have data and workload generator tools for these three applications. Data generator generates data on Lustre file system which is loaded into HDFS as well. Workload generator can initiate concurrent SQLs through Hive with think time between any two SQLs. We have data sizes as 100G, 500G, 1T, 2T and 4T and Concurrency degree as 1,2,8.

**Table 1: Business Queries of three Applications.**

| | **Financial** | **TeleCom** | **Insurance** |
|---|---|---|---|
| **Business Query1** | Print total transaction money for a particular SHARE received after a DATE ordered by ORDER received date | Get total calling minutes for each year and month in sorted manner | Get number of active policies for each vehicle make. |

| | | | |
|---|---|---|---|
| **Business Query2** | Print total transaction money for a particular SHARE during date range | Get service charge and cess charge accumulated for each month in year 2000 in sorted manner | Get number of active policies for Audi vehicle. |

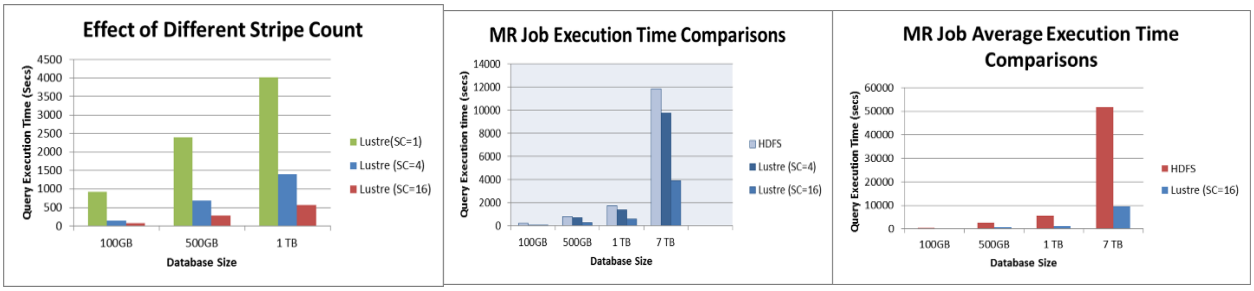
### 5.2 Configuration Details

The MR framework is configured to work optimally in both HDFS and Lustre file system especially in Lustre to control the number of intermediate generated files which increases the workload on MDS and hence may degrade the performance. This is done by controlling map input split size and intermediate sort file size. The other performance sensitive parameters are as follows.

- Lustre Stripe count = 16
- Lustre Stripe size = 4MB
- HDFS replication factor=2
- HDFS block size = 256 MB
- Map Split Size = 256 MB
- Number of Map slots = 24
- Number of Reduce tasks per job = 32

## 6 Results and Analysis

We first did performance analysis of Lustre and HDFS for financial MR job executed as single job as well as with multiple degree of concurrency. We observed application performance is atleast three times better on Lustre than on HDFS with optimal stripe count in Lustre file system as shown in Fig 5(b). The stripe count indicates the OSTs involved in storing the file (i.e. IO level parallelism in accessing file). For default Lustre stripe count (=1), HDFS exhibits better performance since it has IO parallelism of same degree as number of nodes in the cluster as in Fig 5 (a). The concurrent executions of MR jobs has five times better performance on Lustre due to use of Read/Write cache across different tasks as shown in Fig 5(c). We used Markov Reward model to compare performance during compute node failures and repairs (performability) in both the system. Since in HDFS failure of a node implies data transfer to another node for job completion it intuitively perform worse than Lustre during compute node failure which is proven in one of our earlier work [5]. We have also observed the effect of concurrent workload on job performance in Fig 6, where use of read cache in Lustre file system further improves the average execution time of a job however in HDFS it increases for higher degree of concurrency with data size.




(a) Effect of Stripe Count (b) Concurrency Degree =1 (c) Concurrency Degree =8

Fig 5: HDFS vs Lustre Performance Comparison for MR job Execution

The performance comparison was extended for Hive SQL query where number of MR jobs are initiated for single Hive SQL. The detailed measurement results are given in Appendix A. Fig 7 compares the performance of the three workloads on HDFS and Lustre file systems for executing single query at a time, where Lustre shows two time better performance than on HDFS due to 16 way IO parallelism in the Lustre file system.. Fig 8 compares the performance of execution of 8 concurrent queries in each workload, where Lustre exhibits three time better performance than on HDFS due to use of read cache across concurrent tasks in centralized Lustre file system.

