Data Transformation Methodologies between Heterogeneous Data Stores
A Comparative Study

Arnab Chakrabarti[1,2] and Manasi Jayapal[2]
[1]*Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany*
[2]*Databases and Information Systems, RWTH Aachen University, Aachen, Germany*

Keywords: Industrial Data Exchange, NoSQL, Hadoop, Pentaho, Talend, Apache Scoop, Cassandra.

Abstract: With the advent of the NoSQL Data Stores, solutions for data migration from traditional relational databases to NoSQL databases is gaining more impetus in the recent times. This is also due to the fact that data generated in recent times are more heterogeneous in nature. In current available literatures and surveys we find that in-depth study has been already conducted for the tools and platform used in handling structured, unstructured and semi-structured data, however there are no guide which compares the methodologies of transforming and transferring data between these data stores. In this paper we present an extensive comparative study where we compare and evaluate data transformation methodologies between varied data sources as well as discuss the challenges and opportunities associated with it.

1 INTRODUCTION

RDBMS (Relational Database Management System) have been running in data centers for over 30 years and this could no longer keep up with the pace at which data is being created and consumed. Limitations of scalability and storage led to the emergence of NoSQL databases (Abramova et al., 2015). This breakthrough technology became increasingly popular owing to the low cost, increased performance, low complexity and the ability to store "anything". This led the pathway to the solutions that included distributed application workloads and distributed data storage platforms such as MapReduce which transformed into an open source project called "Apache Hadoop" (Schoenberger et al., 2013). These tools and technologies have increased the need to find new ways of transforming data efficiently between different heterogeneous data stores for organizations to derive maximum benefits of being a data-driven enterprise.

For our work we had to study the platforms involved in transforming data in detail and choose the most appropriate storage systems by determining a selection criteria. Fundamental reasons like the varied nature of the platforms and the difficulty in choosing the right technology components made the initial decision making cumbersome. The focus was then to find out the process of transforming data from one database system to the other. It was important to create a systematic procedure of how the process of transformation was done. After which the appropriate technologies involved in transformation were identified and then data was seamlessly transferred between the databases. An important step was to then determine the characteristics of comparison which could be system/platform dependent or application/algorithm dependent. An experimental set up to execute the above steps was then done and finally transformation technologies were compared. While much has been done in describing the platforms and tools in great depth, however, there exists no guide to a comparative study of data transformation between different big data stores and there lies the contribution of our work which is presented in this paper.

2 RELATED WORK

Our comparative study presented in this paper could serve as guidelines for organizations looking to migrate to other platforms by helping them choose the most efficient way of transforming their data from one platform to another. Further, data transformation also finds its utilization in "data exchange" which is the process of taking data structured under a source

Chakrabarti, A. and Jayapal, M.
Data Transformation Methodologies between Heterogeneous Data Stores - A Comparative Study.
DOI: 10.5220/0006438802410248
In *Proceedings of the 6th International Conference on Data Science, Technology and Applications (DATA 2017)*, pages 241-248
ISBN: 978-989-758-255-4

schema and transforming it into data structured under a target schema (Doan et al., 2012). Industrial Data Exchange(IDX) is one such application that provides a solution to the problem of moving or sharing real-time data between incompatible systems (Xchange, 2015). Our work is also a part of another such application called "Industrial Data Space" project which lays the groundwork for the foundation of an Industrial Data Space consortium. This research project by the Fraunhofer Institutes (Otto et al., 2016) aims at creating an inter company collaboration with fast, automated data exchange to help companies achieve an edge in the international competition.

Our work is steered particularly into the comparison of data transformation between the most widely used platforms and tools which will be used effectively to select methodologies amid different partners of the Industrial Data Space project

2.1 Data Transformation

The upsurge of NoSQL databases led to the downfall of relational database. In this scenario many organizations switched to NoSQL databases and needed to transform their data at large. For example, Netflix moved from Oracle to Cassandra (Netflix, 2015) to prevent a single point failure with affordable scalability. Coursera also experienced unexpected deterioration with MySQL and after thoroughly investigating Mongodb, DynamoDB and HBASE shifted to Cassandra (Thusoo et al., 2010). Thus, the study of data transformation can be useful for many enterprises.

