ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 2, February 2014

Deadline Based Map-reduce Workload Management in Multijob

V.Sathya[1], K.Chandramohan[2]

PG Scholar-M.E, CSE, Gnanamani College of Engineering, Namakkal, T.N, India[1]

Head of Department, CSE, Gnanamani College of Engineering, Namakkal, T.N, India[2]

Abstract: A scheduling algorithm and technique for managing multi-job Map Reduce workloads that relies on the ability to dynamically build performance models of the executing workloads, and uses these models to provide dynamic performance management using Adaptive scheduler. One of the design goals of the Map-Reduce framework is mainly based Adaptive scheduler to maximize data locality across working sets, in an attempt to reduce network bottlenecks and increase overall system throughput. Data locality is achieved when data is stored and processed on the same physical nodes. Sometime the server based executing workloads are not delivered to those particulars. Because, the multi-job network areas occurred some problem. So, the server storage is too high. In this paper, overcome this problem by the use of another server that is related to the main server. The problem of main server workload data executing to related server. Finally, the unreachable storage data delivered from related server to the particular receiver. So, every time free storage space and speed process in this server and also improve the server response time.

Key Words: Map Reduce, performance management, task scheduling

I. INTRODUCTION

Cloud computing has dramatically transformed the way many critical services are delivered to customers for example, the Software, Platform, and Infrastructure as a Service paradigms, and at the same time has posed new challenges to data centers. The result is a complete new generation of large scale infrastructures, bringing an unprecedented level of workload and server consolidation, that demand new programming models, management techniques and hardware platforms. At the same time, it offers extraordinary capacities to the mainstream market, thus providing opportunities to build new services that require large scale computing. Therefore, data analytics is one of the more prominent fields that can benefit from next generation data center computing. The intersection between cloud computing and next generation data analytics services points towards a future in which massive amounts of data are available, and users will be able to process this data to create high value services. Consequently, building new models to develop such applications, and mechanisms to manage them, are open challenges. An example of a programming model especially well-suited for large-scale data analytics is MapReduce, introduced by Google in 2004.

Map Reduce workloads usually involve a very large number of small computations executing in parallel. High levels of computation partitioning, and relatively small individual tasks, are a design point of Map Reduce platforms [1]. While it was originally used primarily for batch data processing, its use has been extended to shared, multi-user environments in which submitted jobs may have completely different priorities [6]. This change makes scheduling even more relevant. Task selection and slave node assignment govern a job’s opportunity to progress, and thus influence job performance.

One of the design goals of the Map Reduce framework is to maximize data locality across working sets [3], in an attempt to reduce network bottlenecks and increase overall system throughput. Data locality is achieved when data is stored and processed on the same physical nodes [7]. Failure to exploit locality is one of the well-known shortcomings of most multi-job MapReduce schedulers, since placing tasks from different jobs on the same nodes will have a negative effect on data locality[4][5].

At the same time, there is a trend towards the adoption of heterogeneous hardware and hybrid systems in the computing industry. Heterogeneous hardware will be leveraged to improve both performance and energy consumption, exploiting the best features of each platform. For example, a Map reduce framework enabled to run on hybrid systems has the potential to have considerable impact on the future of many fields, including financial analysis, healthcare, and smart cities-style data management. MapReduce provides an easy and convenient way to develop massively distributed data analytics services that exploit all the computing power of these large-scale facilities. Huge clusters of hybrid many-core servers will bring workload consolidation strategies one step closer in future data centers.

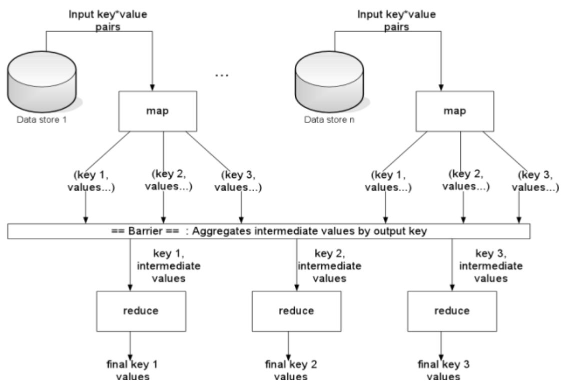

Fig1: A Map Reduce computation

Fig1: A Map Reduce computation

Scheduling algorithm and technique for managing multi-job Map reduce workloads that relies on the ability to dynamically build performance models of the executing workloads, and uses these models to provide dynamic performance management [2]. At the same time, it observes the particulars of the execution environment of modern data analytics applications, such as hardware heterogeneity and distributed storage. Beyond the formulation of the problem and the description of the scheduling technique, a prototype has been implemented and tested on a medium-size cluster. The experiments study, separately, the scheduler's ability to: meet high level performance goals guided by user-defined completion time goals;

- favor data locality;
- Deal with hardware heterogeneity.

