International Association of Scientific Innovation and Research (IASIR)
(An Association Unifying the Sciences, Engineering, and Applied Research)

ISSN (Print): 2279-0063
ISSN (Online): 2279-0071

International Journal of Software and Web Sciences (IJSWS)
(Open Access, Double Blind Peer-reviewed, Refereed and Indexed Journal)
www.iasir.net

Analysis and Computational Solutions to Large Scale Data Management

Sanam Satya Krishna Teja[1], Satwik Kumar B Shiri[2]
[1,2]Computer Science Department
R. V. College of Engineering
Bengaluru, Karnataka, INDIA

Abstract: *The Big Data market continues to exhibit strong momentum as businesses accelerate their transformation into a data-driven companies. This momentum is driving strong growth in the big data-related infrastructure, software, and services. Success in the data management will depend on our ability to properly interpret the high-dimensional, large-scale data sets that are generated by these technologies, which in turn requires us to adopt advances in informatics. Collecting and analyzing large amounts of data, mainly unstructured data, is very difficult task and inefficient. This paper aims to address the large scale data management requirements of several forthcoming data intensive projects using HPC (High Performance Computing).*
Keywords*: HPC; Big Data; Data Management; Computational Solutions; MapReduce*

I. Introduction

A new forecast from the International Data Corporation (IDC) sees the big data technology and services market growing at a compound annual growth rate (CAGR) of 23.1% over 2014-2019 forecast period with annual spending reaching $48.6 billion in 2019. A new IDC Special Study examines spending on big data solutions in greater detail across 19 vertical industries and eight big data technologies.

Addressing the challenges of large scale data requires efficiently targeting limited resources — money, space, power and people — to solve an application of interest. This also requires understanding and exploiting the nature of data and the analysis algorithms. Factors that must be taken care to solve a particular problem most efficiently and effectively include: the size and complexity of the data, the ease with which data can be efficiently transferred over the internet; whether the algorithm to apply to the data can be efficiently parallelized; and whether the algorithm is simple or complex (for example, an algorithm applied to reconstructing Bayesian networks through the integration of diverse types of large-scale data).

One of the most important aspects to consider for computing large data is the parallelization of the analysis algorithms. Data-intensive and critical problems are primarily solved by distributing the tasks over many computer processors. Because different algorithms used to solve a problem are compatible to the different types of parallelization, different computational platforms are needed to achieve the best performance.

II. Challenges posed by large scale data analysis

Big Data are characterized by high dimensionality and large sample size. Although processing individual data is complex, the true challenge is in integrating the multiple sources of data. Analysis of such large high-dimensional data sets poses several hurdles for storage and analysis. Among the most important challenges are: data transfer and management, standardization of data formats and accurate modeling of results by integrating data from multiple dimensions.

A. Data transfer and management

Analysis results can increase the size of raw data, given that all relationships among data and other variables are stored and mined. Therefore, it is important to efficiently move these big data sets around the internet, to provide access control if the data are stored centrally and to properly organize large-scale data in ways that facilitate analyses. Network speeds are slow to enable large amount of data to be routinely transferred over the web. Currently, the most efficient mode of transferring large quantities of data is to copy the data to big storage drive and then ship the drive to the destination. This is inefficient and presents a barrier for the data exchange between groups.

B. Standardization of data formats

Different centre's generate the data in different formats and some analysis tools require the data to be in particular formats or require different types of the data to be linked together. Therefore, time is wasted in reformatting and re-integrating data multiple times during single analysis. For example, next-generation sequencing companies do not deliver raw sequencing data in a format common to all the platforms, as there is no industry-wide standard beyond simple text files that include the nucleotide sequence and corresponding quality values. As a result, carrying out sequence analyses across the different platforms requires tools to be adapted to specific platforms. One solution is to store the data sets centrally and bring high-performance computing (HPC) to the data. Although this is good solution, it has access control challenges such as groups generating the data may want to retain control

over who can access data before they are evaluated. Further, access to big data sets requires IT support, which is costly.

C. Modeling of results

A primary goal for researchers is to integrate diverse and large-scale data sets to construct models that can predict complex phenotypes such as disease. Constructing predictive models could be computationally demanding. For example if we consider reconstructing Bayesian networks using large-scale DNA variation, DNA – protein interaction, protein binding, metabolite and other types of the data. As the scales and diversity of data grow, this type of modeling will increasingly become important for representing complex systems. Computationally, the need for this type of modeling poses an intense problem that falls into the category of NP hard problems.

III. Analysis and Computational Solutions

The two main approaches to large scale data analysis are:

A. MapReduce

Several years ago, a distributed computing paradigm known as MapReduce emerged to provide better scalability and fault tolerance and to simplify development of massively parallel computing applications for computing procedures involving many (>100) simultaneous processes. MapReduce provides middleware and an application programming interface (API) for developers who want to analyze big data. More specifically, when developer defines processing that is compliant with two types of APIs, map and reduce functions, processing is distributed efficiently for execution on multiple servers (Fig. 1).