In Lustre set up, a SQL query performance is limited by the Lustre read/write bandwidth which depends on the transfer speed of the underlying disks and the number of disks provided the connected network has bandwidth more than disks total transfer speed. We have observed in Fig 5, as numbers of concurrent maps are increased in the cluster by increasing the map slots, the SQL performance improves till 196 concurrent map slots and remains the same thereafter. Further increase in number of map slots per node or number of nodes, does not improve the performance.

This concludes that SQL performance on 8 nodes of compute nodes with Lustre setup (having 5 nodes) is same as that on 16 nodes of compute nodes with Lustre setup.

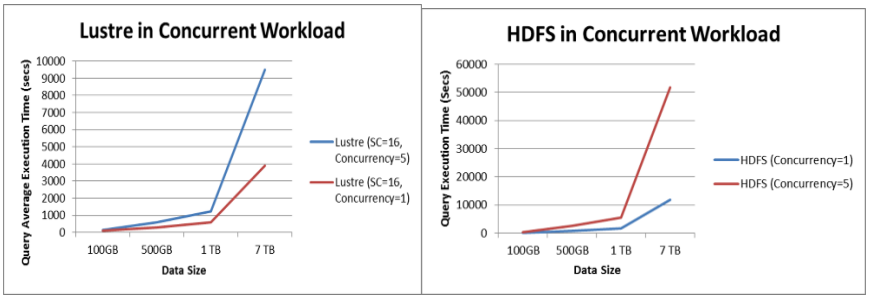



Fig 6: Effect of Concurrent workload (degree = 5) on Performance of MR job in Lustre and HDFS file system

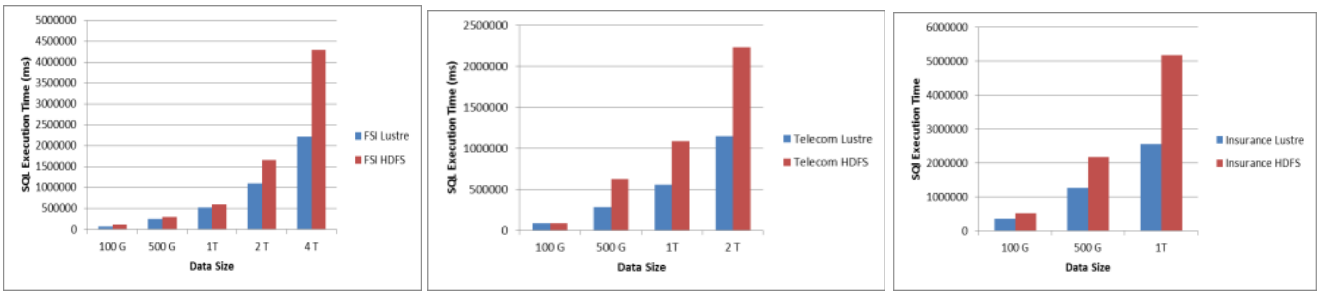



Fig 7: HDFS vs Lustre for Hive SQL based FSI, Insurance and Telecom Workload with Concurrency=1.

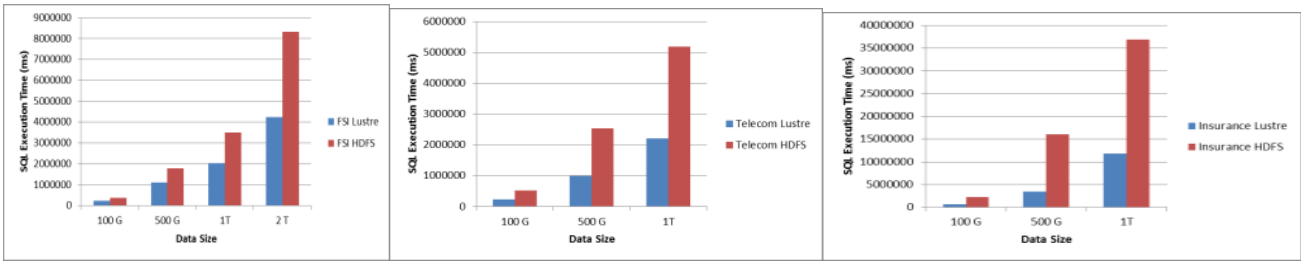



Fig 8 : HDFS vs Lustre for Hive SQL based Financial, Insurance and Telecom Workload with Concurrency=8