Introducing hardware affinity and relative performance characterization. Results presented in this paper are partially based on our previous work. In this paper integrate previous partial contributions to build a complete scheduling approach that faces the challenges of the management of data analytics applications executed on next generation data centers. And also extend previous work with our scheduling proposal to enhance data locality and the experiments to evaluate it. The main contribution of this work is to overcome the problem of data overload in the server. The server sends the sender data to the receiver if it is in online status, otherwise it has been stored in the server. Moreover the main server storage of un-received data is too high. So the Dynamic resource allocation algorithm using the main server of un-received storage data forward to physical node of related server. Finally, the unreachable storage data delivered from related server to the particular receiver. So, every time free storage space and speed process in this server.

II . RELATED WORK

Process scheduling is a deeply explored topic for parallel applications, considering different type of applications, different scheduling goals and different platform architectures ([10]). There has also been some work focused on adaptive scalable schedulers based on job sizes ([11], [12]), but in addition to some of these ideas, our proposed scheduler takes advantage of one of the key features of MapReduce: the fact that jobs are composed of a large number of similar tasks.

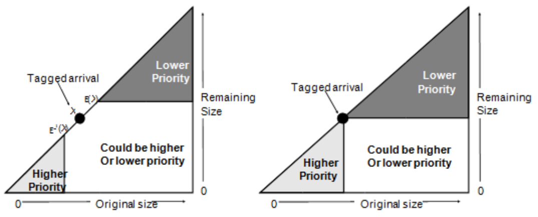

Fig2: Definition of -SMART (Definition 1) and SMART

MapReduce scheduling has been discussed in the literature, and different approaches have been presented. The initial scheduler provided by the Hadoop distribution uses a very simple FIFO policy, considering five different application priorities. In addition, in order to isolate the performance of different jobs, the Hadoop project is working on a system for provisioning dedicated Hadoop clusters to applications [13], but this approach can result in resource underutilization. There are several proposals of fair scheduling implementations to manage data-intensive and interactive applications executed on very large clusters for MapReduce environments ([6], [7]) and for Dryad ([14], [15]).

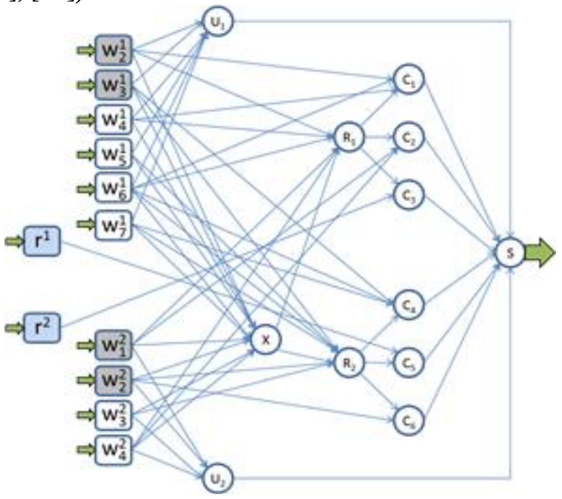

Fig 3: The Quincy flow-based scheduler graph. The figure shows a graph corresponding to the same set of jobs and tasks as the queue based scheduler in Figure 3. There are nodes in the graph for each root and worker task that the scheduler knows about, as well as an "unscheduled node" U_j for each job j. There is also a node C_m for each computer m, a "rack aggregator" node R_l for each rack l, and a "cluster aggregator" node X. S is the sink node through which all flows drain from the graph. Each root task has a single outgoing edge to the computer where it is currently running. Each worker task in job j has an edge to j's unscheduled node U_j, to the cluster-aggregator node X, and to every rack and computer in its preferred lists. Workers that are currently executing (shown shaded) also have an edge to the computer on which they are running. Graph edges have costs and capacities that are not shown in the figure. The Appendix has a detailed explanation of the structure of this graph and the costs and capacities that allow us to map a min-cost feasible flow to a fair scheduling assignment

