BigAnalytic:Large-sacle Data Management & Analysis Framework

[1]Ebraheem M.Alhaddad, [2]Fathy E. Eassa
[1]PhD Student, [2]Full Professor
[1,2]Computer Science Department,
[1,2]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia

***Abstract*—With the advance in development of digital devices and increase in Internet access and speed; it has become possible to generate large volume of data, which is known as big data. This big data is characterized by a unique features such as huge volume, high speed of data generation and the diversity of data structures. This big data contains precious value for decision makers in organizations. To get this precious value, we need systems that is compatible with the characteristics of big data. In this work, we present an innovative framework for big data management and analysis. BigAnalytic is featured by the generation of a metadata store which is used to describe the big data and map data generated from different sources, which in turn facilitates the management of big data. BigAnalytic also featured utilization of software agent for big data analysis. In comparison with existing data management and analysis systems, BigAnalytic provides a significant performance improvement.**

***Index Terms*—Big Data, Agent, Data analysis, Cluster Computing, Data Management, Metadata.**

I. INTRODUCTION

Big data is attracting widespread interest of both information technology specialists in research and business communities. After reaching the zettabyte barrier in 2010 and exceeding 1.8 zettabytes in the generated - increasing in the five years by a factor of 9 [1]. The forecast for 2025, is that the internet will overcome the every living being brain capacity in the whole word. Although we have recognized the importance of big data now a days, researchers are still having different opinions and definitions. This is because of the different needs and approaches for each researcher, enterprises, data analysts and technical practitioners.

Doug Laney[2], an analyst at the META group (presently Gartner) is considered to be the one to offer the first real big data definition, nevertheless the term "big data" had been used earlier. Although, the term big data didn't stated explicitly in the definition, the 3V, which is the main big data characteristics, are introduced for the first time. The Vs resemble volume, velocity and variety. The 3V definition will dominated the big data definition, presented by IT specialists such as Gartner, IBM and Microsoft, for the years to come.

In this report [3] McKinsey \& Company studied the potential value that big data can create for organizations and sectors of the economy and seeks to illustrate and quantify that value. The research discovered that data make a remarkable change in the world economy by improving businesses efficiency and productivity and support them in competing with the public sector which will eventually benefit consumers. For example, the use of big data can enhance the efficiency and quality of the heath care service in US by an estimated value of three hundred billion USD yearly. 2/3 of that in health care expense reduction.

With the tremendous generated and stored data by many sources, it vital for companies and organization to exploit new technologies in order to obtaining maximum value of big data. Volume, Variety and Velocity unique characteristics of big data that cut off the development of new technologies facilitate getting value from data. Big data management and analysis is at the core of the techniques has to be developed for the best benefit of big data value. In literature many frameworks developed for this purpose such as MapReduce, Spark, SCOOPY...etc. Despite the advantages provided by these systems to the fact that it still suffers from some defects and need to be treated. Some analysis operations may need to be performed only part of the data. Currently analysis tasks are performed on all data regardless of this fact. Thus developing methods to facilitate selective data access, which helps improve the performance of data analysis and management systems.

After the map function complete processing the data, its output is send to the reduce cluster nodes. The transmission process degrade the overall system performance. In addition, Extracting metadata about the data to be saved may help in the classification of large amounts of data and thus save the most important data and neglect the data that is less important or redundant and does not add value to the data.

II. BIGANALYTIC

The outline of BigAnalytic is presented in this section. Figure 1 depicts the major components of the framework. As we can see that five major components that form the framework namely: Metadata Manager, Topic Modeling, Topic Tree Builder, Query Mapping. Metadata extractor is a group of specific purpose programs applied using mobile agents. For each data type existed in the big data an adequate extractor is implemented. The main purpose of the extractor is process the big data looking for the best metadata that and then extract that information. Metadata store is where the metadata just extracted by metadata extractor is saved for further processing. As long as we are dealing with big data, the size of metadata is also expected to be big. For instance, let

the size of big data to be 20 Petabyte and the size of metadata extracted from this data constitute only 5% of the original data, this mean that the size of metadata is 1 Petabyte which considered also big amount of data. In order to access this metadata efficiently we suggest using NoSQL data modules for storing metadata. For a deeper understanding of big data and in order to facilitated finding data in response of a data query, Topic modeling a process used applied in order to build a model the describe the topics exist in collection of data. In our framework, the input for the topic molding is the metadata and the output is a model contain the topics of the data and a description of these topics and for each part of data in the big data we have the topic or topics it belongs to. For each data typed, a set of topics have been extracted, then in order to facilitate searching for data these topics are structured in a search tree. In search tree building step a tree is built for each data type. In order to search for data as a response for a user query, the search trees are integrated to form a global search tree.