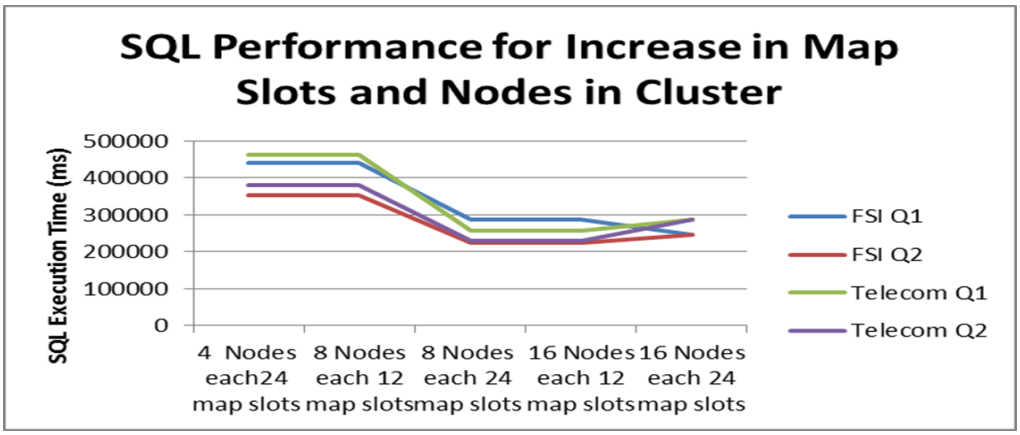



Fig 9: Effect of Number of Concurrent Map slots on SQL performance in Lustre Setup

In case of HDFS setup, we observe in Fig 6, a SQL performance improves with increase in number of nodes in the cluster, but vertical scaling of node by increasing number of concurrent map slots does not improve the performance. Therefore, a SQL performance in HDFS setup is better on 16 nodes cluster than on 8 nodes cluster.

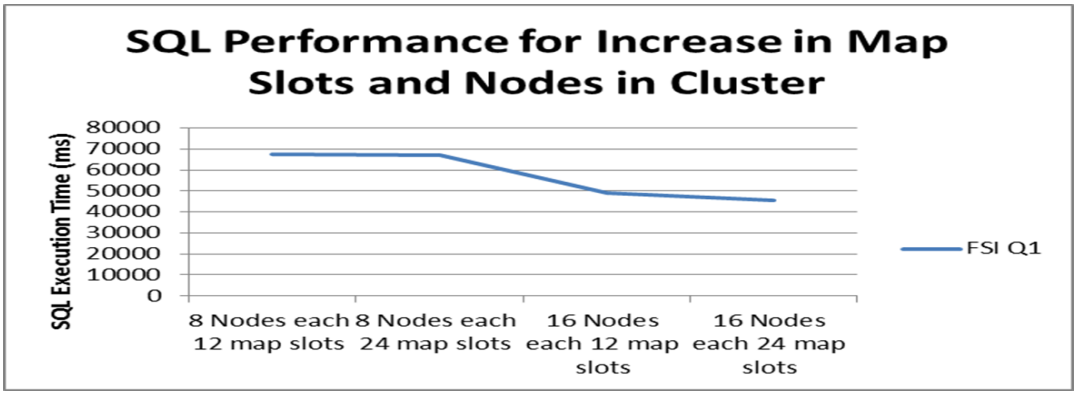



Fig 10: Effect of Number of Concurrent Map slots on SQL performance in HDFS Setup