The main concern of these scheduling policies is to give equal shares to each user and achieve maximum utilization of the resources. However, scheduling decisions are not dynamically adapted based on job progress, so this approach isn't appropriate for applications with different performance goals. There have been other proposals that applications. In addition to our initial implementation [3], others have shown interest in this particular topic. FLEX [16] is a scheduler proposed as an add-on to the Fair Scheduler to provide Service-Level- Agreement (SLA) guarantees. More recently [16] introduces a novel resource management framework that consists of a job profiler, a model for MapReduce jobs and a SLO-scheduler based on the Earliest Deadline First scheduling strategy. In [17], the authors introduce a system to manage and dynamically assign the resources of a shared cluster to multiple Hadoop instances. Priorities are defined by users using high- level policies such as budgets. This system is designed for virtualized environments, unlike the proposed work, which is implemented as a regular Hadoop MapReduce scheduler and thus is able to run on standard Hadoop installations and provide more accurate estimations. Regarding the execution of MapReduce applications on heterogeneous hardware, in [18] the authors consider the influence that hardware heterogeneity may have on the scheduling of speculative tasks. Our proposal in orthogonal to this one as we do not face the scheduling of speculative tasks and we have not enable this option in the configuration of our execution environment. In [19] the authors focus on avoiding stragglers (which may cause the execution of speculative tasks). They show that most of them are due to network traffic. Thus, although dealing with stragglers is not the focus of our proposal, our scheduler is also avoiding them as the percentage of local task that it is able to achieve is around 100%. There are several works in the literature that consider the heterogeneity trend on current execution platforms. Studies the impact of heterogeneity on large clusters and presents techniques to include task placement constraints. More recently, Hadoop schedulers have focused on being more aware of both resources available in each node and resources required by applications. In [8] [9] we adapt the Adaptive Scheduler to be resource-aware.

III. PROPOSED METHOD

The proposed system overcomes the problem of data overload in the server. The server sends the sender data to the receiver if it is in online status, otherwise it has been stored in the server. Moreover the main server storage of un-received data is too high. So the Dynamic resource allocation algorithm using the main server of un-received storage data forward to physical node of related server. Finally, the unreachable storage data delivered from related server to the particular receiver. So, every time free storage space and speed process in this server.

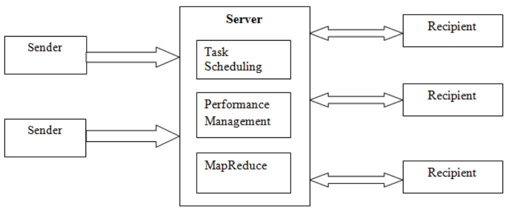

Fig4: System architecture

A. Time Based Task Assignment

A Map Reduce job has two different types of tasks, depending on the execution phase of the job: file tasks and arrival time tasks. In order to get suitable accuracy in the job performance estimation, estimate the performance for each job phase, file and attain time, separately. The task assignment is assigned by the abuser. The abuser sends the files to the other node. And also, the user sets the arrival time of the file which they want to be sent. In this task assignment the abuser state the receiver ID, files and the time to attain the recipient node. Study on task assignment strategies for a complex real-time network system is presented. Firstly, two task assignment strategies are proposed to improve previous strategies. The proposed strategies assign tasks with meeting end-to-end real-time constraints, and also with optimizing system utilization through period modulation of the tasks. Consequently, the strategies aim at the optimization to optimize of system performance with while still meeting real-time constraints. The proposed task assignment strategies are devised using the genetic algorithm switch heuristic real-time constraints in the generation of new populations.

The strategies are differentiated by the optimization method of the two objectives-meeting end-to-end real-time constraints and optimizing system utilization: the first one has sequential genetic algorithm routines for the objectives, and the second one has one multiple objective genetic algorithm routine to find a Pareto solution. Secondly, the performances of the proposed strategies and a well-known existing task assignment strategy using the BnB (Branch and Bound) optimization are compared with one other through some simulation tests. Through the comparison of the simulation results, the most adequate task assignment strategies are proposed for some as system requirements-: the optimization of system utilization, the maximization of running tasks, and the minimization of the number of network node nodes number for a network system

B. Dynamic Scheduling Computation

In this module, the server collects and stores data received from multi user and then it observes the data stored in server and it checks the data and reordering that data based on user mentioned file sending time.