A. Metadata Manager

The metadata manager(MDM) is one of the significant modules in the framework. The main purpose of this module is to store big data metadata. As a result of having metadata, we can access the needed part of the big data and perform specific analysis task instead of loading the whole big data for each task. With this metadata manager, the new framework is having several advantages such as: reduced the amount of data transferred between cluster nodes, increase task processing parallelism and decrease data loading time. The metadata manger consists of two components: Metadata Extractor and Metadata Repository.

1) Meta-data Extractor: The main objective of this component is to extract metadata of the big data and sent it to metadata store where it will be saved. The metadata extractor (MDE) is designed to handle and overcome big data 3Vs characteristics and complicated properties. We have utilized

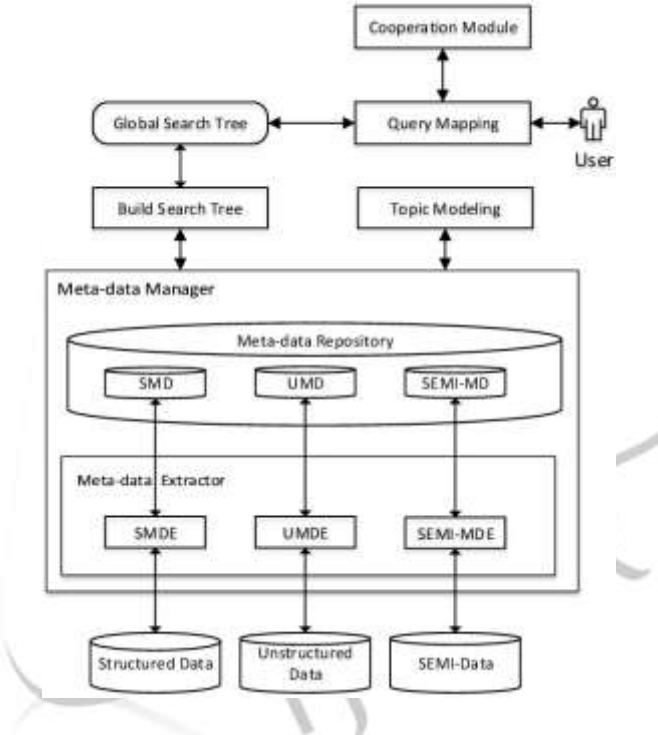

Figure 1. BigAnalytic Framework.SMDE: Structured Meta Data Extractor,UMDE: Unstructured Metadata Extractor, SEMI-MDE: Semi-structured Meta Data Extractor,SMD: Structured Metadata Extractor UMD: Unstructured Metadata SEMI-MD: Semi- structured Metadata

the mobile agent in our design of MDE where each agent is equipped with an appropriate metadata extraction algorithm that fits the data type and operation environment at each specific data node. The MDE creates and sends specific mobile agents to all data nodes. Each agent will send back the metadata to the MDE. After that, the MDE will pass these meta to the MDR where it will be stored. This is a onetime process at the initial deployment of the framework. These meat data will be used by the framework until a metadata update is needed. This usually tack place when we have significant change on the big data.

2) Meta-data Store: Metadata store(MDS) is where all extracted metadata are stored. While dealing with big data, huge metadata are expected as well. For instance, assuming that the metadata constitute 0.1% of the big data size, for big data of 1 exabyte then the size of metadata = 1 petabyte which is still considered as big volume of data. In order to improved metadata accessibility efficiency, we are using NoSQL[4], [5], [6], [7], [8], [9] data base to manage and store these metadata.

B. Topic Modeling:

Topic modeling (TM) is the framework component that classifies big data in to different topics. Big data holds a huge amount of data in different forms, web pages, data base tables, Facebook posts, twits on twitter, free text documents....etc.

Each analysis task will process a specific part of the big data. For instance, let's consider the big data of a logistics and supply chain company and the analysis task issued is: find expected customers for new electronic products. The big data manager will determine which part of big data (from all data sources and structures) is relevant to this specific task before running the analysis

task algorithms to mine for the needed information. Three components in our framework are needed to achieve this: Metadata Manager, Topic Modeling and Topic Tree Builder.