SQL query execution time for all workloads without any concurrency is marginally better in Lustre than in HDFS on smaller data size (< = 1T). However, it is at least 2 times better in Lustre than HDFS for data size more than 2T in Fig 3 and 4. In case of concurrency, Lustre cache reduces the average execution time. We observe the SQL performance to be at least 2 times better than on HDFS which become 3 times better for Insurance workload having multiple joins. In case, of workload having more join operations, the shuffle time become dominant- this is negligible in Lustre setup. Moreover, we can conclude that even for the same BOM, the performance of applications on 16 nodes setup with Lustre (11 compute nodes and 5 lustre nodes) is atleast twice better than on 16 nodes cluster with HDFS setup on larger data size (>= 2TB)

## 7 Conclusions

Analytic applications process large data sets which may require high performance computing (HPC) systems for fast results. Moreover, HPC applications also use past simulations data analysis to reduce their number of simulations for an application. There is convergence of big data and HPC technologies. In this paper we have

evaluated performance of analytic application on traditional HDFS file system and on Lustre, a HPC parallel file system. The performance evaluation is done for financial, telecom and insurance workload for different data sizes and degree of concurrency. We concluded that applications perform at least two times better than on Lustre file system than on HDFS on large data sizes. Lustre file system being a centralized parallel file system avoids time delay towards shuffling of intermediate data across node in the cluster during MR processing. For higher degree of concurrency, the performance gap widens due to use of read cache in Lustre file system, by concurrent tasks in compute cluster. Application performance linearly scales with vertical scaling of compute nodes in Lustre file system unless it is limited by either the connecting network or read/write bandwidth of the Lustre file system. For HDFS, an application performance scales with horizontal scaling of the cluster and limited by the performance of the network interconnects.

**References**


1. Using Lustre with Apache Hadoop, Sun Microsystem, http://wiki.lustre.org/images/1/1b/Hadoop_wp_v0.4.2.pdf , 2008
2. Map/Reduce on Lustre Hadoop Performance in HPC Environments, Nathan Rutman Senior Architect, Networked Storage Solutions, www.xyratex.com/sites/.../Xyratex_white_paper_MapReduce_1-4.pdf , 2011
3. Hadoop MapReduce over Lustre, , Omkar Kulkarni High Performance Data Division , Intel, http://opensfs.org/wp.../LUG2013_Hadoop-Lustre_OmkarKulkarni.pdf , April 16, 2013
4. Jie Yu, Guangming Liu, Wei Hu, Wenrui Dong and Weiwei Zhang, Mechanisms of Optimizing MapReduce Framework on High Performance Computer, 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, 2013
5. Rekha Singhal, Manoj Nambiar, Harish Sukhwani and Kishore Trivedi, Performability comparison of Lustre and HDFS for MR application", IEEE ISSRE Industry Track, Naples, Italy, 2014


## Appendix A : Hive SQL queries and Performance Results

Table 2: SQL queries of Financial, TeleCom and Insurance Workload

| | Financial | TeleCom | Insurance |
|---|---|---|---|
| **SQL Query 1** | SELECT sum(routed_shares_quantity * route_price) FROM external_RT_Query_Extract WHERE issue_symbol like 'XLP' AND from_unixtime(cast ((orf_order_received_ts/1000) as BIGINT),'yyyy-MM-dd') >= "2015-02-22" AND from_unixtime(cast((orf_order_received_ts/1000) as BIGINT),'yyyy-MM-dd') <= "2015-02-23" Group by orf_order_received_ts; | SELECT Year, Calendar_Year_Month, SUM(Total_Minutes) AS Total_Minutes FROM Billing_Fact, Date_Dimension WHERE Billing_Fact.ID = Date_Dimension.ID GROUP BY Year, Calendar_Year_Month ORDER BY Year, Calendar_Year_Month; | select count(policyid) from policy_details pd, policy p, customer c, vehicle v where pd.status = 'active' and c.vin = v.vin and c.custid = p.custid and p.policyid = pd.policyid group by v.make; |
| **SQL Query 2** | SELECT sum(routed_shares_quantity * route_price) FROM external_RT_Query_Extract WHERE issue_symbol like 'XLP' AND from_unixtime(cast ((orf_order_received_ts/1000) as | SELECT Year, Calendar_Year_Month, SUM(Service_Tax) AS SERVICE_TAX, SUM(EDU_CESS) AS SURCHARGE FROM Billing_Fact Bill_Fact, Date_Dimension Date_Dim WHERE Bill_Fact.ID = Date_Dim.ID AND Date_Dim.Year = 2000 GROUP BY | select count(policyid) from policy_details pd, policy p, customer c, vehicle v where pd.status = 'active' and v.make = 'Audi' and c.vin = v.vin |