Security requirements of security-critical real-time applications must be met in addition to satisfying timing constraints. However, conventional real-time scheduling algorithms ignore the applications' security requirements. In recognition that an increasing number of applications running on clusters demand both real-time performance and security, we investigate the problem of scheduling a set of independent real-time tasks with various security requirements. Propose a security overhead model that is capable of measuring security overheads incurred by security-critical tasks. Further, we propose a security-aware scheduling strategy, or SAREC, which integrates security requirements into scheduling for real-time applications by employing our security overhead model. To evaluate the effectiveness of SAREC, we implement a security-aware real-time scheduling algorithm (SAREC-EDF), which incorporates the earliest deadline first (EDF) scheduling algorithm into SAREC. Extensive simulation experiments show that SAREC-EDF significantly improves overall system performance over three baseline scheduling algorithms (variations of EDF) by up to 72.55%. A strategy SAREC for security-aware scheduling of real-time applications on clusters. This strategy is capable for the design of security-aware real-time scheduling algorithms like SAREC-EDF. To make security-aware scheduling algorithms practical, we also proposed a security overhead model to measure overheads of security services.

C. Execution Environment

Evaluate the ability of the scheduler to dynamically manage heterogeneous pools of hardware. For these experiments we use a heterogeneous cluster, consisting of regular nodes and nodes enabled with acceleration support, to evaluate the scheduler with hardware affinity. To simulate an environment in which only some of the nodes are enabled. The files are send to the enabled node at the state of the time to assign in the modeling job performance task.

A Trusted Execution Environment (TEE) is a secure area that resides in the application processor of an A Trusted Execution Environment (TEE) is a secure area that resides in the application processor of an electronic device. To help visualize, think of a TEE as somewhat like a bank vault. A strong door protects the vault itself (hardware isolation) and within the vault, safety deposit boxes with individual locks and keys (software and cryptographic isolation) provide further protection. Separated by hardware from the main os, a TEE ensures the secure storage and processing of sensitive data and trusted applications. It protects the integrity and confidentiality of key resources, such as the user interface and service provider assets. A TEE manages and executes trusted applications built in by device makers as well as trusted applications installed as people demand them. Trusted applications running in a TEE have access to the full power of a device's main processor and memory, while hardware isolation protects these from user installed apps running in a main operating system. Software and cryptographic isolation inside the TEE protect the trusted applications contained within from each other.

Device and chip makers use TEEs to build platform that have trust built in from the start, while service and content providers rely on integral trust to start launching innovative services and new business opportunities. To help visualize, think of a TEE as somewhat like a bank vault. A strong door protects the vault itself (hardware isolation) and within the vault, safety deposit boxes with individual locks and keys (software and cryptographic isolation) provide further protection.

D. Dynamic Re-Scheduling

After completion of the scheduling the server rechecks the storage whatever files to be send. The server dynamically schedules a time for sending remaining file in the storage. The server sends the re-scheduled files when the receiver node comes to online mode. Our technique dynamically adjusts the time of available execution nodes across jobs so as to meet their complete a file sending process.

Today’s manufacturing businesses are facing immense pressures to react rapidly and robustly to dynamic fluctuations in demand distributions across products and changing product mix. Traditional manufacturing systems and approaches to production, involving sequential stages from manufacturing systems design, construct, and setup in the preparation phase to production planning, scheduling, and control in the operational phase, can be challenging in satisfying the requirement of the variation. Efficient and practical methods for scheduling and optimization technology are the key to improve the productivity and efficiency of a manufacturing plant. The dynamic interference of these factors causes that the original dynamic scheduling cannot be implemented successfully. Therefore, the rescheduling model and its solution method are of significant importance for the dynamic scheduling problem.

Dynamic rescheduling method is widely used in the modern production plant. In this paper, the Contract Net Protocol, which is based on MAS, is introduced to the rescheduling of the workshop environment. It is a new way of solving the communication and negotiation problem in this field. After fully considering the effecting of the equipment failure and repairmen in the process of production, the complex dynamic rescheduling process is to be divided into the communication and negotiation processes of multi agents. Therefore, the capability of autonomic decision for tackling the unexpected events, which occur in the production, is extended. By simulation in the actual production workshop, the model and algorithm, which are based on MAS, were identified as effective to the rescheduling problem in the manufacturing system. It is worth pointing out that the test cases studied in this work are not very many. It will explore the efficiency of our model and approach on those problems with a larger number of decision variables in the future.

