Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 50 (2015) 517 – 523

Procedia Computer Science

2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Survey on Programming Models and Environments for Cluster, Cloud, and Grid Computing that defends Big Data

J. Christy Jackson[a], V. Vijayakumar[b], Md. Abdul Quadir[c], C. Bharathi[d]

[a]*J.Christy Jackson, VIT UNIVERISITY-CHENNAI 600063,INDIA*
[b]*V. Vijayakumar, VIT UNIVERISITY-CHENNAI 600063,INDIA*
[c]*Md.Abdul Quadir, VIT UNIVERISITY-CHENNAI 600063,INDIA*
[d]*C.Bharathi, VIT UNIVERISITY-CHENNAI 600063,INDIA*

Abstract

A collection of interlinked stand-alone computers which function together in unity as a single incorporated computing resource is a kind of parallel or distributed processing systems called as the cluster. Clusters and grids are systems which intercommunicate among them and act as a single resource. This type of functionality can be referred to as the multi-computer parallel architecture that runs on certain considerations. Nevertheless cloud computing came forth as a more illustrious programming model to handle large data sets using clusters. A programming model is nothing but how data is carried out for handling the application. Performances, portability, objective architecture, sustainment of code are key measures that have to be commemorated while designing a programming model. Applications which are meant for data analytics generally deal with large data sets that undergo several stages of processing. Some of these stages are performed consecutively, and the others are carried out in parallel on clusters, grids, and cloud. This paper portrays a survey on how programming models which are developed for cluster cloud and grid act as a support for big data analytics. In addition, study on programming models which are currently being employed by leading multinational companies are pictured in the paper.

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Keywords:

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)
doi:10.1016/j.procs.2015.04.025

1. Introduction

Breakthroughs in industrial innovations, next generation scientific discoveries will depend on the capability of systems to analyse on the large volumes of data available. As these masses of information grow rapidly, the challenge faced would be on how to manage these large chunks of data, which in turn causes an increase in complexity of Data Life Cycle management (DLM). DLM comprises of various operations such as transferring, archiving, replication, processing, and deletion. Results are required to automate and enhance data management operations in order to ease the complexity of Data Life Cycle. It is observed that Data Life Cycle is influenced by two constraints. The first constraint being the operations on data which are extracted from the users and applications and the second constraint is the infrastructure itself. The second challenge banks on the fact that data are distributed across variety of systems and infrastructure and not just one single infrastructure [9]. Hence Big Data Applications have to be able to coordinate several systems which address the data and also has to deal with consequences related to data and the events happening. This paper portrays around the second constraint, which is infrastructure itself and is scripted with a complete analyses on programming models and environments which support Big Data infrastructure.

2. Programming Models

Programming model can be determined as the stream and performance of the data manipulation for an application. Execution, portability, target architectures, ease of sustenance code revision mechanisms are some of the things to be considered while developing a programming model. More often than not some of these factors have to be compromised for service. A typical example would be trading computation for storage or for communication of data is an extremely common algorithmic manipulation. These complications can be overcome by using parallel algorithms and hardware. Surely, an application developer or the programmer may have multiple cases of similar algorithm to allow for various performance tuning on different varieties of hardware architecture [4].

Hardware architectures these days are small and high-end high performance computer which form clusters with versatile communications, interconnection technologies and with nodes having processor greater than one. An Illustration for this architecture can be the Earth simulator which is a cluster of enormously powerful nodes with multiple vector processors and with prominent IBM space installation. These simulators have multiple nodes with 4, 8, 16, 32 processors each [4]. The issue arises when a situation originates the requirement of selecting a programming model that acquires the data in the correct place when computational resources are useable. As the number of processors grows, this problem becomes more complex. Scalability is the term employed to suggest the performance of an algorithm, method, or code, relative to single processor. The scalability of an application is principally the consequence of the algorithms capsuled in the programming model which is used in the application [4].

2.1 Active Data Programming

The life cycle of data is the track of operational stages through which data passes from the time they enter the system until they leave the system. Amongst these two points in time, data actually passes through various stages of development. Migration, archiving, duplication, transfer are some of the stages through which data mostly passes. Active Data Life Cycle management is a programming model which assists the developers in writing of applications which implement data life cycle management [9].

2.1.1. Prototyping Life Cycles

Active Data is a mode to model life cycles. This life cycle model is primarily based on Petri networks. Petri Networks are a formal graphical tool extensively used for the analysis of systems with concurrency and resource sharing. Each data item in the data set is tagged with a state. With respect to these states all possible data states with places are represented. As it is common for distributed systems to deal with data reproduction, a petri net

token constitutes a single replica of the same data item. The figure below shows the active data model for the write once read many life cycles.

