Big Data Frameworks for Radio Astronomy Data: A Summary

R. W. Bunney
email: 21298244@student.uwa.edu.au

March 2017

Abstract

The increase in Big Data problems faced by a variety of industries has led to the development and uptake of numerous tools designed to assist in the processing of large data sets. For scientists and research teams investigating problems that result in large data deluge, the ability to use an existing framework without having to be specialists in parallel and distributed computing has the advantage of being time and cost efficient. It is also useful for researchers to know the benefits and drawbacks of a framework before choosing one for their particular use case. This paper reviews literature associated with three popular, related big data tools - MapReduce, Hadoop, and Apache Spark - and discusses their respective approaches. Then, we examine the DALiuGE framework, which has been designed to fill gaps identified in the systems above in the context of radio astronomy research.

1 Introduction

As the complexity and scale of new science projects continues to increase in size, so does the magnitude of computational power required to work with the results. The concept of 'Big Data' processing is now increasingly relevant to scientists, as traditional high-performance computing methods can no longer cope with the size of and the real-time requirements of the data. The Square Kilometre Array (SKA) is one such project that will generate vast quantities of data with specific science requirements, and will have to manage the resultant data under time and cost pressures [4, 11]. In order to efficiently process this data, science teams need to form an understanding of existing approaches to how large-scale distributed systems are managing the ‘deluge’ of data [1], and to establish whether or not they are applicable to radio astronomy.

The aim of this paper is to provide a non-exhaustive review of systems that purport to address Big Data concerns, and evaluate the current ideas prevalent in the literature and their applicability for radio astronomy processing. As the fields of distributed systems, parallel computing, and data processing cover a large area of literature, the paper focuses on key iterations of Big Data tool-sets that are already popular and have a large uptake within the Big Data community. We can summarise the general requirements that a potential framework would have to address as follows:

1. Expertise in parallel or distributed computing is not required;
2. Astronomical algorithms are easy to implement
3. The framework can facilitate astronomical data-sets; and
4. It provides good quality debugging and monitoring.

This paper is organised as follows. Section 2 discusses current approaches to astronomical computing, and why there is a need for change given the scope of the SKA. Section 3 discusses the parallel processing paradigm MapReduce, distributed computing frameworks Hadoop and Spark, and provides a discussion on their use and drawbacks. Section 4 discusses the proposed DALiuGE framework, introduced as a solution to the limitations of the above frameworks in scientific computing. The current shortcomings of DALiuGE are discussed, and concluding remarks are presented in Section 5.

2 Current Approaches to Astronomical Data Processing

Traditional approaches to processing astronomy data involve the use of scheduling systems to run scripts that have been written by astronomers to run software applications necessary for a particular science case. These are statically defined workflow scripts that are executed on systems such as the SLURM Workload Manager (SLURM), and then executed in High Performance Computing (HPC) environments [17].

SLURM is a workload manager that allows users to define tasks to be run in a script, according to which the SLURM daemon will schedule and allocate resources for each task [13]. Designed initially for the Livermore Computing Centre, it is a well-known and popular tool for large-scale HPC environments.

However, for future data challenges such as those presented by the SKA, the approach to computing in a traditional SLURM environment is limited. An example of this is presented in Dodson et al. [10], in which data reduction on the CHILES pipeline was testing on a variety of computing environments. The science team to hard-coded scripts for each computing environment, a time-intensive task that will increases as science teams become more global. This demonstrates a limitation with the application-driven workflow that is the core of models such as SLURM; it becomes time-consuming and costly to have astronomers working in global teams tailoring and re-developing scripts in order to have them work in their environments.

Another limitation of the SLURM workload manager is the need for fine-tuning of scripts to ensure that optimal scheduling and processing is maintained [12]. Given the size of the data payloads and the speed at which the data must be processed, it is impractical for project scientists to maintain a working knowledge of software and hardware requirements for each task in order to plan optimally. [17].