| | | | |
|---|---|---|---|
| | BIGINT),'yyyy-MM-dd') >= "2015-02-22" AND from_unixtime(cast((orf_order_received_ts/1000) as BIGINT),'yyyy-MM-dd') <= "2015-02-23"'; | Year,Calendar_Year_Month ORDER BY Year, Calendar_Year_Month DESC; | and c.custid = p.custid and p.policyid = pd.policyid group by v.make; |

The performance result of workload given in Table 2 for its execution on Lustre file system with single and multiple degree of concurrency is given in Table 3 and 4 respectively. The performance result of workload given in Table 2 for its execution on HDFS file system with single and multiple degree of concurrency is given in Table 5 and 6 respectively. Some of the entries in the following tables are blank because the measurements could not be collected due to unavailability of the system.

Table 3: SQL query execution time on Lustre for different Data size with Concurrency level=1

| Data Size | Financial (Q1) | Financial( Q2) | Telecom(Q1) | TeleCom(Q2) | Insurance(Q1) | Insurance(Q2) |
|---|---|---|---|---|---|---|
| **100G** | 67027 | 66938 | 88259 | 475796 | 352248 | 318519 |
| **500 G** | 247167 | 246176 | 288133 | 288136 | 1266527 | 1090758 |
| **1 T** | 517034 | 499052 | 558147 | 544464 | 2547161 | 3012193 |
| **2T** | 1093403 | 1029179 | 1146230 | 1110944 | 5290394 | 4436477 |
| **4 T** | 2224485 | 2048939 | 2192729 | 2129132 | 11987541 | 10045973 |

Table 4: SQL query average execution time on Lustre for different Data size with Concurrency levels=2,8

| Data Size | Financial (2) | Financial(8) | Telecom(2) | TeleCom(8) | Insurance(2) | Insurance(8) |
|---|---|---|---|---|---|---|
| **100G** | 75676 | 232123 | 107571 | 228672 | 399195 | 707115 |
| **500 G** | 324012 | 1112427 | 306416 | 998194 | 1393627 | 3496120 |
| **1 T** | 582021 | 2040012 | 552772 | 2207245 | 4083082 | 11784969 |
| **2T** | 1210769 | 4227486 | 1131778 | 4312937 | 6681344 | 19040671 |
| **4 T** | 2713224 | 8279089 | 2247804 | 9037845 | 15249303 | - |

Table 5: SQL query execution time on HDFS for different Data size with Concurrency level=1

| Data Size | Financial (Q1) | Financial (Q2) | Telecom(Q1) | TeleCom(Q2) | Insurance(Q1) | Insurance(Q2) |
|---|---|---|---|---|---|---|
| **100G** | 118129 | 62103 | 89588 | 151324 | 511312 | 458880 |
| **500 G** | 300786 | 569018 | 625319 | 328469 | 2167207 | 1837002 |
| **1 T** | 598321 | 1172297 | 1090237 | 532088 | 5165412 | 3631764 |
| **2T** | 1659090 | 1407981 | 2226470 | 1569665 | - | - |
| **4 T** | 4291342 | 4358676 | - | - | - | - |

Table 6: SQL query average execution time for different Data size with Concurrency levels=2,8

| Data Size | Financial (2) | Financial (8) | Telecom(2) | TeleCom(8) | Insurance(2) | Insurance(8) |
|---|---|---|---|---|---|---|
| **100G** | 97085 | 372815 | 198549 | 513243 | 707216 | 2269193 |
| **500 G** | 548077 | 1785157 | 580176 | 2533725 | 3714726 | 16052359 |
| **1 T** | 1176229 | 3515214 | 1103093 | 5187330 | 7842380 | 36921138 |
| **2T** | 1997015 | 8311708 | 4038339 | - | - | - |
| **4 T** | 6883126 | - | - | - | - | - |