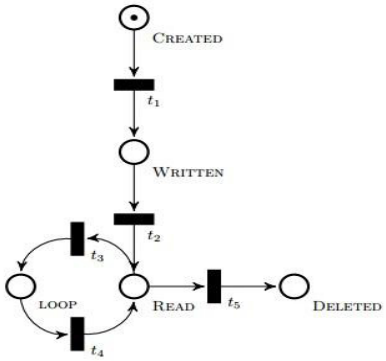

Fig 1: Active data model for the "write once, read many" life cycle [9]

As an example, as shown in the figure a data item starts in the CREATED stage. The only data replica, represented by a single token can be written only once. From the READ stage the token can loop through t_3 and t_4. This means that the data item can be read multiple times. Eventually, the token can pass through t_5 and end the life cycle in the DELETED stage.

2.1.2 Scheduling Application

Scheduling the applications is a critical concept in data transitions. Precious information can be gathered from transitions in Active programming model. The Active Data runtime offers an Application Programming Interface (API) that permits programmers to code in the environment. The supplied code is then executed and whenever there is a typical data transition triggered. Handler is the term given for the code. The usual sequences of events that lead to the execution are described below. The programmer writes and compiles code into an API which is nothing but the Active Data Handler. The Programmer then uses the client API on machine 1 to record the handler for a particular transition named as't'. Concurrently a remote system named as machine 2 sues or alters data and varies its state matching transition't'. The remote program then sends a report to the runtime stating that the transition't' has happened. Following that the user given code is run in machine 1. The final two steps are looped as long as the handler subscription persists, every time the transition is triggered, on any data, on any node manipulating data.

2.2 Grid Programming Model

To start off with, identifying and allocating grid resources, channelizing the data as well as the program, and launching the application is complicated as well as it is only partially standardized. It is noticed that large scale grid computing has a practically more applicability. Satin is a programming model that is distinctively designed for grids. This programming model for grid address the primary issues in such a way that it conceals much of the complexity of a grid, it can be followed by efficaciously on a grid, and it can be utilized for a moderately broad range of applications

The programming model is practically easier to use than message passing and other models. It has been implemented in a system called as Satin, which is used by many applications such as the N-body simulations, SAT-solvers, VLSI-routing, grammar induction, and many others, and has been run efficiently on heterogeneous grids with up to 1000 processors [7].

2.2.1 Implementation of Satin

Marker interfaces are used in satin and there is no use of language extensions. Java compiler (.javac) is used for compiling Satin programs. The output is a accumulation of class files which contain byte code that can be accomplished consequently on any JVM. The reason for doing so is that it is useful for testing and benchmarking purposes. An extra compilation step is carried on in the generated byte code to run in parallel. The Satin compiler, satinc, rescripts the sequential bytecode, inserting code to enforce the spawn and sync operations. The rescripted bytecode can be deployed on the grid with any grid middleware system, such as Globus [3] or our Java Grid Application Toolkit [7]

3. Survey

The fact that processors these days are considerably faster than the ones a decade ago, the speed of accessing data on disks or volumes was still lesser than the CPUs. Similarly applications that required to process large amounts of data did not benefit much from the increased speed. However programming languages these days assist multi-programming concepts such as multi-threading executions such as libraries for C++ and built in functionalities in Java. Parallel programming model takes the reward of servers with multiple CPUs and multiple cores by cutting down the time taken to process the data.

3.1 Data Parallel Programming Model

Data parallel programs generally employ large datasets either to stack up large input data in application specific structures or to lend in computing the output. However in both the scenarios the data is disseminated onto processors to decrease the applications running time. The input is usually a file holding geometrical data (coordinates and connecting property information) reporting a discrete physical domain. The output data are the values from unknown property information such as fluid pressure, temperature etc. which are worked out for the given input data [2].

The system primarily deploys a predefined representation of the unstructured mesh data which is used for the application input. Through the sub domain component the user has access to partition of this data using the local instance. UserData is a system providing representation of the generic user data that takes part in non-trivial data parallel computations. UserData is sub classed by the user and also provides access functions to read and write a specific location. Instances of this specified object are called as the distributed user object [2].

The Subdomain data is does not alter during the life time of the application. Therefore, no read/write dependencies occur for input data stored by an application. Hence it can be seen that a computation require the Subdomain data does not involve consistency [2].