3 OVERVIEW OF THE APPROACH

In Figure 1 below we depict the solution scope of our work. It presents a classification of the data stores that are addressed in our work with the red dotted lines indicating the transformation process between the data stores that we report in this paper.

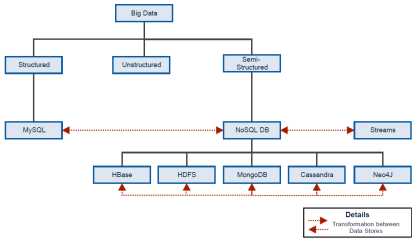

Figure 1: Data transformation between data stores.

Keeping one data store as the source, all the possible transformation technologies to the other data stores were found. For example, transformation from Cassandra to Neo4j could be done using Talend, Pentaho, Neo4j-Cassandra data import tool and such possible technologies were considered individually.In addition to the technologies, another important goal was to find an overlap of methods between them. For instance, if the transformation technologies between data store D1 and data store D2 were M1, M2 and M3. i.e,

$$D1 \rightarrow D2 = M1, M2, M3 = \text{Set A (Consider)}$$

Similarly,

$$D3 \rightarrow D4 = M4, M5 = \text{Set B (Consider)}$$
$$D4 \rightarrow D1 = M6, M7 = \text{Set C (Consider)}$$

Set A, Set B and Set C should not be totally independent of each other i.e, there should be an overlap of methods between them. This was to ensure a meaningful comparative study at the end. Figure 1 depicts all the potential transformation possibilities. However, an important point to note here is that this list is not exhaustive and there could be other technologies available for transformation.

3.1 Selection of the Data Stores

Based on the research done in the field of comparative study (Abramova et al., 2015),(Kovacs, 2016), leading surveys (DBEngines, 2008) and Google trend based on the most popularly used NoSQL databases - MongoDB (document database), Cassandra and HBASE (column-store database) were chosen for our work. Though HDFS(Hadoop Distributed File System) is good for sequential data access, being a file system it lacks the random read/write capability. Thereby HBASE, a NoSQL database that runs on top of the Hadoop cluster was also included in our work. Keeping in mind the rule of variety in diversity, including a graph database was important. Graph databases are fast becoming a viable alternative to relational databases. They are relatively new, so most of the comparative studies do not compare graph databases in their work. One of the primary goals our work was to study the results and challenges in data transformation for graph databases. Thus we also include Neo4j as one of the NoSQL data stores.

3.2 Selection of the Data Transformation Platforms

The scope of our work was not to study the various tools individually in detail but limited to choosing the platforms based on a selection criteria and then comparing the transformation technologies between them. Further for an optimal solution to manage big data,

choosing the right combination of a software framework and storage system was of utmost importance. Although the software framework and storage system share a very important relationship, data transformation between platforms is mainly dependent on the storage systems chosen to work with the framework. For instance, if an organization chooses to run Apache Spark with MongoDB and another organization chooses Apache Spark with Cassandra, data transformation techniques need to be applied between MongoDB and Cassandra. Therefore, the primary focus of this work is on the transformation between various data stores and the coming sections head in that direction.

3.3 Choosing the Right Data Sets

Each of the chosen databases have a unique way of storing data and it was important to choose the dataset that can be replicated and stored across all of them to obtain uniform results. For example, the chosen dataset would be stored as JSON in MongoDb, CSV in Cassandra and a graph in Neo4j. Other factors were also considered like Hadoop's HDFS demon called *Namenode* holds the metadata in memory and thus, the number of files in the filesystem is limited to the Namenode's memory. The best option in this case would be to use large files instead of multiple small files (White and Cutting, 2009). Of all the options available, the Yahoo Webscope Program provided a "reference library of interesting and scientifically useful datasets for non-commercial use by academics and other scientists" (Labs, 2016). Different sizes of Yahoo! Music ratings data were requested and finally 3 data sets consisting of 1 million, 2 million and 10 million records were selected for the proof of concept. These data sets consisted of random information like music artists, songs, user preferences and ratings etc. This variation of data was needed to test if every chosen technique was able to scale and perform well with the increase in size.

