Data Management in Data Intensive Computing Systems - A Survey

Mayuri K P
Department of Computer Science & Engineering
Sir MVIT, Bangalore, Karnataka, India

Mohamadi Ghousiya Kousar
Department of Computer Science & Engineering
Sir MVIT, Bangalore, Karnataka, India

Dr. Shakti Mishra
Department of Information Science & Engineering
Sir MVIT, Bangalore, Karnataka, India

Abstract

Data intensive computing is a class of parallel computing applications for processing large amount of data such as big data. In order to store, manage, access and process vast amount of data available on internet, data intensive computing systems are needed. These systems also enable to search, analyze, mine and visualize the structured and unstructured data. This paper mainly focuses on the different data intensive computing systems such as Supercomputing, Hadoop Map Reduce and Hadoop Distributed File System (HDFS), High Performance Computing Cluster (HPCC) and Grid Computing regarding how the big data is managed in each architecture and a comparative study is made on the characteristics of data intensive computing systems for each computing strategy.

Keywords: Data Intensive Super Computing (DISC), Map Reduce, HDFS, HPCC, Grid Computing

I. INTRODUCTION

Parallel processing of data is achieved through the Data intensive computing system which involves partitioning or sub dividing the data into multiple segments that can be processes independently in parallel on an appropriate computing platform, then reassembling the results to produce the final output data. The fundamental challenges for data intensive computing are managing and processing exponentially growing data volumes.
The common characteristics of data intensive computing systems are [2]:

A. Co-Location Of Data And Programs:

To achieve high performance in data intensive computing, it is important to minimize the movement of data. Data intensive computing uses distributed data and distributed file systems (DFS) in which the data is located across a cluster of processing nodes and instead of moving the data, the program or an algorithm is transferred to the nodes where the data is residing. This principle is called "move the codes to the data". This is extremely effective because program size is usually small than the large data sets processed by the data intensive systems and results in much less network traffic since data can be read locally instead across the network for e.g. ECL (Enterprise Data Control Language) in Lexis Nexis HPCC systems supports both local (operations are performed on data local to node) and global (operations performed across nodes) modes.

B. Programming Model Utilized:

Data intensive computing systems utilizes machine independent approach in which applications are expressed in terms of high level operations on data. If the programming model is machine dependent, that requires specialized software packages, low level programmer processing control and the node communication using conventional programming languages will add complexity to the parallel programming tasks and reduces programmer productivity.

C. Focus on Reliability And Availability:

Data intensive computing systems are designed to be fault resilient. This requires maintaining redundant copies of all data files and the intermediate results on the disk. The node and processing failures has to be automatically detected.

D. Inherent Scalability of hardware and software architecture:

Data intensive computing systems are scaled linearly for accommodating large volumes of data. In order to achieve BORPS (Billions of Records Per Second) processing rates additional processing nodes has to be added to system configuration which will meet the time critical performance requirements.

II. SUPER COMPUTING

Data intensive super computing systems (DISC) are open source to perform large-scale computations over the data and maintain continually changing data sets.

DISC supports sophisticated forms of computation over its structured data where storage and computation are co-located. These systems are operated using a batch processing style. With DISC, web search can be possible without language barriers so that user can type a query in any language, the relevant document across the World Wide Web are retrieved by the engine in all languages and translates them into user preferred language. This can be possible only through sophisticated statistical language models and translation algorithms [1].

DISC Systems utilizes the data which is stored in separate repository or servers and transfer the data to the processing systems for computations. This leads to more network traffic because large data sets are transferred for computation. A conventional super computing system utilizes machine independent programming models. This requires low level programmer processing control and node communication using conventional programming languages and specialized software packages. This adds complexity to the parallel programming tasks and reduces the programmer productivity [2].

One of the issues in the machine dependent programming model is it requires significant tuning and is more susceptible to single points of failure.

Scalability can be predicted by periodically check pointing the state of the program. Serious failures requires bringing the machine down , running diagnostic tests, replacing failed components and only then restarting the machine. This is an inefficient, non-scalable mechanism, not suitable for interactive use [1].

Reliability mechanism ensures data integrity and availability as part of the system function by contrast current super computer centers provides short term, large scale storage to their users, high bandwidth communication and plenty of computing power, but they provide no support for data management.