Topic modeling is a group of statistical algorithms that is applied on a data set in the purpose of exploring themes or patterns [10]. These algorithms don't require prior information about the patterns, they emerge as an output of the algorithm. The simplest and famous topic modeling algorithm is Latent Dirichlet Allocation (LDA)[11].A topic, them or pattern is defined as a group of words that exist together with high probability in a corpus. For example, computing topic include words like: computer, hard drive, processing, Laptop, RAM, speed ....etc. A text includes any of these words can be classified as a computing topic and as more computing words found in the text the more its pertinence to computing topic Many tools developed in order to implement topic modeling such as: MALLET[?],Matlab Topic Modeling Toolbox[?] and Stanford Topic Modeling Toolbox[?]. Topic modeling is a handy tool for text annotation especially in large text data sets.

The exploitation of TM in our framework includes exploring topics and classifying big data instances into these topics. The stored metadata is the TM input, while the output is a set of topics and pertinence of each big data instance in to these topics. Let's consider the set of topics $T = t_1\{, t_2, t_3, \ldots, t_k\}$ where each member of this set is a topic and each topic t is represented by set of words; the number of topics k and number of words in each topic are specified by experiments. Each big data instance can be classified into one or more topics. The pertinence p of an instance i is calculated by TM.

In case an instance i classified into j topics, i pertinence is represented as $P=\{p_1, p_2, p_3,..., p_j\}$ where . A threshold can be specified to exclude topics with low relevance. By introducing TM within our framework design, we uncover a set of topics that run through the big data and get the pertinence of each instance in to these topics; this information will feed in to Topic Tree Builder framework component for the purpose of big data exploring efficiently.

C. Topic Tree Builder(TTB):

In order to achieving our goal stated earlier, determining relevant big data segments for analysis tasks. The MDM will extract and store metadata, TM will classify big data instances into themes or topics. Now we need to map the analysis task into the most relevant topics which is accomplish by the Topic Tree Builder(TTB) framework component. It is expected to get tens or maybe hundreds of topics as result of TM as well as massive amount of tasks. Therefore speeding up tasks and topics mapping is critical, so a new method built to enhance the system performance.

We have developed three algorithms in the TTB component: Build Topic Tree (BTT),Topics Similarity (TS) and Level Nodes Similarity (LNS). BTT consists of four major steps:

Algorithmus 1 Build Topic Tree.

$T : \{t_1, t_2, t_3, ..., t_k\}$. *T set of k topics,*
$level_{l=0}$ *l is # of levels in topic search tree.*
n_l is # of nodes in level l.
$\mathcal{T}$ threshold value to merge level nodes

```
1: input: set of topics T
2: var l = 0;
3: calculate similarity between all topics in T.
4: Build level l = 0 by assigning each topic in T to a node.
5: create level l + 1
6: while n_l > 1 do
7: calculate similarity between level l nodes.
8: Create new level l = l + 1;
9: end while
10: return tree
```

First, BTT start by calling TS algorithm to calculate similarity between all topics, second level l_0 is built by assigning each topic to a level node, fourth that using the similarity between topics BTT build level *1*. Finally, while a level *1* consist more than *1* node, BTT will start to build the topic tree in down up pattern. All topics are considered as leaf nodes at level 0. BTT will create a level at each iteration. The input to any iteration is all nodes at level l and the output is the nodes for level $l+1$. At each iteration, topics with similarity grater that a predefined threshold value τ ($sim(t_i,t_j)>\tau$) are merged into one node. The algorithm will reach the tree root and stops when the iteration results in to one node.

The TS algorithm calculate the similarity between every two topics t_i, t_j in the set of all topic T. Each topic t is represented as a set of fixed number of words n and each word w is associated with a value v that represent the importance of the word w in topic t. The words in a topic t are in descending order by the value v. The TS algorithm compare all words in ti, tj and if a words in ti match a word in t_j the sim (t_i, t_j) is increased by $t_i\,[n].v + t_j\,[m].v$ where n, m are the positions of the words in t_i, t_j.

D. Query Mapping:

The Query Mapping(QM) is an interface between the users and the BigAnalytic framework. First,QM receive the query generated by system user. After that, this query is converted in a suitable format for STB. Then the query, in the new format, is send to STB. The QM role in the BigAnalytic as broker between framework components.