4 CHARACTERISTICS OF COMPARISON

In this section we present a set of well defined characteristics that we considered for our comparative study. Previous research (Tudorica and Bucur, nd) indicates that NoSQL databases are often evaluated on the basis of scalability, performance and consistency. In addition to these, system or platform dependent characteristics could be complexity, cost, time, loss of information, fault tolerance and algorithm dependent characteristics could be real-time processing, data size support etc. For the scope of this work we only considered *Quantitative Characteristics* which is a set of values that focuses on explaining the actual results observed by performing the transformation. These numerical aspects were carefully studied before collecting the data to give the best comparative picture at the end. Below we present the metrics that we have used to evaluate our results.

- **Maximum CPU Load:** This is the maximum percentage of the processor time used by the process during the transformation. This is a key performance metric and useful for investigating issues if any. There was no quota set and the process was monitored by shutting down all other unnecessary processor technologies.
- **CPU Time:** CPU time is the time that the process has used to complete the transformation.
- **Maximum Memory Usage:** Maximum memory usage is the maximum percentage of the physical RAM used by the process during the transformation. An important metric to keep a track of resource consumption and impact it has on the time.

Analyzing the changes in the resource consumption is an important performance metric. Maximum CPU load, CPU time and maximum memory usage were calculated for each of the transformation techniques using *htop* command in ubuntu.

- **Execution Time:** The total time taken to complete the transformation. This was measured using the respective tools for each for the transformation techniques to compare the faster means of transforming data between any two given databases. This time included the time taken to establish a connection to the source and destination databases, reading data from the source and writing data to the destination. As a common unit, all the results were converted into seconds. However, some transformations that took a long time to complete, were expressed in minutes and hours.
- **Speed:** Speed is computed as the number of rows transformed per second. For each of the transformation techniques, this value was obtained from the tools using which the transformation was performed. The value of speed was important, for example, in the transformations to Neo4j. Slow transformations of 3 - 30 rows/second suggested the need to find alternative faster techniques.

5 IMPLEMENTATION DETAILS

In this section we will describe in detail the implementation of this study in which an experiment to execute the transformations between the data stores was setup.

5.1 Experimental Setup

An ubuntu machine with the following configuration was chosen to run all the transformations described in our work:

- Ubuntu : 15.10
- Memory : 15.6 GiB
- Processor : Intel R Core TM i7 -3770 CPU @ 3.40 GHz *8
- OS type : 64-bit
- Disk : HDD 500 GiB, Norminal media rotation rate : 7200, Cache/ Buffer size : 16 MB, Average seek read : $<$ 8.5 ms, Average seek write : $<$ 9.5 ms

Only the technology and the concerned databases were made to run whereas all the other processes were shut down to make sure that no other variables had an impact on the results. After the completion of every job, the technologies and databases were restarted. Htop command was used to analyse the processes running on the machine with respect to the CPU and memory utilization. The process specific to the technology was studied using Htop and the quantitative characteristics like maximum CPU%, Memory% and Time are documented as Maximum CPU load, Maximum Memory Usage and CPU Time respectively. Figure 2 shows an instance of the Htop command in which the characteristics are highlighted.

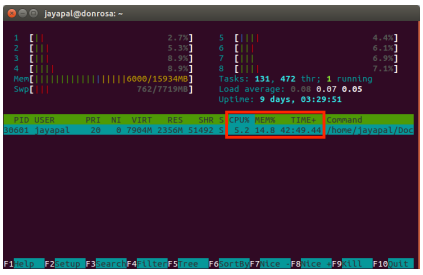

Figure 2: Htop instance.

5.2 Workflow

For the experimental setup as described in the paper a total of 74 transformations were performed and each of them were first tested using 1 million records and then subsequently with 2 million records. The initial plan was to use 10 million records for every transformation but, some transformations took a long time to complete and given the time restriction it was not feasible to do so. Every technology underwent a stress test to check if it could complete a job with 10 million records and the other transformations were tested with 1 and 2 million records. Below as shown in Figure 3 we give a workflow that we adhered to during the entire course of the experiment. This was made to systematically run and verify each job as it was essential in concluding this study with a fair comparison between the technologies.

