Dynamic MapReduce Workload Management in Multiple Job Scheduling

Mrs. R. Adaline Suji[1] Miss. K. Nandini[2]
[1] M. E. & (Ph. D) Assistant Professor [2] M. E. (Final Year)
[1, 2] RatnaVel Subramaniam College of Engineering & Technology

Abstract---A scheduling algorithm & technique for managing multi-job map reduce workloads that rely on the ability to dynamically build performance models of the executing workloads, and use these models to provide dynamic performance management. Sometime the server based workloads may happen due to status of the receiver which is not in online. This may cause the data not to be delivered to the receiver. So, the server storage is too high (overloaded). In this paper to overcome the overloading problem, a subserver is used which is related to the main server. If the receiver node take too much of time to come online, then the sub server send the file to receiver using mail account with encrypted data. The data can be decrypted only when the receiver uses the sender mentioned IP address. If the system IP address does not match to the sender defined IP address the file cannot open.

Keywords: Dynamic Map reduce, Scheduling, Network Management

I. INTRODUCTION ABOUT THE SPHERE

A. *Domain description*

Network management refers to the broad subject of managing computer networks. There exists a wide variety of software and hardware products that help network system administrators manage a network.

B. *Benefits of Network Management*

Security: Ensuring that the network is protected from unauthorized users. Performance: Eliminating bottlenecks in the network. Reliability: Making sure the network is available to users and responding to hardware and software malfunctions.

II. SYSTEM ANALYSIS

A. *Existing Method*

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 design point of Map Reduce platforms. 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. 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 MapReduce framework is mainly based Adaptive scheduler to maximize data locality across working sets, in an attempt to reduce network bottlenecks and increase (where possible) overall system throughput. Data locality is achieved when data is stored and processed on the same physical nodes. 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.

B. *Limitations of Existing System*

- Scheduling capacity is high
- Jobs overloading
- Node approach level is normal
- Manual removal processing

C. *Architecture Diagram*

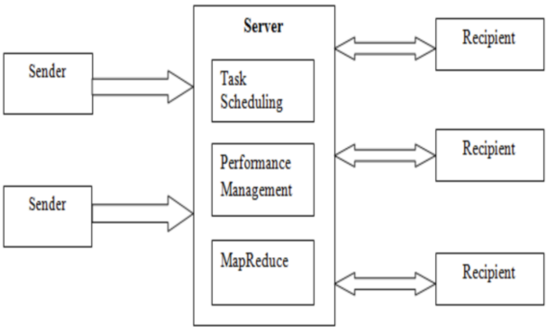

Fig. 1: Architecture Diagram of Existing Method

This architecture diagram represents the existing model of having multiple senders and multiple receivers governed by the centralized server. The centralized server follows different techniques including task scheduling, performance management and map reduce. This will help the server in scheduling the multiple jobs.

III. PROPOSED METHOD

The below architecture diagram represents the proposed model which additionally includes sub server. This sub server follows the Mailing concepts with the help of encryption and decryption techniques.

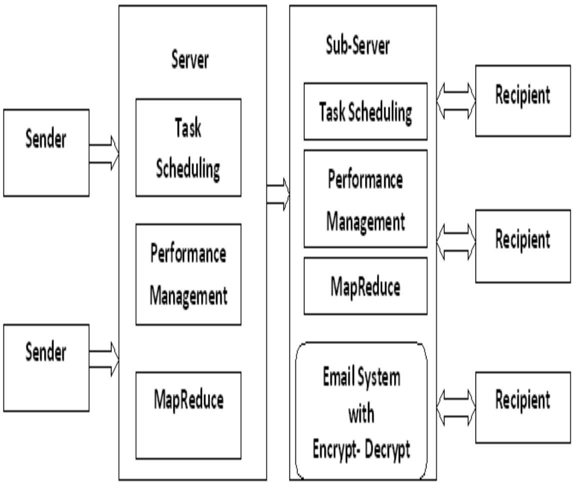

Fig.2. Architecture Diagram of Proposed Method

The proposed system overcomes the problem of data overload in the server. The server sends the sender data to the receiver if it is (receiver) 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. The file is send to the receiver through mail when it takes a long time to come up to online. The receiver downloads the file as encrypted. The file decrypt on only in a correct IP addresses that defined by sender node. If the system IPad dress is not match to the sender defined IP address the file doesn't open.

A. Advantages of Proposed System

- Easy jobs maintenance
- Server response is high
- Automatic removal processing□
- Free storage space

IV. PROPOSED ALGORITHM

A. Resource Allocation Algorithm:

Resource allocation is used to assign the available resources in an economic way. It is part of resource management. In project management, resource allocation is the scheduling of activities and the resources required by those activities while taking into consideration both the resource availability and the project time. Resource allocation may be decided by using computer programs applied to a specific domain to automatically

& dynamically distribute resources to applicants.