3.2 Distributed Data Parallelism

Computations on the data are relatively simple for batch processing of log and text files for web crawling and page analysis. However when the size of input data increases these computations have to be distributed across several machines in order to reduce the time in completing the job. To accomplish this task a large amount of code has to be written to run the programs in a distributed environment. Distributing the data to multiple nodes, parallelizing the computations, and handling co-ordination of load balancing and failures have to be considered while creating a distributed environment. This entire process was simplified by Jeffrey Dean and Sanjay Ghemawat from Google and created a programming model called as the MapReduce [3].

3.3 MapReduce

MapReduce is a patented data parallel model framework introduced by Google to support distributed computing on large datasets as well as cluster of computers [14]. MapReduce is designed to reduce framework and has two computations namely map and reduce. Map, described by the user takes a pair of input and gives a set of intermediate key/value pairs. The MapReduce library combines all intermediate values related to the same intermediate key and transfers it to the reduce function. The reduce function on the other hand is also described by the user, consents an intermediate key and a set of values for that key. It fuses together these two values to build a potential smaller set of values. The intermediate value is then furnished to the user reduce function through an integrator. Addressing lists of values that are excessively big to fit in memory are done in this way [13].

In spite MapReduce having a very simple concept for its programming model falls short in few places. Incorporating MapReduce in some of the mainstream programming such as Java, C++, or python is a complex procedure. Adapting to other applications when one-input two phase data flow is inflexible. Unintelligible nature of the map and reduce functions blocks optimization. Despite falling short in all this aspects MapReduce makes many of the distributed environment issues easier to code. Apache in Hadoop Project implemented MapReduce programming model and gained much popularity [14].

3.4 Hadoop

The Apache Hadoop is an open source software project patronized by Apache Software foundation. A simple programming model, apache Hadoop library allows the distributed processing of large data sets across clusters of computer. The programming model is planned in such a way that it can be scaled up from a single server to a thousand machines which can have a local storage in each one of them. In addition, rather than relying on hardware to deliver high-availability, failures at the application layer are identified by the HDFS library. Hence HDFS is a extremely usable service that can be installed on clusters of computers, which do not have anything built to minimize the failure rate [12].

The engine comprises of one JobTracker, to which client applications render MapReduce jobs and multiple TaskTrackers. Non-busy TaskTracker are assigned jobs by the JobTracker while keeping the work as close as possible to the data. Results are then collected by running the TaskTrackers.

3.5 PIG and Pig Latin

The primary reason for scheming Pig was to make Hadoop more accessible and useable for non-developers. Pig is a synergistic, script based, execution environment defending Pig Latin, a language used to carry data flows. Pig Latin language endorses the loading and processing of input data with a series of operators that translate the input data and acquire the necessary output. The Pig execution environment has two modes such as local mode and Hadoop. Local mode runs all the scripts and does not require Hadoop MapReduce and HDFS. On the other hand Hadoop are run on separate Hadoop Cluster. Data are retrieved in an abstract way focuses on the data and not the structure of custom software program. Prototyping using Pig is very simple [5].

Pig Framework executes a sequence of MapReduce jobs which are got from compiling a pig latin program. Operations namely load, for each, filter, can be carried out as map function and operations such as group, store can be carried out as reduce function.

3.6 Hive and HiveQL

Services such as summarization, ad-hoc queries and analysis are provided by Hive, a data warehouse system built on Hadoop. HiveQL is a declarative language very similar to SQL. Programs that run on Hive are developed using HiveQL. Most SQL characteristics are backed up and also some of the custom MapReduce scripts by HiveQL. HiveQL statements which are got from HiveQL compiler are engineered into an acyclic graph of MapReduce jobs

which are later passed on to Hadoop for execution [11]. Hive and HiveQL are practicable for ad-hoc querying of very large datasets. The resemblance of HiveQL and SQL makes it more prosperous for mainstream programmers and analyst to use.

4. Analysis

Analysing large data sets which are related to web were exercised comfortably with the MapReduce programming model and its related languages. Even though it has a rich features associated with them there are several short comings. Below mentioned are few of the issues faced in MapReduce and Hadoop

- It follows the principle of Map and only then reduce functionality. This functionality is not suitable for applications that update databases or generate data.
- The data actually passes through multiple stages which results in a complex jobs pipeline and multiple stages
- Error margin is very poor between stages.
- Assumption is made that Map has data available all the time, it does not expect data to be generated
- The output of Reduce is stored in two network copies and three disks which requires extra space/device
- It compounds or "Joins" the input of different types which results in redundant type conversion
- Similarly "Split" makes outputs of different types
- Coding is despicable
- Performance penalty is difficult to stave off
- Few Merge and Joins are high-priced
- Does not fit non-batch applications

To tackle complex issues faced by Map and Reduce jobs there are few solution which are developed namely Dryad and DradLINQ