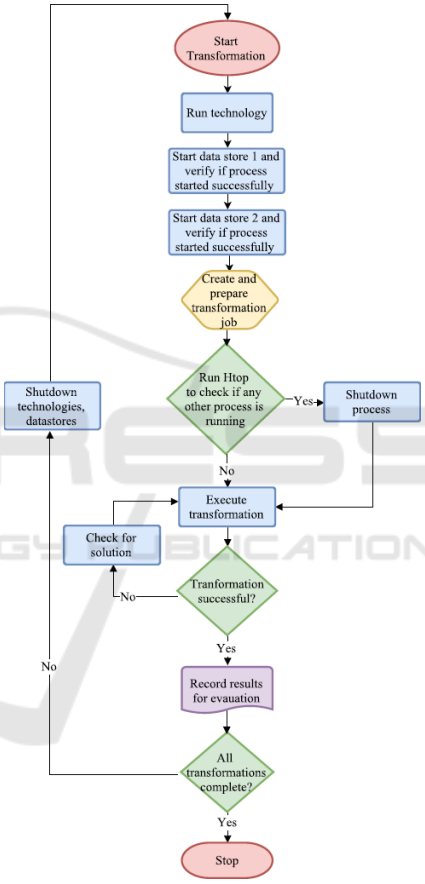

Figure 3: Workflow to run the transformations.

5.3 Designing the Data Store

An obstacle that was not expected in the initial design phase was finding the right combination of database versions to be used with the technologies. The same version had to be compatible with all the technologies of transformation. First, the decision to find the right version of the technologies was made.

Talend is an open source tool and the latest available version was used. However, the latest releases of Pentaho are not open source anymore and an older

stable version 5.0.1 was used. Sqoop was used in accordance with Hadoop.

Thus, the following versions of the technologies were chosen:

- Pentaho data-integration - 5.0.1 stable version
- Talend open studio for big data - 6.1.1
- Apache Hadoop - 1.2.1
- Apache Sqoop - 1.4.4

Below we report the versions of the databases that we selected for our study :

- MySQL - 5.5.46
- Cassandra - 1.2.4
- Mongodb - 3.0.9
- Neo4j - 2.2
- Hbase - 0.94.11

It was then vital to store the chosen datasets into all the databases. Since the datasets procured were in the TSV format, an initial idea was to import it into Cassandra and then transform them into the other data stores. However, a decision was made to generalize this as a procedure and first the dataset was stored into Hadoop. The big elephant is not datatype dependent and engulfs everything thrown into it. Further, it was much faster to start the Hadoop demons and upload the datasets into HDFS.

After the data was stored into 64 MB chunks in HDFS, the technologies were started one by one. All databases were subjected to a connection test to check if they were successfully connected to the tools. Next, a trial run of the transformations was done. In this way, two goals were achieved - one testing the whole system and other getting the datasets into all the data stores.

5.4 Manual Methods of Transformation

The results of all transformations between Cassandra and Neo4j were recorded for 2 million records. However, Neo4j was becoming extremely unresponsive after transforming 2 million records. For records upto 580,000, successful results were obtained and a decision was made to restrict the number of records in transformations involving Neo4j and Talend/Pentaho. To make matters worse, the number of rows transformed per second into Neo4j using various technologies was limited between 3-28. For example, HDFS to Neo4j using Pentaho, data was transformed at an average speed of 18.5 rows/second and it took a total of 8.7 hours to complete. Similarly for others, the job would have taken hours to complete and it was not in the scope of our work to test each one of them. Thus, alternative manual methods of transformation were tested. Surprisingly, the transformation yielded better results and the jobs ran to completion. Manual method of transformation was able to surpass the 5,80,000 barrier and run jobs with 2 million records.

5.4.1 Cassandra - Neo4j Transformation Process

This is a multi step transformation process. As a first step, it was important to make sure that enough memory was reserved for the Neo4j-Shell. The memory mapping sizes were configured according to the expected data size. Transforming data from Cassandra to Neo4j was done in the following steps:

- **Exporting Cassandra Table:** The data needed to be transformed was exported into a CSV file using a simple cqlsh query. Cassandra also stores data in a CSV format, hence there was no change in the structure of the file.
- **Validating CSV File:** The exported CSV file had to be checked for format description, consistency and the like. To ensure efficient mapping into Neo4j, the file was checked for binary zeros, non-text characters, inconsistent line breaks, header inconsistencies, unexpected newlines etc. This was done with the help of a user friendly tool called CSVkit.
- **Load CSV into Neo4j:** Neo4j provides "LOAD CSV" for the medium sized datasets and "Super Fast Batch Importer" for very large datasets. For the datasets chosen by us, LOAD CSV was sufficient to complete the transformation. LOAD CSV is used to map CSV data into complex graph structures in Neo4j. According to the Neo4j developer site, it is not only a basic data import cypher query but a powerful ETL tool. It is often used to merge data and convert them into graph relationships. Using the LOAD CSV, the validated CSV file was transformed into Neo4j.

