Review of Optimistic Data Computation Approaches To Big Data In Cloud Environment

Roslin Dayana K[1], Dr.Vigilson Prem M[2]
[1, 2] Department of Computer Science and Engineering
[1, 2] R.M.D Engineering College, Gummidipoondi Taluk,Thiruvallur Dist.

Abstract*- Cloud computing is the widely used technology to provide different type of services to users in a pay as you go basis. Data which are stored on Cloud are generally huge and can be called Big Data. Big data is a broad term that deals with handling of very, very large amount of data which is not possible with traditional systems. This includes collection, storage, search, analysis, visualization of the data. These data can be accessed in a large amount and this leads to the finding of better approach for the quick computation of data. In this paper we have discussed about the various approaches that are existing to handle Big Data which can be used for optimistic data computation in Cloud Environments. We also have discussed about the problems that are yet to be solved in Big Data Computation in heterogeneous environment.*

Keywords*-* Big Data, Cloud Computing, Heterogeneous Environment, Optimistic Data Computation.

I. INTRODUCTION

Nowadays, huge and various type of data called Big Data are stored in cloud environment for the benefit of various end-users like virtualized desktop users, non technical end users, cloud choreographers and cloud service providers. These data need to be computed with optimal job performance and can be quickly accessed when a request arrives. This will help the end user to get the required data with minimum access time and effort. Recently many researches are going on, in improving the data computation performance. Some of the existing approaches to data intensive computation are Hadoop Map Reduce Version 2 and Resource-aware Adaptive Scheduling (RAS) and Phase and Resource Information-aware Scheduler for MapReduce clusters. In this paper we can explore the recent approaches to data intensive computation.

II. LITERATURE REVIEW

In this paper, we have discussed the various data intensive computation methods and their benefits. Among them, PRISM approach is the more efficient one in a homogeneous environment. Following are the existing approaches of optimistic data computation.

1. PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce [1]

The data-driven decision making has fueled the development of MapReduce, a parallel programming model that has become synonymous with large scale, data-intensive computation. In MapReduce, a job is a collection of Map and Reduce tasks that can be scheduled concurrently on multiple machines, resulting in significant reduction in job running time. Many large companies, such as Google, Facebook, and Yahoo!, routinely use MapReduce to process large volumes of data on a daily basis. Consequently, the performance and efficiency of MapReduce frameworks have become critical to the success of today's Internet companies.

A central component to a MapReduce system is its job scheduler. Its role is to create a schedule of Map and Reduce tasks, spanning one or more jobs, that minimizes job completion time and maximizes resource utilization. A schedule with too many concurrently running tasks on a single machine will result in heavy resource contention and long job completion time. Conversely, a schedule with too few concurrently running tasks on a single machine will cause the machine to have poor resource utilization.

Phase and Resource Information-aware Scheduler for MapReduce clusters (PRISM) is a scheduler that performs resource-aware scheduling at the level of task phases. Specifically, we show that for most MapReduce applications, the run-time task resource consumption can vary significantly from phase to phase. Therefore, by considering the resource demand at the phase level, it is possible for the scheduler to achieve higher degrees of parallelism while avoiding resource contention. To this end, a phase-level scheduling algorithm with the aim of achieving high job performance and resource utilization is developed.

Through experiments using a real MapReduce cluster running a wide-range of workloads, PRISM delivers up to 18 percent improvement in resource utilization while allowing jobs to complete up to 1:3 faster than current Hadoop schedulers.

An overview of the PRISM architecture is shown in Fig. 1.1. PRISM consists of three main components: a phase-based scheduler at the master node, local node managers that coordinate phase transitions with the scheduler, and a job progress monitor to capture phase-level progress information.