4.1 Comparison between Dryad and MapReduce

Dryad executes in the execution layer whereas MapReduce executes in both the execution as well as the application model. Jobs are finished by Arbitrary DAG for dryad and for MapReduce it takes a combination of Map, sort and Reduce. The plug in policies varies slightly in both the programming model. Dryad has programs which are graph generated and for MapReduce it is Map and Reduce. Dryad has a fairly new maturity which is less than two years on the other hand MapReduce is has a well renowned maturity model (> 4 years). Deployment is minimal for dryad an in contrary MapReduce has a wide spread deployment. Since dryad is a product of Microsoft it is being distributed internally only and is not available either as open source or commercially. MapReduce is distributed by Hadoop

5. Conclusion

In conclusion, Big data is hard to manage, in other words data is generated everyday but large scale computing of those commodity computers is difficult. 10K core clusters are available but it is hard to program 10k co-occurring threads. Analyzing petabytes of data is difficult. To solve some of the issues mentioned in the article several solutions have been enforced and partially implemented. The most productive programming model is MapReduce/Hadoop which determines the read-only datasets that can be read, split, and analyzed using Map and Reduce operators. Though MapReduce/Hadoop proves to be the most productive it has certain consequences which are overpowered by new languages or language extensions such as Pig Latin, HiveQL. They render a less complicated programming model and environment for analyzing large quantities of data on clusters of commodity

computers. As a result these new and mightier languages and runtime will make Big data processing less problematic.

REFERENCES

[1] Attebury, G., Baranovski, A., Bloom, K., & Bockelman, B. (2009). Hadoop Distributed File System for the Grid. *IEEE Nuclear Science SymposiumConferenceRecord*, 1056-1061. Doi: 9781-4244-3962-1/09

[2] Diaconescu, R., & Conradi, R. (2002). A Data Parallel Programming Model Based on Distributed Objects. *IEEE*. Retrieved from 0-7695-1745-5/02

[3] Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems. In IFIP. *International Conference on Network and Parallel Computing* Retrieved from LNCS 3779

[4] Hou, K., Zhang, J., & Li, J. H. (2012). Review of Data-parallel Programming Model. *International conference on Computer science and Education*, *4*(28), 629-633. Retrieved from 978-1-4673-0242-5/12

[5] Hurwitz, J., Nugent, A., Halper, F., & Kaufman, M. (2011, January). *Hadoop Pig and Pig Latin for Big Data - For Dummies*. Retrieved from http://www.dummies.com/how-to/content/hadoop-pig-and-pig-latin-for-big-data.html

[6] Jagannath, V., Yin, Z., & Budiu, M. (2011). Monitoring and Debugging DryadLINQ Applications with Daphne. *2011 IEEE International Parallel and Distribute Processing Symposium*, *1530-2075/11*, 1266-1273. doi:10.1109/IPDPS.2011.268

[7] Kendall, R. A., Sosonkina, M., Gropp, W. D., Numrich, R. W., & Sterling, T. (n.d.). Parallel Programming Models Applicable to Cluster Computing and Beyond.

[8] Moscovich, E. (2012). Big Data for Conventional Programmers. Retrieved From http://www.ca.com/~/media/Files/About%20Us/CATX/big-data- forconventional-programmers-moscovich.pdf

[9] Nieuwpoort, R. V., Wrzesinska, G., Jaacobs, C. J., & Bal, H. E. (n.d.). Satin: A High-Level and Efficient Grid Programming Model. *ACM Transactions on Programming Languages and Systems*, *10*(10), 1-40. Retrieved from ACM 0164-0925/20x/0500-0001

[10] Simonet, A., Fedak, G., & Ripeanu, M. (2012). *Active Data: A Programming Model for Managing Big Data Life Cycle* (1). Retrieved from informatics mathematics website:http://hal.inria.fr/docs/00/72/90/02/PDF/RR-8062.pdf

[11] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., & Murthy, R. (2010). Hive? A Petabyte Scale Data Warehouse Using Hadoop. *IEEE*, 996-1005. Retrieved from 978-1-4244- 5446-4/10

[12] Tomasic, I., Ugovsek, J., Rashkovska, A., & Trobec, R. (2012). Multicluster Hadoop Distributed File System. *Mipro Croatian society*, 301-305.

[13] Yang, X., Liu, Z., & Fu, Y. (2011). MapReduce as a Programming Model for Association Rules Algorithm on Hadoop.

[14] Yang, G. (2011). The Application of MapReduce in Cloud Computing. *International Symposium on Intelligence Information Processing and Trusted Computing*, 154-156. doi:10.1109/IPTC.2011.46