Similarly, in the Neo4j - Cassandra transformation, the graph database was first exported from Neo4j using the copy query and then imported into Cassandra after creating an appropriate table.

6 RESULTS AND DISCUSSION OF THE CHALLENGES

In this section we discuss the results of the experiment and also report the challenges that we faced during the entire phase.

6.1 Comparing Transformation Dependent Quantitative Characteristics

This formative evaluation was used to monitor if the study was going in the right direction. The data transformation methodologies that were implemented in this work were compared with one another and evaluated in the matrix as described. This on-going evaluation was done in parallel with the implementation stage to facilitate revision as needed. Since everything could not be anticipated at the start of the study and due to uncertain changes that occurred at different phases, a revision of the methodologies was necessary at every stage.

6.1.1 Transformation Results

An environment as described earlier was setup; the values of maximum CPU load, CPU time, maximum memory usage were recorded using the Htop command, outcome of execution time, speed were documented from the respective technology used in the transformation and the results were compiled as shown from table 4 to table 9. There are 6 data stores in this study and keeping the source constant and varying the target data store results in 5 combinations per data store. The technologies involved in this transformation were Pentaho, Talend and Sqoop (Mappers from 1 to 4). The number of Mappers are represented with "M" in the table.

- MySQL – > Other Data Stores: Sqoop was much easier to implement and use in most cases; however, an increasing number of mappers did not show a pattern in the execution time. To HDFS, although Pentaho performed this transformation in the best way, the output file records in HDFS consisted of extra 0's which were added when it was converted into BIGINT datatype. To MongoDB, each row of the table in MySQL was assigned an object id when it was transformed into JSON objects to be stored in MongoDB. To Neo4j, Pentaho and Talend were not successful in transforming large datasets into Neo4j. The transformation was very slow in the order of 3-28 rows/second. Thus, both the transformations were not completed till the end.
- HDFS –> Other Data Stores: To MySQL, the output table according to the desired schema should be declared before starting the transformation. Although, Talend started off with 372 percent maximum CPU usage, it went down to 1% CPU utilisation during most of the time with mysql utilising 98% of the CPU load during this time. To HBASE, Talend was faster when compared to Pentaho but, it became unresponsive after the transformation was complete. To Neo4j, since this was one of the first instances to be tested, Pentaho was allowed to run the transformation into completion and it took 8.7 hours; Talend transformed data at 3.2 records per second and it would have taken approximately a day's time to complete.
- HBASE —> Other Data Stores: On the whole, performance of HBASE was not up to the mark with Talend. Most of the transformations involving Talend and HBASE ended in making Talend unresponsive.
- MongoDB –> Other Data Stores: To Neo4j, the quality and structure of the output graph in Neo4j using Pentaho depends on the cypher query given manually in the generic database connection. On the other hand talend has an inbuilt Neo4j connector and the resulting graph is generated based on the schema declared by the user.
- Cassandra –> Other Data Stores: To Neo4j, Pentaho and Talend were very slow in transforming data. An alternative manual method was then used which completed transforming 2 million records in a total time of 153.224 seconds. However, the output of the manual method depends on the cypher query used. Despite completing 2 million records, Neo4j became unresponsive in the end.
- Neo4j –> Other Data Stores: To MySQL, since Pentaho 5.0.1 was connected to Neo4j using a generic database connection, MySQL was throwing errors with respect to the length of the record specified from the graph database. The transformation to HDFS was a little different than the others in this category. For all the other databases, the schema was identified automatically in Pentaho, but since Hadoop is not datatype dependent, attributes from Neo4j nodes were directly saved into HDFS. As Pentaho 5.0.1 does not support Neo4j directly, there was no option to specify the schema explicitly.