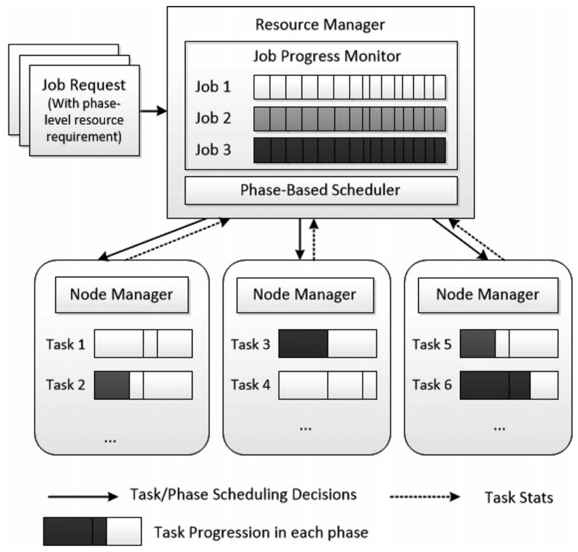

Figure 1. System Architecture

The phase-level scheduling mechanism used by PRISM is illustrated by Fig. 1.2. Similar to the current Hadoop implementation, each node manager periodically sends a heartbeat message to the scheduler. When a task needs to be scheduled, the scheduler replies to the heartbeat message with a task scheduling request (Step 1). The node manager then launches the task (Step 2). Each time a task finishes executing a particular phase (e.g. shuffle phase of the reduce task), the task asks the node manager for a permission to start the next phase (e.g. reduce phase of the task) (Step 3). The local node manager then forwards the permission request to the scheduler through the regular heartbeat message (Step 4). Given a job's phase-level resource requirements and its current progress information, the scheduler decides whether to start a new task, or allow a paused task to begin its next phase (e.g., the reduce phase), and then informs the node manager about the scheduling decision (Step 5). Finally, once the task is allowed to execute the next phase, the node manager grants the permission to the task process (Step 6). Once the task is finished, the task status is received by the node manager (Step 7) and then forwarded to the scheduler (Step 8). To perform phase-level scheduling, PRISM requires phase-level resource information for each job.

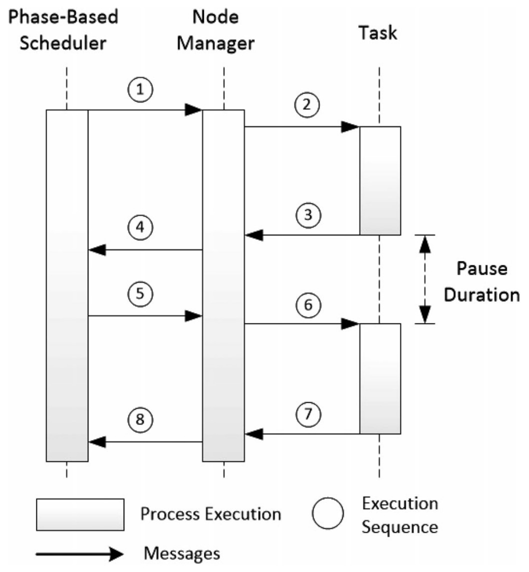

Figure 2. Phase Level Scheduling Mechanism

2. On Cloud computational models and the heterogeneity challenge [2]

Cloud computing is by far the most cost-effective technology for hosting Internet-scale services and applications. The MapReduce model, in particular, is largely used nowadays in cloud infrastructures to meet the demand of large-scale data and computation intensive applications. Despite its success, the implications of MapReduce on the management of cloud workload and cluster resources are still largely unstudied. In this paper, the authors show that dealing with the heterogeneity of workloads and machine capabilities is a key challenge. In today's cloud environment, workloads can have varied sizes, lengths, resource requirements, and arrival rates. The machines also have varied CPU, memory, I/O speed, and network bandwidth capacities. Jointly they pose difficult challenges pertaining, among others, to job scheduling, task and data placement, resource sharing and resource allocation. We analyze the heterogeneity challenge in these specific problem domains and survey the representative state-of-the-art works that try to address them.

3. Themis MR: An I/O Efficient MapReduce [3]

"Big Data" computing increasingly utilizes the MapReduce programming model for scalable processing of large data collections. Many MapReduce jobs are I/O-bound, and so minimizing the number of I/O operations is critical to improving their performance. In this work, we present ThemisMR, a MapReduce implementation that reads and writes data records to disk exactly twice, which is the minimum amount possible for data sets that cannot fit in memory. In order to minimize I/O, ThemisMR makes fundamentally different design decisions from previous

MapReduce implementations. ThemisMR performs a wide variety of MapReduce jobs – including click log analysis, DNA read sequence alignment, and PageRank – at nearly the speed of TritonSort's record-setting sort performance.

4. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types [4]

The authors consider the problem of fair resource allocation in a system containing different resource types, where each user may have different demands for each resource. To address this problem, they proposed Dominant Resource Fairness (DRF), a generalization of max-min fairness to multiple resource types. The DRF, unlike other possible policies, satisfies several highly desirable properties. First, DRF incentivizes users to share resources, by ensuring that no user is better off if resources are equally partitioned among them. Second, DRF is strategy-proof, as a user cannot increase her allocation by lying about her requirements. Third, DRF is envy free, as no user would want to trade his/her allocation with that of another user. Finally, DRF allocations are Pareto efficient, as it is not possible to improve the allocation of a user without decreasing the allocation of another user. The authors have implemented DRF in the Mesos cluster resource manager, and show that it leads to better throughput and fairness than the slot-based fair sharing schemes in current cluster schedulers.