It is clear that whilst these traditional HPC approaches are useful within traditional clusters and HPC settings, we can see that by reflecting on the requirements outlined in the introduction, the real-time demands of the SKA require a more comprehensive and versatile approach.

3 Data Parallel Frameworks

The increase in the use of technology in enterprise and the subsequent data ‘deluge’ [1] has resulted in the demand for technology that allows businesses to efficiently manage the constant inflow of large data sets. This demand has led to the development of big data paradigms such as MapReduce, distributed file systems for data storage like Hadoop, and cluster computing tools in the form of software like Apache Spark. These tools are all utilised by the ‘Big Data’ community and employed by companies such as Yahoo and Google to manage their industrial-scale data processes [3, 18].

This section discusses the various implementation details and approaches to data parallelism and processing in distributed systems that these frameworks utilise, and contrasts them with the requirements of large-scale science projects like the SKA.

3.1 Map Reduce

Introduced in Dean and Ghemawat [3], MapReduce is a model designed for the process and generation of large data sets. A developer can specify a

Map function that takes a key-value pair and generates a set of intermediate pairs. These pairs are grouped together by their intermediate key I, and then passed to a reduce function that will merge all intermediate values associated with the same key. An example application of this model presented in the original paper is the counting of URL access frequency; the map function will process web request logs and the output, and the reduce function adds up all the values for the same URL and returns a URL-TotalCount pair [3].

Results presented in this paper demonstrated how promising the extent to which the MapReduce model is applicable; large scale graph computations, clustering problems, and data extraction for queries all experienced an increase in performance. The paper also comments on the ability for those without experience in distributed or parallel systems to exploit the framework for their own use cases with ease - a significant benefit if one is to consider system being used by non-specialists.

Whilst this programming model definitely has its benefits to the applications listed above, as well as others , they represent problems that are more ‘embarrassingly parallel’ than those faced in astronomy. The core operations involved are also more commutative and associative than that of the astronomical counterparts; for example, astronomical algorithms such as CLEAN [5] produce undesirable result when using this type of approach [17]. A database search for all references of a given value can be partitioned, and the results returned, without the order of partitions a necessity. However, deconvolution algorithms like CLEAN work iteratively on an image as it aims to find the point source with the highest value; this means the order of each iteration has an impact on the final outcome of the return value, and the algorithm will return a different result. This demonstrates the difficulty astronomers would have in implementing existing, well-known astronomical methods into the framework. It is evident, then, that whilst MapReduce works effectively for the demands of an Internet company, the model lacks the flexibility required in radio astronomy operations.

3.2 Hadoop

Shvachko, Kuang et al. [15] introduce the Hadoop Distributed File System (HDFS) as a solution to storing and streaming large data sets in a reliable manner. The file system splits content into blocks of data (typically 128 megabytes), and each block is replicated on multiple nodes. The entire namespace of a HDFS instance is kept in memory, and Hadoop is designed to store data in this way across thousands of servers through a master/slave architectural approach [9]. When using the Hadoop system, data is automatically split up into data partitions that are then managed by the cluster and HDFS. Initially developed for Yahoo to cope with scaling clusters with the MapReduce framework [15], it has been applied to a variety of other applications; for example, Patel et al. [9] have used the cluster to successfully increase the performance of software analysing geological data.

Given the ease of processing that Hadoop offers, and the increase in performance with in-memory read and write access, the Hadoop ecosystem appears to be a good candidate for processing astronomical data sets. However, splitting up astronomy data into 128 megabyte partitions is a non-trivial process; in Dodson et al. [10], the process required to slice data sets in the frequency domain involves using Fast Fourier Transforms (FFTs), which are computationally non-trivial tasks. In order to place this data into the Hadoop framework, one would have to pre-partition the data so that it would fit to the requirements of the framework. This becomes a time expensive process that greatly reduces the benefits of having system which identifies its partitioning process as a competitive feature.