MySQL–>Other databases						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
MySQL–>HBASE	Pentaho	88.5	26.50	4.3	43.63 sec	45840
	Talend	138	8.32	6.9	17.82 sec	112233
	Sqoop(M=1)	22.4	3.43	1.8	28.7963 sec	69453
	Sqoop(M=2)	32.6	2.52	2.7	21.8568 sec	91504
	Sqoop(M=3)	27.3	2.74	2.4	27.1181 sec	73751
	Sqoop(M=4)	34.9	2.58	2.9	23.785 sec	84086
MySQL–>HDFS	Pentaho	46.4	1.6	12.9	4.643 sec	430755
	Talend	354	1.9	6.6	5.53sec	361663
	Sqoop(M=1)	34	3.38	2.1	12.038 sec	166140
	Sqoop(M=2)	36.4	2.37	2.1	10.6057 sec	188577
	Sqoop(M=3)	31.1	3.85	1.9	13.5969 sec	147092
	Sqoop(M=4)	29	3.94	2.4	13.663 sec	146380
MySQL–>Cassandra	Pentaho	30.3	457.26	14.8	2152.291sec	929
	Talend	132	7.82	5.4	14.38sec	139082
MySQL–>MongoDB	Pentaho	340	104.31	10.2	194.048sec	10306
	Talend	169	61.37	6.3	133.38sec	14994
MySQL–>Neo4J	Pentaho	40.7		6		28
	Talend	167		6.8		3.2

Figure 4: MySQL to other data stores.

HDFS-->Other databases						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
HDFS-->MySQL	Pentaho	80.4	90.18	5.1	219.6858 sec	9103
	Talend	372	103.47	6.8	318.57 sec	6278
	Sqoop(M=1)	26.3	10.38	2.8	47.5353 sec	42073
	Sqoop(M=2)	28.6	9.56	2.7	41.5702 sec	48111
	Sqoop(M=3)	26.7	9.82	2.9	45.0631 sec	44382
	Sqoop(M=4)	32.3	3.74	3.3	13.663 sec	146380
HDFS-->HBASE	Pentaho	76.4	14.39	1.6	37.63 sec	53149
	Talend	136	8.27	6.6	12.7 sec	157480
HDFS-->Cassandra	Pentaho	34.8	505.8	8.1	33.66045 mins	990
	Talend	265	4.32	6.7	11.09sec	180342
HDFS-->MongoDB	Pentaho	91.2	27.64	4.6	44.673 sec	44769
	Talend	123	53.38	6	128.01 sec	15623
HDFS-->Neo4J	Pentaho	36		6.2	8.7 hours	18.5
	Talend	133		6.6		3.32

Figure 5: HDFS to other data stores.

HBASE-->Other databases						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
HBASE-->MySQL	Pentaho	56	75.48	4.7	192.54 sec	10280
	Talend	74.2	153.74	7	441.28 sec	4532
HBASE-->HDFS	Pentaho	127	56.83	6.2	146.3 sec	13670
	Talend	412	43.27	6	123.25 sec	16227
HBASE-->Cassandra	Pentaho	103	184.92	4.8	342.23 sec	5844
	Talend	216	74.22	5.3	128.48 sec	15566
HBASE-->MongoDB	Pentaho	73	26.62	6.1	72.48 sec	27593
	Talend	249	138.28	5.6	296.16 sec	6753
HBASE-->Neo4J	Pentaho	146		5.7		24.7
	Talend	368		6.6		26.1

Figure 6: HBASE to other data stores.

MongoDB-->Other databases						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
MongoDB-->MySQL	Pentaho	281	93.82	8.6	259.899 sec	7695
	Talend	99	104.28	10	279.56 sec	7154
MongoDB-->HBASE	Pentaho	157	21.35	7.3	47.38 sec	42211
	Talend	318	8.32	9.8	15.74 sec	127064
MongoDB-->HDFS	Pentaho	208	4.29	6.7	16.588 sec	120569
	Talend	99.5	3.51	6.7	8.96 sec	223214
MongoDB-->Cassandra	Pentaho	49.4	643.75	2.7	2025.733 sec	987
	Talend	372	6.94	6.5	15.05 sec	132890
MongoDB-->Neo4J	Pentaho	125		6.4		20 - 23
	Talend	48.5		6.8		23 - 24

Figure 7: MongoDB to other data stores.