5. Starfish: A Self-tuning System for Big Data Analytics [5]

Timely and cost-effective analytics over "Big Data" is now a key ingredient for success in many businesses, scientific and engineering disciplines, and government endeavors. The Hadoop software stack—which consists of an extensible MapReduce execution engine, pluggable distributed storage engines, and a range of procedural to declarative interfaces—is a popular choice for big data analytics. Most practitioners of big data analytics—like computational scientists, systems researchers, and business analysts—lack the expertise to tune the system to get good performance. Unfortunately, Hadoop's performance out of the box leaves much to be desired, leading to suboptimal use of resources, time, and money (in pay as-you-go clouds). The authors introduced Starfish, a self-tuning system for big data analytics. Starfish builds on Hadoop while adapting to user needs and system workloads to provide good performance automatically, without any need for users to understand and manipulate the many tuning knobs in Hadoop. While Starfish's system architecture is guided by work on self-tuning database systems, the authors discussed how new analysis practices over big data pose new challenges; leading them to different design choices in Starfish.

6. Resource-aware Adaptive Scheduling for MapReduce Clusters [6]

The authors presented a resource-aware scheduling technique for MapReduce multi-job workloads that aims at improving resource utilization across machines while observing completion time goals. Some of the other MapReduce schedulers define a static number of slots to represent the capacity of a cluster, creating a fixed number of execution slots per machine. This abstraction works for homogeneous workloads, but fails to capture the different resource requirements of individual jobs in multi-user environments. The authors proposed technique leverages job profiling information to dynamically adjust the number of slots on each machine, as well as workload placement across them, to maximize the resource utilization of the cluster. In addition, their technique is guided by user-provided completion time goals for each job.

7. Resource provisioning Framework for MapReduce Jobs with Performance [7]

Many companies are increasingly using MapReduce for efficient large scale data processing such as personalized advertising, spam detection, and different data mining tasks. Cloud computing offers an attractive option for businesses to rent a suitable size Hadoop cluster, consume resources as a service, and pay only for resources that were utilized. One of the open questions in such environments is the amount of resources that a user should lease from the service provider. Often, a user targets specific performance goals and the application needs to complete data processing by a certain time deadline. However, currently, the task of estimating required resources to meet application performance goals is solely the user's responsibility. In this work, the authors introduced a novel framework and technique to address this problem and to offer a new resource sizing and provisioning service in MapReduce environments. For a MapReduce job that needs to be completed within a certain time, the job profile is built from the job past executions or by executing the application on a smaller data set using an automated profiling tool. Then, by applying scaling rules combined with a fast and efficient capacity planning model, they generated a set of resource provisioning options. In addition, the authors designed a model for estimating the impact of node failures on a job completion time to evaluate worst case scenarios. The authors validated the accuracy of their models using a set of realistic applications. The predicted completion times of generated resource provisioning options are within 10% of the measured times in the 66-node Hadoop cluster.

8. Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling [8]

As organizations start to use data-intensive cluster computing systems like Hadoop and Dryad for more applications, there is a growing need to share clusters between users. However, there is a conflict between fairness in scheduling and data locality (placing tasks on nodes that contain their input data). To address the conflict between locality and fairness, the authors proposed a simple algorithm called delay scheduling: when the job that should be scheduled next according to fairness cannot launch a local task, it waits for a small amount of time, letting other jobs launch tasks instead. They find that delay scheduling achieves nearly optimal data locality in a variety of workloads and can increase throughput by up to 2x while preserving fairness. In addition, the simplicity of delay scheduling makes it applicable under a wide variety of scheduling policies beyond fair sharing.