E. Cooperation Module:

In case that ABMD is deployed in a multi-cluster environment, a mean for cooperating jobs that require data from more that one cluster is needed. The Cooperation Module (CO) sent a copy of a query received at a local cluster to all other clusters. Then,

CO receive the results from remote clusters and pass them to QM. Another role played by CM is when receiving a quire from other cluster. In this case, CM send this query to QM and when the answer is ready CO send it back to the original query owner.

Algorithmus 2 Topics Similarity(t_1, t_2)

$t_1 : \{(w_1, v_1), (w_2, v_2), (w_3, v_3), \ldots\ldots, (w_n, v_n)\}$ *n words in each topic, where each element (w,v) has two properties: the word w itself and a value v that represent the importance of a word w in topic t where:*

$t_1[1].value \geq t_1[2].value,$
$t_1[2].value \geq t_1[3].value,$
$\vdots$
$t_1[n-1].value \geq t_1[n].value.$

1: **input two topics** t_1, t_2
2: **for** i $= 1; j \leq t_1.length; i++$
3: **for** j $= 1; j \leq t_2.length; j++$
4: **if** $t_1.word_i == t_2.word_j$
5: $sim(t_1, t_2) = sim(t_1, t_2) + t_1.value_i + t_2.value_j;$
6: **end if**
7: **end for**
8: **end for**
9: **Return** $sim(t_1, t_2)$

Algorithmus 3 Level Nodes Similarity.

Each level l consists of a number m of nodes
Each node consists of a number of topics
$NodsSim[]$ is an arry containing similarity between to all nodes m in level l
x is the number of topics in node i n_i
y is the number of topics in node j n_j

1: **begin**
2: **var** $NodSim(n_1, n_2) := 0;$
3: for all nodes m in level l
4: **for** $i = 1; i \leq x; i++$
5: **for** $j = 1; j \leq y; j++$
6: $tempNodsSim[n_i, n_j] = tempNodsSim[n_i, n_j] + getsim(t_i, t_j)$
7: **end for**
8: **end for**
9: $NodSim(n_1, n_2) := \frac{tempNodesSim[n_i, n_j]}{x*y}$
10: **end for**

III. EVALUATION

In this section we present an evaluation to our framework a long with some comparisons with other big data management frameworks. The comparison is built in terms of the structure and nonfunctional characteristics.

A. Scalability and Fault Tolerance:

The BigAnalytic frame work is a dynamic solution, able to handle any big data regardless the size, number of node our data type. The number of agents can always be adjusted to handle any number of nodes or data types perfectly. This provides very high scalability compared with other data management frameworks. The framework is capable of handling any failures among the data node and providing high fault tolerance as well. In case of node failure, both, the original agent and the main frame will act to minimize the impact. Agents will keep sending intermediate results and insure updating system to minimize any failure impact. The interval of each intermediate result can be set based on the result size, time or both. On the other hand the system will send a new instant of the agent to the node with all backup data to prevent work repetition as much as possible.

B. Metadata Manager:

MDM is considered one of the important components in our framework which ease accessing data. It also increases task parallel execution and improved the overall performance. Acting as a first analysis & filtering layer, MDM explore the data and provide relevant parts only to the framework. In contrast to other big data managers such as MapReduce, it doesn't store a metadata and consequently execute each analysis task on the entire data. On top of reducing the processing time, the MDM executes tasks on one part of the data only. This gives the chance to the remaining tasks to run on other parts of the data and increase system parallelism.

C. Data Loading:

The proposed framework exploit mobile agent in executing big data analysis task. Consequently, there is no need to transfer data to remote machine for processing, instead process travels to the data location. This decreases the amount of data transfer; only intermediate results are transferred instead not the whole data. While, for instance, MapReduce framework require loading data before data processing tack place, BigAnalytic design eliminate data loading time and efforts to the remote machines.

D. Data Model Enforcement:

In our solution we are not enforcing any specific data format for the data to be processed. A specific agent is created for each data type within the big data. This save a significant amount of time and enhance system performance. On the other hand, the Bigtable data model in MapReduce framework stores the data where a full data conversion to the Bigtable data model is needed before performing any task.

E. Metadata Extraction Process:

Figure 2 shows how the components of the system interact in order to perform the metadata extraction process. The metadata extraction process implemented by the framework does the following:

1) Metadata extractor (MDE)send a copy of extractors to raw data(RD).
2) MDE send the extracted metadata to the metadata store.
3) Topic modeling (TM) read metadata from metadata store and perform topic modeling.
4) TM update metadata in metadata store.
5) Building search tree (ST) read metadata from metadata store.
6) BST run algorithms to build the ST.
7) BST up data metadata with the ST.