3.3 Apache Spark

In an attempt to build upon the MapReduce paradigm and the HDFS, Zahria et al. [18] discuss the development of the Spark framework, designed to take advantage of the above technologies and leverage them for use on data sets that processed on multiple environments in parallel. Examples of these applications include machine learning algorithms and interactive data analysis tools. Spark uses an abstraction called Resilient Distributed Data sets (RDDs), which are read-only object collections designed to be partitioned across a cluster. This process allows object sets to be rebuilt if one partition is lost [18]. The Spark ecosystem allows for a generalised approach to big data computation, in which task can be scheduled; data shared; and on which batch jobs, SQL analytics, and other processes can be run. Developers can interface with Spark across a variety of APIs in languages such as Java, Scala, and Python, and require no previous experience with parallel software development [1].

Sehrish et al. [14] provide an evaluation of Spark in the context of High Energy Physics (HEP) research. HEP data sets are similar to that of radio astronomy data sets in that they involve processes such as pattern matching, parameter estimation, and classification. The results of the paper demonstrate that whilst Spark provides ease of scaling without any developer expertise in distributed systems, the result is that an untuned MPI implementation of an event classification algorithm performed 10 times faster than the respective Spark implementation. The researchers provide a number of potential reasons for this result. Firstly, the frameworks 'black-box' approach to system optimisation means that data distribution and task assignment is abstracted from a user through a First-in-First-Out (FIFO) scheduler [16]; this approach does not guarantee an optimal schedule. Secondly, the researchers found that the use of wrapped types lead to a lot of boxing and unboxing of values; testing showed that there was a 28% increase in performance if using a primitive type.

Also noted was the lack of a high performance linear algebra library available in the Spark ecosystem may have contributed to the performance decrease. Whilst the first two issues are definitely potential drawbacks for the system, it is unfair to judge the framework based on the lack of a particular library. Given the ability for developers to write plug-ins for the Spark ecosystem, this particular issue is less of an indictment on the framework itself, and more of a call for someone to develop a particular application within it.

3.4 Concluding Remarks

The current literature demonstrates that there are obvious benefits to the above frameworks - particularly in the way they allow non-specialists to create sophisticated and scalable solutions to a variety of problems. However, it is clear that they lack the flexibility required for processing radio astronomy data. In order to fill this gap, a new tool to address the requirements of SKA-scale data processing has been developed, influenced by past research conducted on SKA-precursor data [10].

4 DALiuGE

In answer to the above drawbacks and failings of the discussed models and frameworks, Wu et al. [17] propose a solution using a data-driven scheduling model [8], that allows scientists to plan a logical workflow of applications for a given science project. The system translates this workflow into a Directed Acyclic Graph (DAG) that takes into account task dependencies [2, 7]. Finally, a schedule is generated, which aims to minimise the execution time for a given set of resources.

The paper discusses success with applying the framework in two science use cases, both radio astronomy related. Whilst this is obviously a good thing for distributed computing in radio astronomy, there is no discussion on how well it extends beyond the field. Further experimentation with data sets and applications from another would strengthen the framework's appeal to other scientists.

There are limitations to the DALiuGE framework presented in Wu et al [17]. Current approaches to the scheduling assume all resources are homogeneous; this means that the benefits of systems that are heterogeneous in nature, such as cloud computing environments, will receive less-than-optimal schedules. This is a waste of computational – and potentially financial – resources.

Another potential issue is that data scheduling occurs before the execution of any tasks [6]. This means that too many node failures across a particular schedule will lead to a halt in processing, and a new schedule would have to be generated.