III. HADOOP MAP REDUCE

Hadoop is an open source large-scale data processing framework which increases its flexibility and adaptability to many problem domains and uses simple machine dependent programming models to process large chunks of data in a distributed way. Hadoop offers easy installation, flexible configuration based on commodity hardware. The programmer productivity can be improved using add-on capabilities such as pig, HBase, and Hive etc [2]. Google introduced the Map Reduce software framework in 2004 which is then adopted by Apache Hadoop that contains “Map” and “Reduce” phases. Map Reduce programming consists of writing two functions, a Map function and a Reduce function. A Map function takes a key-value pair as an input and outputs a list of intermediate values with the key [5][3]. The Reduce function takes the output of Map function and do some process like combining the values to generate the desired results in an output file.

Hadoop Map Reduce works in a master/slave fashion that leads to workstations, so they work in parallel that enables faster execution of tasks. Hadoop with its efficient data mining technique and its programming framework based on the concept of mapped reduction is a powerful tool to manage the big data. Using the knowledge discovery from the big data it is easy to get the information from the complicated data sets [4] .

Some of the challenges for Hadoop adoption include:

- Lack of flexible resource management.
- No support for application deployment.
- Multiple data source availability is less.
- Architectural bottlenecks exist in the Hadoop implementation leads to delays in scheduling Map Reduce tasks.

IV. HADOOP DFS

DFS provides access to the clients and process the data stored on the server ,so which is called as client/server-based application and it behaves as if the data is on their own computer. A distributed file system provides the following types of services:

Storage service:Provides a logical view of the storage system by allocating and managing the space on a secondary storage device.

True file service:Includes file-sharing semantics, file-caching mechanism, file replication mechanism, concurrency control, multiple copy update protocol etc.

Name/Directory service:Responsible for directory related activities such as creation and deletion of directories, adding a new file to a directory, deleting a file from a directory, changing the name of a file, moving a file from one directory to another etc.

Distributed file system provides some of the desirable features such as Transparency on Structure, Access, Naming and Replication , User mobility, Performance, Simplicity and ease of use, Scalability, High availability, High reliability, Data integrity, Security and Heterogeneity.

The types of DFS include Network File System (NFS), parallel file system, Access transparent DFS

The Network File System (NFS) is a client/server application that allows a computer user view and optionally store and update file on a remote computer as if they were on the user's own computer. The user's system requires an NFS client and the other computer needs the NFS server.

A parallel file system is a type of distributed file system that distributes file data across multiple servers and provides for concurrent access by multiple tasks of a parallel application.

In Access transparent DFS, the files could be present on a totally different set of servers which are physically distant apart and a single set of operations should be provided to access these remote as well as the local files. The examples illustrating this property are the File system in Network File System (NFS), SQL queries, and Navigation of the web.

Hadoop uses a distributed user level file system (HDFS) to manage storage resources across the cluster. To achieve the maximum portability HDFS is implemented using java at user level file system in heterogeneous environment. And also it provides global access to files in the cluster [7].

HDFS is implemented by two services i.e. Namenode and Datanode. The Namenode is centralized service in the cluster operating on a single node. It is also responsible for maintaining HDFS directory tree. Clients will contact the Name node in order to perform common file system operations such as open, close, rename and delete [5]. Each data node stores HDFS blocks on behalf of local or remote clients. In the node's local file system, each block is saved as a separate file. Namenode will make a request to a Datanode based on that, blocks are either created or destroyed on the data nodes. After making the request Namenode is responsible for validating and processing the request from clients.

V. HIGH PERFORMANCE COMPUTING CLUSTER

HPCC is an open source platform and it is an integrated set of systems software and other architectural components designed to provide data intensive computing capabilities from raw data processing to high performance query processing and data mining. HPCC is also known as DAS (Data Analytics Supercomputer) and its execution environment not limited to specific computing paradigm (programming model-machine independent).HPCC includes system configurations to support both parallel batch data processing and high performance online query applications using indexed data files. For parallel data processing ECL (Enterprise data Control Language) can be used.

HPCC system architecture includes two different cluster processing environment, each can be optimized independently.

A. Data Refinery Cluster:

The general purpose of the data refinery cluster is to process large volumes of data of any type. This stage is typically used for data cleansing and load processing of raw data. To support high performance structured queries and data warehouse application, the keyed data and indexes are created.