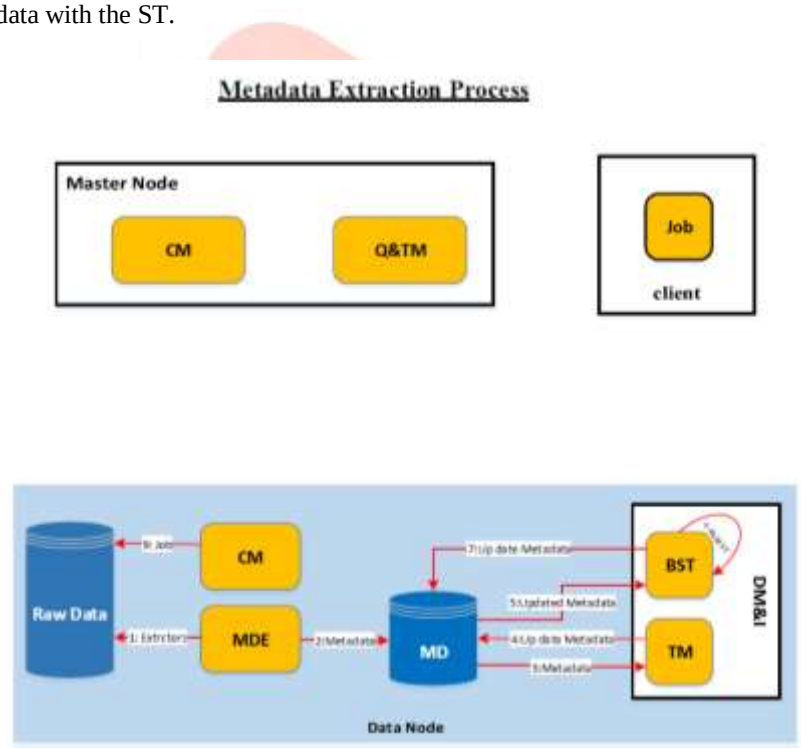

Figure 2. Metadata Extraction Process.

F. Knowledge Discovery Process:

Figure 3 shows how the components of the system interact in order to perform the knowledge discovery process. The knowledge discovery process implemented by the framework does the following:

1) A client send a processing job.
2) Query and Task Mapping component send a data query to Cooperation Module (CM).
3) CM broadcast the data query to all DN.
4) DN's Cooperation module send data query to Data Mapping and Indexing(SM&I) module.
5) DM&I replies with the location of relevant data to local CM.
6) DN's CM send location to Master CM.
7) Q&DM map send the job to CM.
8) Master CM send a copy of the job to all DNs that has relevant data.
9) DN's CM Launch the job using RD.
10) When job is finished, DN' CM receive locale task result.
11) DN's CM send locale result to master CM.
12) Master CM aggregate results and send final result to Q&TM.

13) Q&TM Map final result and send it to the user.

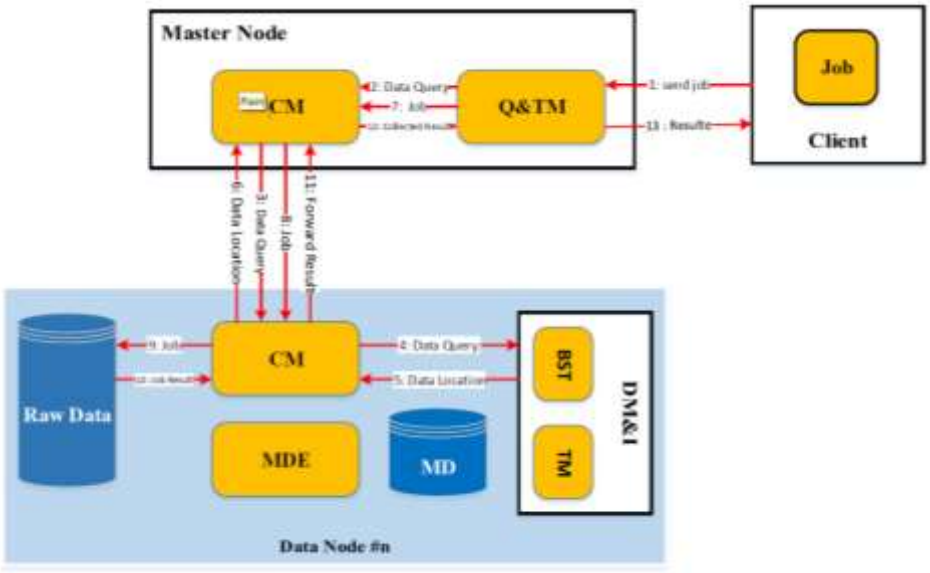

Figure 3. Knowledge Discovery Process.