Cassandra-->Other databases						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
Cassandra-->MySQL	Pentaho	44.9	63.72	14.8	266.560 sec	7503
	Talend	153	96.28	5.8	525 sec	3809
Cassandra-->HBASE	Pentaho	78	17.35	4.3	48.38 sec	41339
	Talend	86	26.57	5.6	64.720 sec	30902
Cassandra-->HDFS	Pentaho	22.3	3.27	9.7	23.910 sec	83647
	Talend	107	21.86	5.8	56.48 sec	35410
Cassandra-->MongoDB	Pentaho	231	36.91	5.9	59.460 sec	33636
	Talend	202	89.47	6.2	268 sec	7462
Cassandra-->Neo4j	Pentaho	30		7.1		22
	Talend	142		6.6		16
	Manual(Step1)	257	52.38	26.3	140.345 sec	14250
	Manual(Step2)	274	4.69	23.3	12.879 sec	155291

Figure 8: Cassandra to other data stores.

Neo4J-->Other databases (5,80,000 records)						
	Technologies	Maximum CPU Load	CPU Time (seconds)	Maximum Memory Usage	Execution Time	Speed (rows/sec)
Neo4J-->MySQL	Pentaho	57.4	11.74	3.8	56.480 sec	10269
	Talend	214	3.28	4.9	7.19 sec	80667
Neo4J-->HBASE	Pentaho	92.4	5.35	4.6	14.63 sec	39644
	Talend	209	2.92	5.7	7.5 sec	77333
Neo4J-->HDFS	Pentaho	146	1.27	4.2	3.729 sec	155537
	Talend	200	0.93	6.1	4.68 sec	123931
Neo4J-->Cassandra	Pentaho	31.1	114.86	6.3	592.77 sec	978
	Talend	214	1.01	6	4.02 sec	144278
	Manual(Step1)	104	9.77	18.7	26.358 sec	22004
	Manual(Step2)	94.8	92.53	16.8	303.178 sec	1913
Neo4J-->MongoDB	Pentaho	246	2.41	5.5	7.913 sec	73297
	Talend	202	18.39	6	39.03 sec	14860

Figure 9: Neo4j to other data stores.

6.1.2 Processing Speed of Transformation Methodologies

The processing speed of all transformation methodologies involved in transforming data from one database to the other have been plotted as shown from figure 10 to figure 15. This gives a clear picture of which technology was the fastest in comparison to the others. The average rows transformed per second for each technology has also been plotted to convey the capability of each technology.

6.2 Summarization of the Results

Although, a conclusion on the fastest tool amongst them on the whole cannot be drawn, there were cer-

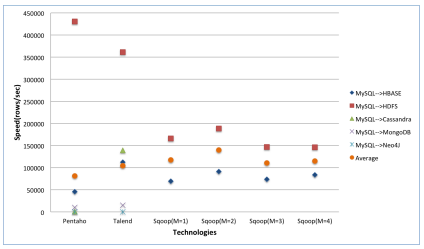

Figure 10: MySQL to other data stores.

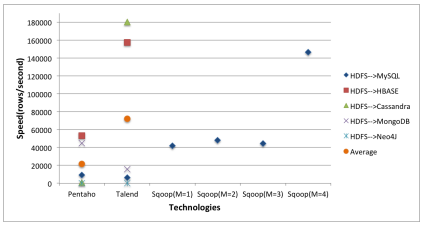

Figure 11: HDFS to other data stores.

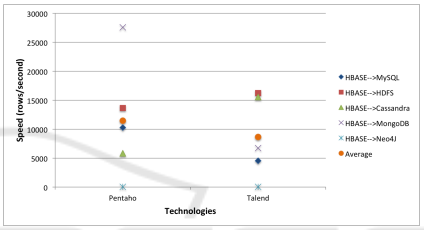

Figure 12: HBASE to other data stores.

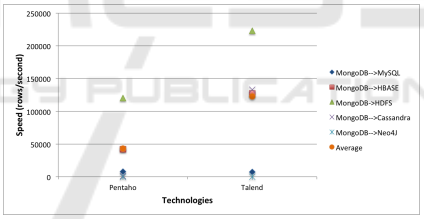

Figure 13: MongoDB to other data stores.

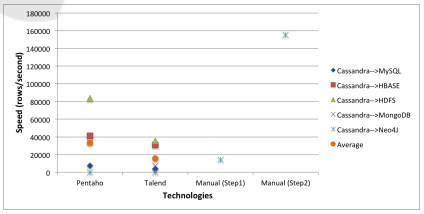

Figure 14: Cassandra to other data stores.