B. Roxie Cluster (Rapid Online XML Information Exchange):

This cluster functions as a rapid data delivery engine. High performance online processing can be achieved by using the distributed indexed file system in an optimized execution environment. Both data refinery and roxie clusters utilize ECL for implementing applications [2].

The Programmer productivity can be significantly improved for complex operations without the need for extensive user defined functions, when the HPCC uses ECL language with extensive built in capabilities for parallel data processing. This system typically able to continue operation with reduced number of nodes following a node failure with automatic and transparent recovery of incomplete processing; this enables the system towards high level of fault resilience.

VI. GRID COMPUTING

Grid computing is rapidly emerging as dominant paradigm for wide area distributed computing. The main goal of grid computing is to provide service oriented infrastructure which enables pervasive access and coordinated sharing of geographically distributed hardware, software and information resources. Giving access to a vast set of heterogeneous resources for the user in an easy, simple and transparent way is called "Metacomputing". If Metacomputing is done on Wide Area Network or geographically distributed area network, it is referred as Grid computing environment [8].

Grid computing has evolved from its roots in parallel and distributed computing to its current state. To efficiently program the parallel machine, a long listing of parallel programming languages and tools were developed, that include Linda, Concurrent Prolog, BSP, Occam, Fortran-D, Compositional C++, pc++, Nexus, lightweight threads and the parallel virtual machine [9].

Grid computing possesses some of the characteristics which include Heterogeneity, Scalability, Fault tolerance and Security. The components of the grid computing are classified into three levels: User level, Middleware level and Resource level.

A. User Level:

The application and high level interfaces are available at the user level. Middleware services can be accessed by the application and the user via interfaces and protocols which is implemented by the high level interfaces.

B. Middleware Level:

This layer provides the major functionalities of the grid and also services like resource discovery, resource scheduling and allocation, fault tolerance, security mechanisms and load balancing. This layer also provides a transparent view of the resources which are available to the user.

C. Resource Level:

This layer provides the local services that render computational resources like CPU cycle, storage, computers, network infrastructure, software etc.

Computational grids have designed to serve the different communities with varying characteristics and requirements. Due to this reason there is not a single uniform architecture for grids. But basic services that almost all the grids can provide though different grids use different approaches. The grid architecture organizes the components into layers such as Fabric layer, Connectivity layer, Resource layer, Collective layer and Application layer.

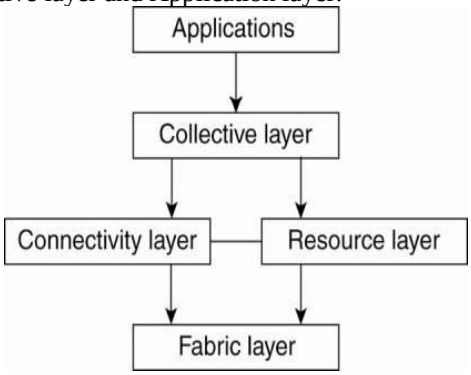

Fig. 1: Basic Grid Architecture

1) Fabric Layer:
This layer mainly provides the resources that contain computers [OS- windows NT/UNIX], storage devices and databases. Also resource management mechanisms provide quality of service.
2) Connectivity Layer:
This layer encompasses core communication and authentication protocols for transactions. These two protocols can be drawn from TCP/IP protocol stack. Communication protocols deals with data exchange between fabric layer and resource layer whereas authentication protocol provide secure cryptographic mechanisms for the user and resource identification.
3) Resource Layer:
This layer builds on the communication and authentication protocols of the connectivity layer to define API and SDK for sharing operations. The resource layer protocols are classified into Information protocols and Management protocols. Information protocol mainly used to obtain the necessary information regarding the structure and the state of the resources and the management protocol used to avoid the access to the shared resources.
4) Collective Layer:
This layer helps in co-coordinating the multiple resources whereas resource layer concentrate on interaction with single resources. This layer also supports the services such as directory, scheduling, monitoring and software discovery services.
5) Application Layer:
This layer contains user applications and programs.
The inherent scalability, heterogeneity, dynamism and non-determinism of grids and grid applications have resulted in complexities, making both the infrastructure and the applications brittle and insecure. This leads to number of recent research initiatives such as autonomic grids, cognitive grids and semantic grids.