IV. RELATED WORK

SCOPE (Structured Computations Optimized for Parallel Execution) is a SQL like scripting language [12]. It is designed for big data analysis executed using computer clusters. In addition, the implementation complication of tasks implementation in distributed environment are transparent using SCOPE. The data is represented as rows with set of attributes. A group of similar rows has a defined schema. Microsoft use SCOPE for big data mining and analysis tasks.

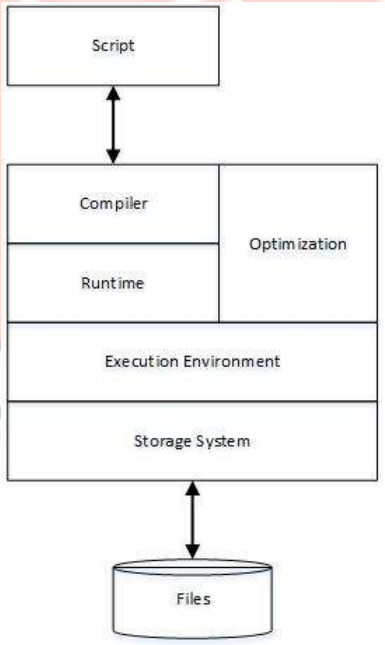

Figure 4. Cosmos Platform Structure [12].

SCOPE is a declarative and extensible language. The compiler is responsible of the underlining execution complexity of query and and optimizer is responsible of finding the best query execution plan. In addition, user defined functions are visible resulting in flexibility in manipulating data.

COSMOS is a distributed computing platform developed by Microsoft [12]. It is used for managing and analyzing big data. Figure 4 depict the framework components.

- *Cosmos storage:* is the subsystem for storing huge amount of data. It is an append-only file system where only append writes are executed. Concurrent write operations are serialize by the storage subsystem. Replicating data on different servers is the fault tolerance technique. A file is a group of extents and each extent is a few hundred megabytes in size. In tern each extent consist of a number of blocks and each block typically a few megabytes in size.

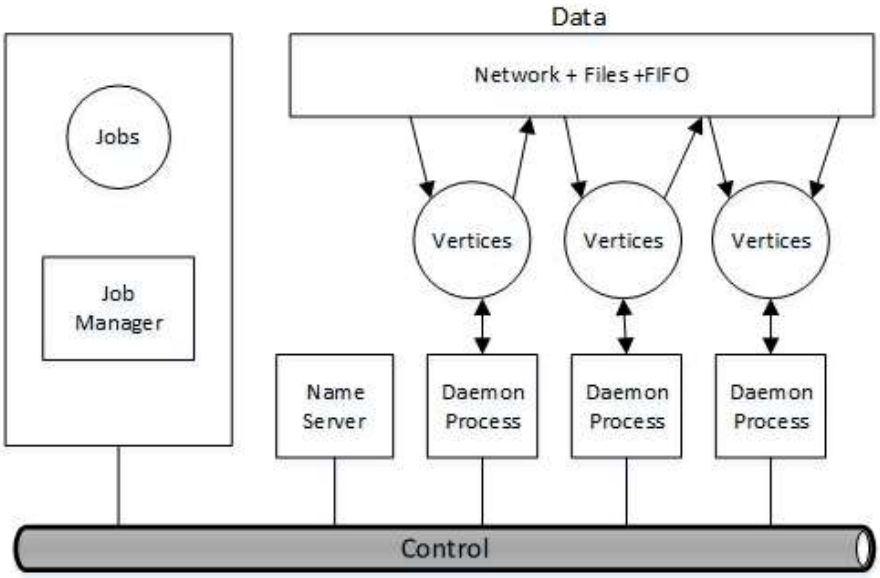

Figure 5. Dryad System Structure [13].