Mechanism of MapReduce ensures that complexity involved in developing a system that works on numerous servers can be concealed from developer. More specifically, this approach has the following advantages.

- Scheduling: The servers that execute processing defined by the map and reduce functions are selected automatically by middleware. The middleware selects nearby server in which a chunk of the data is stored to be the map function execution server. This reduces volume of data transfers and enables efficient processing.
- Synchronized processing: Data transfers between servers for map and reduce functions are synchronized.
- Fault tolerance: To ensure that processing can continue overall even when several servers have failed, the data backups and intermediate processing results are stored automatically. If failure actually occurs, servers restart processing using the intermediate processing results and the backups.

The synchronization and scheduling processing become more complex as the scale of data and servers increases; thus, server breakdowns often occur. Since such complexities are concealed from developer by MapReduce, he or she can focus on developing big data analysis methods.

B. Hadoop MapReduce

Hadoop MapReduce is software framework for easily writing applications which process vast amounts of data (multi-terabytes of data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a fault-tolerant and reliable manner.

MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in acompletely parallel manner. The framework sorts outputs of the maps, which are then input to reduce tasks. Typically both input and output of the job are stored in a file-system. The framework takes care of scheduling the tasks, monitoring them and re-executing the failed tasks.

Typically compute nodes and storage nodes are same, that is, the MapReduce framework and the Hadoop Distributed File System are running on same set of nodes. This configuration allows framework to effectively schedule tasks on nodes where data is already present, resulting in very high aggregate bandwidth across cluster.

The MapReduce framework consists of single master ResourceManager, one slave NodeManager per cluster-node, and a MRAppMaster per application.

Minimally, applications specify the input/output locations and supply the map and the reduce functions via implementations of appropriate abstract-classes and/or interfaces. These, and other job parameters, comprise job configuration.

The Hadoop job client then submits the job (jar/executable etc.) and configuration to ResourceManager which then assumes responsibility of distributing the software/configuration to slaves, scheduling tasks and monitoring them, providing status and diagnostic information to job-client. Although Hadoop framework is implemented in Java™, MapReduce applications need not be written in Java.

- Hadoop Streaming is a utility which allows users to create and run the jobs with any executables as the mapper and/or the reducer.
- Hadoop Pipes is SWIG-compatible C++ API to implement MapReduce applications (non JNI™ based).

C. Combining MapReduce and cloud computing.

HPC has been transformed in the past decade by maturation of cluster-based computing. With wide range of components available like different networks, storage systems and computational nodes, clusters can be optimized for many classes of computationally intense applications. There is significant cost associated with the building and the maintaining of cluster.

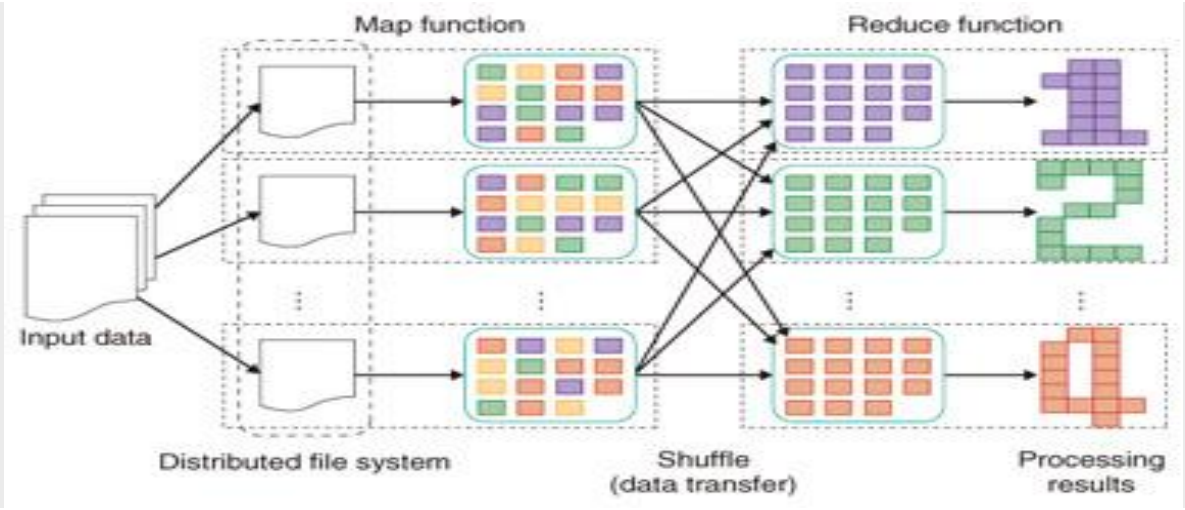

Figure 1: Flow of MapReduce processing

Even after the hardware components are acquired and assembled to build HPC cluster, substantial costs are associated with operating the cluster, including those of power, cooling, space, backup, fault recovery and IT support. In this sense a cluster contrasts with the grid computing, an earlier form of cluster-like computing. In this, combination of loosely coupled networked computers that are administrated independently work together on a common computational tasks at low or no cost.