Grid computing offered the advantage about storage capabilities and the processing power. Also makes the big contributions among the scientific research, help the scientists to analyze and store the large and complex data [6].

VII. DATA INTENSIVE COMPUTING SYSTEMS- COMPARATIVE STUDY

Characteristics	*Supercomputing*	*Hadoop*	*HPCC*	*Grid*
Platform	*Open source platform*	*Open source software project sponsored by apache software foundation.*	*Open source data intensive computing system platform developed by Lexis Nexis risk solutions.*	*Open source platform*
Working Environment	*Depends on application (homogeneous/heterogeneous)*	*Portability across heterogeneous hardware and software platforms*	*Heterogeneous Environment*	*Heterogeneous Environment*
Operating Systems	*Unix/Linux*	*Linux, FreeBSD, Mac OS/X, Solaris and Windows*	*Linux/Windows*	*Windows NT/Unix*
Programming Model	*Machine dependent*	*Simple machine dependent programming models*	*Machine independent*	*Machine dependent*
Programming Languages/Tools	*Fortran & C, Java, Python*	*Pig, HBase, Hive, Java*	*ECL*	*Linda, Concurrent Prolog, BSP, Occam, Fortran-D, Compositional C++, pc++, Nexus, lightweight threads and the parallel virtual machine.*
Data Management	*Utilizes the data from shared repositories and servers leads to more network traffic because of the high data movement.*	*Back-end data processing. Each data block is stored as a separate file in the local file system.*	*Data movement is faster-moving the code to the nodes where data is residing leads to less network traffic.*	*Sophisticated data sharing*
Load Balancing	*Centralized load balancing algorithm suffers from scalability problems on the machine with a small amount of memory*	*General purpose workloads are balanced by the existing I/O scheduler, which are designed for that purpose.*	*Query request across Roxie clusters is implemented using external load balancing communication devices*	*Grid Agents are responsible for load balancing.*
Scalable Mechanisms	*Non-scalable mechanisms, not suitable for interactive users*	*Hadoop scalability depends on the vertical scalability of the Name node server*	*Defined based on number of clusters ,nodes in the clusters & their type to meet system performance and user requirement*	*Depends on well-known and widely adopted services, domains and style of computations.*
Applications	*Public health, political science, economics and scientific computations.*	*Marketing analytics, image processing, machine learning, sophisticated data mining*	*Earthquake and hurricane predictions, protein dynamics, e-commerce.*	*Chemistry, neuro-sciences, nuclear engineering, scientific research, data mining and network simulation*

VIII. CONCLUSION

Data management in supercomputing has less support, to achieve maximum performance it requires tedious optimization. Hadoop offers a complete infrastructure to handle the big data. However, the issues such as lack of resource management, application deployment and multiple data sources poses a challenge to Hadoop's adoption, whereas HPCC provides high level of fault resilience and language capabilities, which reduce the need for reprocessing in case of system failures and also HPCC systems provide a better support for data management. Finally the Grid supports layered architecture which makes the big contributions among the scientific research, help the scientists to analyze and store the large and complex data.

REFERENCES

[1] Randal E. Brayant (May 2007), Data- Intensive Supercomputing: The case for DISC.
[2] Anthony M. Middleton, LexisNexis Risk Solutions (April 2011), HPCC Systems: Data Intensive Super Computing Solutions.
[3] R.A. Fadnavis, samrudhi tabhane.Big Data Processing Using Hadoop: vol 6(1), 2015.

[4] Sagiroglu S; Sinanc D, Big Data: A Review". May 2013
[5] Jeffrey Shafer, Scott Rixner, and Alan L. Cox. The Hadoop Distributed File System: Balancing Portability and Performance
[6] Garlasu D; Sandulescu V; Halcu E; Neculoice G (17-19) Jan 2013. "A Big data implementation based on Grid computing", Grid computing.
[7] HDFS (Hadoop distributed file system) architecture. http:// Hadoop.apache.org/common/docs/current/hdfs_design.html, 2009.
[8] Grid computing by Sunil Shankar.
[9] Manish Parashar, Grid Computing: Introduction and Overview.