- *Cosmos execution environment:* An environment for deploying, executing, and debugging distributed applications. It consists of two components interface and a run time system. The interface is high-level language interface whereas the run time system handles the details of optimization, fault tolerance, data partitioning, resource management, and parallelism. A directed cyclic graph (DAG) is the data model for applications with edges representing data flows and vertices representing processes. The job manager (JM) is the coordination process that coordinate vertices execution.
- *SCOPE:* is a programming language that use C# expressions and SQL capabilities. A typical SCOPY script take raw-set data as input, perform data processing and transformation operations and the produce output.

Dryad system structure is shown in figure5. Job manager (JM) is the system process for coordinating the execution of the user jobs. It contain the job source code and it resides either at user workstation or at the cluster. The data send and received by JM is only control messages so it is not a bottleneck to the system. The name and the location within the network of all computers available at the cluster are registered at the name server (NS). A process called daemon (D) is created at each cluster computer. This process is responsible of creating processes at each computer and communicating between the JM and vertices V. At first job execution, the vertices(V) code is sent from the job manager to the D process. Google created a programming model for processing big data [14]. This model, called MapReduce, follow divide and conquer method in handling big data processing. The model consists of two functions namely Map and Reduce. In the Map function, the problem is sub divided into many small sub-problems that take data as key/value pair, process the data and generate intermediate key/value pair output. In tern, the Reduce function process this all intermediate output related to the same key/value pair and produce the final output. MapReduce run on clusters composed of hundreds, even thousands, of commodity PCs. The frame work hide the execution details, such as task scheduling, flat tolerance, location of data, from the programmer.

Despite MapReduce fame and wide spread among all other large-scale data analytic frameworks, it suffer from a number of limitations [15], [16], [17]:

- *Data Access:* is a critical task for efficient and high performance of execution of a job. For cretin types of big data analytic tasks, access of input data of only some data nods or even accessing a selected data in those data nods is needed. MapReduce implement a brute force access for all input data at each job execution, which de- grades the system performance. Hadoop++[18], HAIL [19] are two indexing system proposed to resolve the data access limitation of MapReduce, A another approach to resolve data access problem is by using data layouts. A survey of data layouts approaches is in [20].A column file approach, where data is partitioned vertically and then grouped based on correlated columns, is proposed in Llama[21] and Cheetah[22]. A combination between vertical and horizontal partitioning of data is introduced in RCFile[23].
- *High Communication Cost:* The output of the Map task is send to the Reduce nodes. The size of the transmitted data depend on the type of input data and type of analytic job. As the size of data increase, drain the network throughput and decrease overall system performance. CoHadoop [24] partition data and place correlated data in same node intentionally.
- *Redundant and wasteful processing:* in MapReduce, jobs the use the same input data and perform similar sub tasks are not allowed to share results, resulting in redundant data processing and wasting processing power. Techniques such as sharing results [25], [26], [27], queries batch processing [28] and Incremental job processing [29], [30], [31] were proposed to overcome this problem.

Moreover, other MapReduce shortcoming such as re- computation, lack of early termination, Lack of iteration, Quick retrieval of approximate results, load balancing, lack of inter- active or real-time processing and Lack of support for n-way operations were reported[17].

V. CONCLUSION AND FUTURE WORK

In conclusion, in this paper we introduce a new a big data manager. This manager contain a key architectural components such as MDM and utilize software agent technology. As a result, improvements in terms of performance, scalability, fault tolerance and network utilization are noticeable.

This is a part of an ongoing large project. A prototype and experiments are ongoing for the BigAnalytic. In addition, another version of BigAnalytic, that include more distribution of work done by the framework components, is also under development.

REFERENCES

[1] J. Gantz and D. Reinsel, "Extracting Value from Chaos State of the Universe: An Executive Summary," IDC iView, no. June, pp. 1–12, 2011. [Online]. Available: http://idcdocserv.com/1142

[2] D. Laney, "3d data management: Controlling data volume, velocity and variety," Gartner., 2001.

[3] J. Manyika, M. Chui, B. Brown, and J. Bughin, "Big data: The next frontier for innovation, competition, and productivity," no. June, 2011. [On- line]. Available: http://www.citeulike.org/group/18242/article/9341321

[4] R. Cattell, "Scalable SQL and NoSQL data stores," ACM SIGMOD Record, vol. 39, no. December, p. 12, 2011.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. a. Wallach, M. Bur- rows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," 7th Symposium on Operating Sys- tems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA, pp. 205–218, 2006.