Cloud Computing: Recently, supercomputing has been made more accessible and affordable through the development of the virtualization technology. The Virtualization software allows systems to behave like a true physical computer, but with flexible specification of details such as number of processors, disk size and memory, and operating system. Multiple virtual machines can be run from a single physical server, providing significant cost savings in server hardware, maintenance and administration. The use of these on-demand virtual computers is known as cloud computing.

Combination of the distributed MapReduce and the cloud computing can be effective answer for providing petabyte-scale computing to a wide set of practitioners. The MapReduce model is often a straightforward fit for data-parallel approaches in which single task, such as the read alignment, is split into smaller identical sub-tasks that operate on an independent subset of data. One of the biggest advantage of combining the MapReduce and the cloud computing is the reuse of a growing community of developers and software tools, and with just a few mouse clicks you can set up MapReduce cluster with many nodes. For example, Amazon EC2 cloud-computing environment provides a specific service for streamlining the set-up and running of Hadoop-based workflows. Distributed MapReduce should be viewed as an emerging successful model for petabyte-scale computing, but not necessarily only model to consider when mapping a particular problem to larger-scale computing.

D. Parallel DBMSs

The database systems capable of running on clusters of shared nothing nodes have existed since the late 1980s. All these systems support standard relational tables and SQL, and thus the fact that data is stored on multiple machines is transparent to the end-users. Many of these systems build on pioneering research from the Gamma and Grace parallel DBMS projects. The key aspects that enable parallel execution are most tables are partitioned over the nodes in a cluster and that the system uses an optimizer that translates SQL commands into query plan whose execution is divided amongst multiple nodes. Because the programmers only need to specify their goal in high level language, they are not burdened by underlying storage details, such as join strategies and indexing options. Consider a SQL command to filter the records in table T1 based on predicate, along with a join to a second table T2 with an aggregate computed on the result of join. A basic sketch of how this command is processed in parallel DBMS consists of three phases. Since the database will have already stored T1 on some collection of nodes partitioned on some attribute, filter sub-query is first performed in parallel at these sites similar to filtering performed in the Map function. Following this step, one of the two common parallel join algorithms are used based on the size of data tables. For example, if the number of records in T2 is small, then DBMS could replicate it on all the nodes when data is first loaded. This allows join to execute in parallel at all the nodes. Following this, each node then computes the aggregate using its portion of answer to join. A final "roll-up" step is required to compute the final answer from these partial aggregates. If the size of data in T2 is large, then T2's contents will be distributed across multiple nodes. If these tables are partitioned on different attributes than those used in join, the system will have to hash both T2 and the filtered version of T1 on join attribute using a common hash function. The redistribution of both the T2 and the filtered version of T1 to the nodes is similar to processing that occurs between Map and Reduce functions. Once each node has the necessary data, it then performs a hash join and

calculates preliminary aggregate function. Again, a roll-up computation must be performed as the last step to produce the final answer.

IV. Conclusion

Large scale data management is very much required in today's ever growing data usage. Addressing the challenges of large scale data requires money, space, power and people. Large scale data can be managed effectively and efficiently using the MapReduce technique combined with cloud computing. Using cloud computing we are able to increase the reliability of the data. The large scale data management is in research and we can expect lot, more to new techniques to come.

References

[1] Rui Kubo, Yoshifumi Fukumoto, and Makoto Onizuka. "Efficient Large-scale Data Analysis using MapReduce".

[2] Owens JD, et al. A survey of general-purpose computation on graphics hardware. Comput. Graph. Forum. 2007;26:80–113.

[3] Snir, M. *MPI-The Complete Reference* 2nd edn (MIT Press, Cambridge, Massachusetts, 1998).

[4] Liu Y, Maskell DL, Schmidt B. CUDASW + + : optimizing Smith–Waterman sequence database searches for CUDA-enabled graphics processing units. BMC Res. Notes. 2009;2:73.

[5] Linderman MD, et al. High-throughput Bayesian network learning using heterogeneous multicore computers; Proc. of the 24th ACM Int. Conf. on Supercomputing; Japan; 2–4 Jun 2010; Tsukuba, Ibaraki. New York: ACM; 2010. pp. 95–104.

[6] Nickolls J, Buck I, Garland M, Skadron K. Scalable parallel programming with CUDA. Queue.2008;6:40–53.740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] Armbrust M, et al. Above the Clouds: A Berkeley View of Cloud Computing. Berkeley: University of California; 2009.

[8] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi David J. DeWitt Samuel Madden Michael Stonebraker. "A Comparison of Approaches to Large-Scale Data Analysis".

[9] Matsunaga, A., Tsugawa, M. & Fortes, J. in *4th IEEE International Conference on eScience.* 222–229 (IEEE, Indianapolis, Indiana, 2008).

[10] Owens, J. D. *et al*. A survey of general-purpose computation on graphics hardware. *Comput. Graph. Forum* **26**, 80–113 (2007).

[11] Garey, M. R. & Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman)

[12] Dean, J. & Ghemawat, S. MapReduce: simplified data processing on large clusters. *6th Symp. on Operating System Design and Implementation*[online].