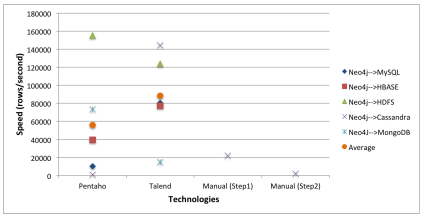

Figure 15: Neo4j to other data stores.

tain results worth noting:

- Talend reached a higher maximum CPU load than Pentaho in most scenarios.
- In transformations from all data stores (except Neo4j) to MySQL, Pentaho proved to be faster than Talend.
- There was a significant difference in the results of the transformation from all stores to Cassandra. Talend was much faster in performing transformations, sometimes even 10,000 times faster than Pentaho in these cases. Ironically, for the transformations from Cassandra to all other data stores, Pentaho was much faster compared to Talend.

Throwing light on the other transformation technologies: Although manual method is a multi-step (export, data validation, load) process of transformation, it was much faster than Pentaho or Talend. While its counterparts were extremely slow in some cases and transformed only 5,80,000 records in others, manual method was able to complete the jobs with 2 million records. A big drawback was that in some cases Neo4j became unresponsive.

7 CONCLUSION AND FUTURE WORK

The main aim of our work was to compare these technologies in detail using well defined characteristics and datasets.Though there exist other ways of transforming data like using commercial tools, but the crux of this study was to compare the open source tools which are freely available resources for every user. When tools like Pentaho and Talend did not deliver, other alternatives like manual methods were defined. All the 74 transformation methodologies in this work were implemented individually and evaluated. Further as a reference, all the challenges faced during the course of this work have been documented. Our vision for our work in this paper is that it could serve as a guidelines to choose suitable transformation technologies for organizations looking to transform data, migrate to other data stores, exchange data with other organizations and the like. Although the number of technologies could be limited with some factor, there can always be more data stores that can be a part of this comparative study. Every organisation has it's own needs and thereby follows different database solutions. Adding more databases will help widen the study and prove beneficial for future users.

REFERENCES

Abramova, V., Bernardino, J., and Furtado, P. (2015). Sql or nosql? performance and scalability evaluation. *International Journal of Business Process Integration and Management*, 7(4):314.

DBEngines (2008). Db-engines ranking. http://db-engines.com/en/ranking. Accessed: 2016-01-25.

Doan, A., Halevy, A., and Ives, Z. G. (2012). *Principles of data integration*. Morgan Kaufmann Publishers In, Waltham, MA.

Kovacs, K. (2016). Cassandra vs mongodb vs couchdb vs redis vs riak vs hbase vs couchbase vs hypertable vs elasticsearch vs accumulo vs voltdb vs scalaris comparison: Software architect kristof kovacs. http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis. Accessed: 2016-01-20.

Labs, Y. (2016). Webscope datasets. http://webscope. sandbox.yahoo.com/. accessed: 2016-02-22.

Netflix (2015). Case study: Netflix. http://www.datastax.com/resources/casestudies/netflix. Accessed: 2016-02-02.

Otto, B., Juerjens, J., Schon, J., Auer, S., Menz, N., Wenzel, S., and Cirullies, J. (2016). Industrial data space - digital sovereignity over data. *Fraunhofer-Gesellschaft zur Foerderung der angewandten Forschung e.V.*

Schoenberger, V. M., Cukier, K., and Schonberger, V. M. (2013). *Big data: A revolution that will transform how we live, work, and think*. Eamon Dolan/Houghton Mifflin Harcourt, Boston.

Thusoo, A., Anthony, S., Jain, N., Murthy, R., Shao, Z., Borthakur, D., Sharma, J. S., and Liu, H. (2010). Data warehousing and analytics infrastructure at facebook. *SIGMOD'10*, 978-1-4503-0032-2(10):06.

Tudorica, B. G. and Bucur, C. (n.d). A comparison between several nosql databases with comments and notes.

White, T. and Cutting, D. (2009). *Hadoop: The definitive guide*. O'Reilly Media, Inc, USA, United States.

Xchange, I. D. (2015). Industrial communications, industrial it, opc, profibus - industrial data xchange (idx). http://www.idxonline.com/Default.aspx?tabid=188. accessed: 2015-12-04.