[6] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database," in 2011 6th International Conference on Pervasive Computing and Applications, 2011, pp. 363–366.

[7] R. Hecht and S. Jablonski, "NoSQL evaluation: A use case oriented survey," in Proceedings - 2011 International Conference on Cloud and Service Computing, CSC 2011, 2011, pp. 336–341.

[8] S. A. Clarence J M Tauro, Aravindh S, "Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Databases," International Journal of Computer Applications, vol. 48, no. 20, pp. 1–4, 2012.

[9] M. Seeger, "Key-value stores: a practical overview," Computer Science and Media, pp. 1–21, 2009.

[10] D. Blei, L. Carin, and D. Dunson, "Probabilistic topic models," IEEE Signal Processing Magazine, vol. 27, no. 6, pp. 55–65, 2010.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, vol. 3, no. 4-5, pp. 993–1022, 2012.

[12] R. Chaiken, B. Jenkins, and P. Larson, "SCOPE: easy and efficient parallel processing of massive data sets," Proceedings of the VLDB Endowment, vol. 1, no. 2, pp. 1265–1276, 2008.

[13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," ACM SIGOPS Operating Systems Review, vol. 41, no. 3, pp. 59–72, 2007.

[14] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[15] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," Proceedings of the 35th SIGMOD international conference on Management of data, pp. 165–178, 2009.

[16] B. Y. J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," Communications of the ACM, vol. 53, pp. 72–77, 2010.

[17] C. Doulkeridis and K. Nørvåg, "A survey of large-scale analytical query processing in MapReduce," VLDB Journal, vol. 23, no. 3, pp. 355–380, 2014.

[18] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)," in Proceedings of the VLDB Endowment, vol. 3, no. 1-2, 2010, pp. 515–529.

[19] J. Dittrich, J.-A. Quia e Ruiz, S. Richter, S. Schuh, A. Jindal, O. Schad, J. A. Q. Ruiz, S. Richter, S. Schuh, A. Jindal, and J. Schad, "Only aggressive elephants are fast elephants," in PVLDB: Proceedings of the VLDB Endowment, vol. 5, no. 11, 2012, pp. 1591–1602.

[20] J. Dittrich and J.-a. Quian, "Efficient Big Data Processing in Hadoop MapReduce," in Proceedings of the VLDB Endowment, vol. 5, no. 12, 2012, pp. 2014–2015.

[21] Y. Lin, D. Agrawal, C. Chen, B. C. Ooi, and S. Wu, "Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework," Proceedings of the 2011 international conference on Management of data - SIGMOD '11, p. 961, 2011. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1989323.1989424

[22] S. Chen, "Cheetah: A high performance, custom data warehouse on top of mapreduce," in Proceedings of the VLDB Endowment, 2010, pp. 1459–1468.

[23] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu, "RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems," in Proceedings - International Conference on Data Engineering, 2011, pp. 1199–1208.

[24] M. Y. Eltabakh, Y. Tian, F. Özcan, R. Gemulla, A. Krettek, and J. McPherson, "CoHadoop: flexible data placement and its exploitation in Hadoop," in Proc. VLDB Endow., vol. 4, no. 9, 2011, pp. 575–585.

[25] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas, "MR- Share: sharing across multiple queries in MapReduce," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 494–505, 2010.

[26] P. Agrawal, D. Kifer, and C. Olston, "Scheduling shared scans of large data files," Vldb, vol. 1, no. 1, p. 11, 2008.

[27] Y. N. Silva, P. A. Larson, and J. Zhou, "Exploiting common subex- pressions for cloud query processing," Proceedings - International Conference on Data Engineering, vol. 1, pp. 1337–1348, 2012.

[28] A. Aboulnaga and D. I. Elghandour, "ReStore: Reusing results of MapReduce jobs," Vldb, vol. 5, no. 2012, p. AESNP3, 2012.

[29] P. Bhatotia, A. Wieder, R. Rodrigues, U. a. Acar, and R. Pasquin, "Incoop: MapReduce for incremental computations," Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC '11, pp. 1–14, 2011.

[30] B. Li, "A Platform for Scalable , Low-Latency Analytics using MapRe- duce," Doctoral Dissertations 2014-current. Paper 378, pp. 1–186, 2015.

[31] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "SCALLA: a platform for scalable one-pass analytics using MapReduce," ACM Transactions on Database Systems, vol. 37, no. 4, pp. 1–43, 2012.