IV. CONCLUSION

Dynamic map-reduce technique presents, avoiding overload in server at scheduling technique for multi-job MapReduce environments, and demonstrate its allocation method. The technique dynamically adjusts the allocation of available execution slots across jobs so as to meet their completion time goals, provided at submission time. The system continuously monitors the average task length for all jobs in all nodes, and uses this information to calculate and adjust the expected completion time for all jobs. Dynamic resource allocation algorithm using the main server of un-received storage data forward to physical node of related server. Finally, the unreachable storage data delivered from related server to the particular receiver. So, project reducing the total volume of network traffic for a given workload.

REFERNCES

[1] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in Proc. 2004 Symposium on Operating System Design and Implementation, pp. 137–150.

[2] Y. Becerra, V. Beltran, D. Carrera, M. Gonzalez, J. Torres, and E. Ayguade´, “Speeding up distributed MapReduce applications using hardware accelerators,” in Proc. 2009 IEEE International Conference on Parallel Processing, pp. 42–49.

[3] J. Polo, D. Carrera, Y. Becerra, M. Steinder, and I. Whalley, “Performance-driven task co-scheduling for MapReduce environments,” in Proc. 2010 IEEE/IFIP Network Operations and Management Symposium, pp. 373–380.

[4] J. Polo, D. Carrera, Y. Becerra, V. Beltran, J. Torres, and E. Ayguade, “Performance management of accelerated MapReduce workloads in heterogeneous clusters,” in Proc. 2010 International Conference on Parallel Processing, pp. 653–662.

[5] Apache Software Foundation, Hadoop MapReduce Tutorial. Available: http://hadoop.apache.org/mapreduce/

[6] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Job scheduling for multi-user Map-Reduce clusters," Tech. Rep. UCB/EECS-2009-55, 2009.

[7] "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in Proc. 2010 European Conference on Computer Cystems, pp. 265–278.

[8] J. Polo, C. Castillo, D. Carrera, Y. Becerra, M. Steinder, I. Whalley, J. Torres, and E. Ayguade´, "Resource-aware adaptive scheduling for MapReduce clusters," in 2011 ACM/IFIP/USENIX International Mid- dleware Conference.

[9] J. Polo, "Adaptive Scheduler," 2009. Available: https://issues.apache.org/ jira/browse/MAPREDUCE-1380

[10] D. G. Feitelson and L. Rudolph, "Parallel job scheduling: issues and approaches," in JSSPP, 1995, pp. 1–18.

[11] P. R. Jelenkovic, X. Kang, and J. Tan, "Adaptive and scalable comparison scheduling," in 2007 SIGMETRICS.

[12] A. Wierman and M. Nuyens, "Scheduling despite inexact job-size in- formation," in Proc. 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 25–36.

[13] Apache Hadoop on Demand. Available: {\scriptsizehttp://hadoop. apache.org/core/docs/r0.20.0/hod\ user\ guide.html}

[14] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in 2007 ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 59–72.

[15] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Gold- berg, "Quincy: fair scheduling for distributed computing clusters," in 2009 ACM SOSP.

[16] J. Wolf, D. Rajan, K. Hildrum, R. Khandekar, V. Kumar, S. Parekh, K.- L. Wu, and A. Balmin, "FLEX: a slot allocation scheduling optimizer for MapReduce workloads," in Middleware 2010, vol. 6452, pp. 1–20.

[17] A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: automatic resource inference and allocation for MapReduce environments," 2011 IEEE International Conference on Autonomic Computing.

[18] T. Sandholm and K. Lai, "MapReduce optimization using regulated dynamic prioritization," in Proc. 2009 International Joint Conference on Measurement and Modeling of Computer Systems, pp. 299–310.

[19] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in Proc. 2008 USENIX Conference on Operating Systems Design and Implementation, pp. 29–42.

[20] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in Map-Reduce clusters using Mantri," in Proc. 2010 USENIX Conference on Operating Systems Design and Implementation, pp. 1-16