Table 1: Requirements of SKA-scale Computing Frameworks

	SLURM	MapReduce	Hadoop	Spark	DALiuGE
No Expertise	X	X	X	X	X
Astronomical algorithms	X			X	X
Astronomical data-sets	X				X
Debugging and monitoring			X	X	X

5 Conclusion

By looking at key papers describing three of the most popular Big Data tools, we have been able to demonstrate the potential pitfalls associated in their use for specialist, scientific data processing. For example, the naive data partitioning approach used by Hadoop is not feasible for complex data sets used in Radio astronomy; nor is the commutative nature of the MapReduce model applicable to non-commutative data sets as found in radio astronomy. Table 1 provides a complete overview of how these frameworks fulfil the requirements outlined in the introduction. The DALiuGE framework [17] sets out to address the limitation of some of these frameworks; however, given its relative infancy, the success is currently only limited to radio astronomy. This makes it a less generic option than the above frameworks, and the inflexibility of its scheduling process means it may suffer in production environments with high node failure.

References

[1] Armbrust, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., Stoica, I., Wendell, P., Xin, R., and Zaharia, M. Scaling Spark in the Real World: Performance and Usability. *Proceedings of the VLDB Endowment 8*, 12 (2015), 1840–1843.

[2] Chaudhary, V., and Aggarwal, J. K. A generalized scheme for mapping parallel algorithms. *IEEE Transactions on Parallel and Distributed Systems 4*, 3 (Mar 1993), 328–346.

[3] Dean, J., and Ghemawat, S. Mapreduce: Simplified Data Processing on Large Clusters. *Commun. ACM 51*, 1 (Jan. 2008), 107–113.

[4] Dewdney, P., Turner, W., Braun, R., Santander-Vela, J., Waterson, M., and Tan, G.-H. Ska1 system baselinev2 description. *Document number SKA-TEL-SKO-0000308 1*, 1 (2015).

[5] Högbom, J. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series 15* (1974), 417.

[6] ICRAR. DALiuGE: Using the Logical Graph Editor, 2016.

[7] Kwok, Y.-K., and Ahmad, I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Comput. Surv. 31*, 4 (Dec. 1999), 406–471.

[8] Nikhil, R., et al. Executing a program on the mit tagged-token dataflow architecture. *IEEE Transactions on computers 39*, 3 (1990), 300–318.

[9] Patel, A. B., Birla, M., and Nair, U. Addressing Big Data Problem Using Hadoop and Map Reduce. In *2012 Nirma University International Conference on Engineering (NUiCONE)* (Dec 2012), pp. 1–5.

[10] R. Dodson and K. Vinsen and C. Wu and A. Popping and M. Meyer and A. Wicenec and P. Quinn and J. van Gorkom and E. Momjian. Imaging SKA-scale data in three different computing environments. *Astronomy and Computing 14* (2016), 8 – 22.

[11] Ratcliffe, S., Graser, F., Carlson, B., and B, O. SKA1 LOW SDP - CSP Interface Control Document. *SKA Project Document number 100-000000-002 1*, 1 (March 2016).

[12] SchedMD. Slurm: High Throughput Computing Administration Guide. `https://slurm.schedmd.com/high_throughput.html`. Accessed: 2017.

[13] SchedMD. Slurm Quick Start User Guide. `https://slurm.schedmd.com/quickstart.html`. Accessed: 2017.

[14] Sehrish, S., Kowalkowski, J., and Paterno, M. Exploring the Performance of Spark for a Scientific Use Case. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International* (2016), IEEE, pp. 1653–1659.

[15] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. The Hadoop Distributed File System. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on* (2010), IEEE, pp. 1–10.

[16] Spark. Job scheduling. `https://spark.apache.org/docs/1.2.0/job-scheduling.html`. Accessed: 2017.

[17] Wu, C., Tobar, R., Vinsen, K., Wicenec, A., Pallot, D., Lao, B., Wang, R., An, T., Boulton, M., Cooper, I., Dodson, R., Dolensky, M., Mei, Y., and Wang, F. DALiuGE: A Graph Execution Framework for Harnessing the Astronomical Data Deluge. *ArXiv e-prints* (Feb. 2017).

[18] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2010), HotCloud'10, USENIX Association, pp. 10–10.