B. Steps:

```
Lock_resources(S)
/*S is the set of resources to lock*/
lock(Lock Manager)
get timestamp
i:=0
while i<[s] do
i:=i+1
trylock($S_i$)
IF could not lock $S_i$
add self to $S_i$ wait queue
for j:=1 to i do unlock $S_j$ end
unlock(LockManager)-signal    HANDOFF-Wait
i:=0
end
end
signal HAOFF
IF this subsystem still holds a lock Manager
unlock(Lock Manager)
End
```

C. We show the results for the following two configurations:

1. Shows how the Adaptive Scheduler arbitrates allocation inside the accelerated pool when accelerable jobs are competing. In addition, the tight deadline for accelerable jobs forces the scheduler to allocate them nodes from the non-accelerated pool.
2. Illustrates how a non accelerable job can steal nodes from the accelerated pool when needed to meet its completion time goal.

V. MODULE DESCRIPTION

A. Time Based Task Assignment

A MapReduce 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, we 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.

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.

C. Performance Management

The server checks the recipient node is available or not when the time to send the file. If the node are disable means the server could not sends the file at the scheduling time where declare in the arrival time tasks. At this time, the server stores the files in the same server location. In this performance management task, the server automatically remove files when they hurl (send) to the accurate node.

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.

E. Resource Allocation with Security

In this module, the files in waiting list are sending to sub-server from main server. The

Sub-server sends that file to destination IP. Suppose that the receiver IP is in offline mode means, it stores that files in waiting list. After completed of time, sub-server sends the notification message to sender. The sender sends Receiver Email-ID to sub-server and then sub-server forward the file encrypted to receiver Mail-ID. The receiver downloads that file decrypted only from particular IP address

VI. TECHNIQUE

A. Hidden Vector Encryption Technique:

Predicate encryption schemes are encryption schemes in which each ciphertext Ct is associated with a binary attribute vector x=(x1... xn) and keys K are associated with predicates. A key K can decrypt a ciphertext

Ct if and only if the attribute vector of the ciphertext satisfies the predicate of the key. Predicate encryption schemes can be used to implement fine-grained access control on encrypted data and to perform search on encrypted data. Hidden vector encryption scheme are encryption schemes in which each ciphertext Ct is associated with a binary vector x = (x1... xn) and each key K is associated with binary vector y = (y1... yn) with "don't care" entries (denoted with ☆). Key K can decrypt ciphertext Ct if and only if x and y agree for all *i* for which y*i* is not. Hidden vector encryption schemes are an important type of predicate encryption schemes as they can be used to construct more sophisticated predicate encryption schemes (supporting, for example, range and subset queries). We give a construction for hidden-vector encryption from standard complexity assumptions on bilinear groups of prime order. Previous constructions were in bilinear groups of Composite order and thus resulted in less efficient schemes. Our construction is both payload-hiding and attribute-hiding meaning that also the privacy of the attribute vector, besides privacy of the clear text, is guaranteed.

VII. CONCLUSIONS

In this paper we present a scheduling technique for multi job Map Reduce environments, and demonstrate its efficacy through experiment implementation and evaluation on the Hadoop platform. Our 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. Beyond completion time objectives, the presented scheduler also observes two additional high-level performance goals: first, it enforces data locality when possible, reducing the total volume of network traffic for a given workload; and secondly, it is also able to deal with hybrid machines, composed of generic processors and hardware accelerators that can carry specialized tasks. In our experiments, we compare the results for our scheduler with a state of the art Hadoop Scheduler, and show how our scheduler enables system administrators to define and predict the performance of the system more accurately and easily, according to user-defined high-level performance metrics.

VIII. FUTURE WORK

Several key functions of AJS (Advanced Job Scheduling) are now available from a Web browser. In future you can remotely access scheduled jobs, job groups, scheduled job activity and activity log information for AJS. When listing scheduled jobs and scheduled job activity, any column can be used for sorting or filtering. This is great for selecting jobs that begin with certain letters and then sorting by the next scheduled date and time. While you can’t make changes to jobs this release, you have many ways to manage your scheduled jobs.

REFERENCES

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

[2] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, “Modeling and synthesizing task placement constraints in Google compute clusters”.

[3] Los Alamos National Laboratory, HPC: Roadrunner.

[4] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini, “An energy case for hybrid datacenters,” in 2009 Power Aware Computing and Systems.

[5] M. Gschwind, P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, “A novel SIMD architecture for the cell heterogeneous chip-multiprocessor,” 2005.