9. Quincy: Fair Scheduling for Distributed Computing Clusters [9]

This paper addresses the problem of scheduling concurrent jobs on clusters where application data is stored on the computing nodes. This setting, in which scheduling computations close to their data is crucial for performance, is increasingly common and arises in systems such as MapReduce, Hadoop, and Dryad as well as many grid-computing environments. We argue that data intensive computation benefits from a fine-grain resource sharing model that differs from the coarser semi-static resource allocations implemented by most existing cluster computing architectures. The problem of scheduling with locality and fairness constraints has not previously been extensively studied under this model of resource sharing. They introduced a powerful and flexible new framework for scheduling concurrent distributed jobs with fine-grain resource sharing. The scheduling problem is mapped to a graph data structure, where edge weights and capacities encode the competing demands of data locality, fairness, and starvation-freedom, and a standard solver computes the optimal online schedule according to a global cost model. The authors evaluated their implementation of this framework, which they call Quincy, on a cluster of a few hundred computers using a varied workload of data- and CPU-intensive jobs. They evaluated Quincy against an existing queue-based algorithm and implement several policies for each scheduler, with and without fairness constraints. Quincy gets better fairness when fairness is requested, while substantially improving data locality. The volume of data transferred across the cluster is reduced by up to a factor of 3.9 in their experiments, leading to a throughput increase of up to 40%.

10. DryadLINQ: A System for General-Purpose Distributed Data-Parallel [10]

DryadLINQ is a system and a set of language extensions that enable a new programming model for large scale distributed computing. It generalizes previous execution environments such as SQL, MapReduce, and Dryad in two ways: by adopting an expressive data model of strongly typed .NET objects; and by supporting general-purpose imperative and declarative operations on datasets within a traditional high-level programming language. A DryadLINQ program is a sequential program composed of LINQ expressions performing arbitrary side effect-free transformations on datasets, and can be written and debugged using standard .NET development tools. The DryadLINQ system automatically and transparently translates the data-parallel portions of the program into a distributed execution plan which is passed to the Dryad execution platform. Dryad, which has been in continuous operation for several years on production clusters made up of thousands of computers, ensures ef- ficient, reliable execution of this plan. We describe the implementation of the DryadLINQ compiler and runtime. We evaluate DryadLINQ on a varied set of programs drawn from domains such as web-graph analysis, large-scale log mining, and machine learning. We show that excellent absolute performance can be attained—a general-purpose sort of 1012 Bytes of data executes in 319 seconds on a 240-computer, 960- disk cluster—as well as demonstrating near-linear scaling of execution time on representative applications as we vary the number of computers used for a job.

11. Improving MapReduce Performance in Heterogeneous Environments [11]

MapReduce is emerging as an important programming model for large-scale data-parallel applications such as web indexing, data mining, and scientific simulation. Hadoop is an open-source implementation of MapReduce enjoying wide adoption and is often used for short jobs where low response time is critical. Hadoop's performance is closely tied to its task scheduler, which implicitly assumes that cluster nodes are homogeneous and tasks make progress linearly, and uses these assumptions to decide when to speculatively re-execute tasks that appear to be stragglers. In practice, the homogeneity assumptions do not always hold. An especially compelling setting where this occurs is a virtualized data center, such as Amazon's Elastic Compute Cloud (EC2). We show that Hadoop's scheduler can cause severe performance degradation in heterogeneous environments. We design a new scheduling algorithm, Longest Approximate Time to End (LATE), that is highly robust to heterogeneity. LATE can improve Hadoop response times by a factor of 2 in clusters of 200 virtual machines on EC2.

12. MapReduce: Simplified Data Processing on Large Clusters [12]

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

In processing large quantities of data, a fundamental problem is to obtain a summary which supports approximate query answering. Random sampling yields flexible summaries which naturally support subset-sum queries with unbiased estimators and well understood confidence bounds. Classic sample-based summaries, however, are designed for arbitrary subset queries and are oblivious to the structure in the set of keys. The particular structure, such as hierarchy, order, or product space (multi-dimensional), makes range queries much more relevant for most analysis of the data. Dedicated summarization algorithms for range-sum queries have also been extensively studied. They can outperform existing sampling schemes in terms of accuracy on range queries per summary size. Their accuracy, however, rapidly degrades when, as is often the case, the query spans multiple ranges. They are also less flexible—being targeted for range sum queries alone—and are often quite costly to build and use. In this paper the authors proposed and evaluated variance optimal sampling schemes that are structure-aware. These summaries improve over the accuracy of existing structure-oblivious sampling schemes on range queries while retaining the benefits of sample-based summaries: flexible summaries, with high accuracy on both range queries and arbitrary subset queries.

13. Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture [14]

The authors provided a case study illustrating the challenges of real-time data processing in the era of "big data". They tell the story of how their system was built twice: first implementation was built on a typical Hadoop-based analytics stack, but was later replaced because it did not meet the latency requirements necessary to generate meaningful real-time results. The second implementation, which is the system deployed in production, is a custom in-memory processing engine specifically designed for the task. They conclude that the current typical usage of Hadoop as a "big data" platform, while great for experimentation, is not well suited to low latency processing, and points the way to future work on data analytics platforms that can handle "big" as well as "fast" data. In Twitter's Scribe infrastructure, Scribe daemons on production hosts send log messages to Scribe aggregators, which deposit aggregated log data onto per-datacenter staging Hadoop clusters. Periodic processes then copy data from these staging clusters into main Hadoop data warehouse.

14. A High-Performance Retrieval Method of Mass Data Oriented to Cloud Computing [15]

Data storage and processing technology has achieved rapid development, and the cloud computing is also arisen. Cloud computing is a distributed data calculation model with the property of high efficient, reliable, flexible and inexpensive. Currently, a various platform based on cloud computing is applied by many big enterprises, and the data size of these platforms is no longer MB and GB, but TB or PB. Thus, the mass data retrieve is the core of the cloud computing platform. Hadoop is the distributed file system that is suitable for large-scale data storage and access, and has the features of extensible, economy, reliability and efficiency, which is the preferred platform to store and retrieve mass data efficiently. So, the mass data storage and retrieval system based on Hadoop is built. In order to develop the applicable search engine, Hbase is selected as the distributed and unstructured database, and has a good performance.

III. CONCLUSION

In this paper, we have discussed about the various approaches developed by different authors for the intensive data computation. In this regard, efficient optimistic data computation has been a recent issue that if achieved will have the benefit of retrieving any computed data from a huge collection of data called Big Data.

IV. FUTURE ENHANCEMENT

There are many interesting paths to be achieved as future exploration. For example, scheduling problem for machines with heterogeneous performance characteristics in a cloud environment and improving the scalability of PRISM, an efficient scheduling method on huge complex data using distributed schedulers.

REFERENCES

[1] Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutaba and Bernard Wong, "PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce", IEEE Transactions on Cloud Computing, vol 3, no2, April/May 2015

[2] Raouf Boutaba, Lu Cheng and Qi Zhang, "On Cloud Computational Models and the Heterogeneity Challenge", J. Internet Serv. Appln., vol. 3, no.1, pp. 1–10, 2012.

[3] A. Rasmussen, M. Conley, R. Kapoor, V. T. Lam, G. Porter and A. Vahdat, "Themis MR: An I/O-Efficient MapReduce," in Proc. ACM Sump. Cloud Comput., 2012, p. 13.

[4] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker and I. Stoica, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types", in Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2011, pp. 323-336.

[5] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin and S. Babu, "Starfish: A Self-tuning System for Big Data Analytics" in Proc. Conf. Innovative Data Syst. Res., 2011, pp. 261-272.

[6] J. Polo, C. Castillo, D. Carrera, Y.Becerra, I. Whalley, M. Steinder, J. Torres and E. Ayguade, "Resource-aware Adaptive Scheduling for MapReduce Clusters", in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 187–207.

[7] A. Verma, L. Cherkasova, R. Campbell, "Resource Provisioning Framework for MapReduce Jobs with Performance Goals", in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 201, pp. 165-186.

[8] M. Zaharia, T. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenkar and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling", in Proc. Eur. Conf. Comp. Syst., 2010, pp. 265-278.

[9] M. Isard, V. Prabhakaran, J. Currey, U. Wieder and K. Talwar, "Quincy: Fair Scheduling for Distributed Computing Clusters", in Proc. ACM SIGOPS Symp. Oper. Syst. Principles, 2009, pp. 261–276.

[10] Y. Yu, M. Isard, D. Fetterly, M.Budiu, U. Erlingsson, P. Gunda and J. Currey, "DryadLINQ: A System for General-Purpose Distributed Data-Parallel", in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, pp. 1-14.

[11] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving Map Reduce Performance in Heterogeneous Environments", in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, vol.8, pp. 29-42.

[12] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Commun. ACM, vol.51, no.1, 2008, pp. 107-113.

[13] G. Mishne, J. Dalton, Z. Li, A. Sharma, and J. Lin, "Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture", in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2013, pp. 1147–1158.

[14] Jin Tao, "A High Performance Retrieval Method of Mass Data Oriented to Cloud Computing", International Conference on Robots and Intelligent System (ICRIS), Year: 2016, Pages: 16 - 21, DOI: 10.1109/ICRIS.2016.3.