ISSN No: - 2456 - 2165

An Efficient Degraded Primary Scheduling Algorithm For Virtual Mapreduce Clusters

Mr.S.S.Aravinth /AP / CSE / Knowledge Institute of Technology, Salem
M.Mangaiyarkarasi / II-M.E / CSE / Knowledge Institute of Technology, Salem
A.kesavini / I-M.E / CSE / Knowledge Institute of Technology, Salem

Abstract:- The witnessed an increasing adoption of erasure coding in modern clustered storage systems to reduce the storage overhead of traditional 3-way replication. However, it remains an open issue of how to customize the data analytics paradigm for erasure-coded storage, especially when the storage system operates in failure mode. The propose degraded first scheduling, a new MapReduce scheduling scheme that improves MapReduce performance in erasure-coded clustered storage systems in failure mode. Its main idea is to launch degraded tasks earlier so as to leverage the unused network resources. The proposes degraded-first scheduling algorithm, whose main idea is to schedule some degraded tasks at earlier stages of a MapReduce job and allow them to download data first using the unused network resources. The experiment conduct mathematical analysis and discrete event simulation to show the performance gain of degraded-first scheduling over Hadoop's default locality-first scheduling.

Keywords: *Degraded First Scheduling Algorithm, Mathematical Analysis And Discrete, Erasure-Coded Storage*.

I. INTRODUCTION

A. Cloud Computing

Cloud computing has been envisioned as the next generation information technology structural design for enterprises, suitable to its extensive list of extraordinary advantages in the IT record on-demand self-service, everywhere network access, locality independent reserve pooling, rapid resource spring, usage-based pricing and conversion of threat. As a troublemaking knowledge with reflective implications, and the cloud computing is transforming to the very nature of how businesses use information technology. One essential feature of this prototype variable is that data are being federal or outsourced to the cloud. From users perception, together with both the individuals and IT enterprises, storing the data slightly to the cloud in a flexible on-demand manner brings pleasing benefits, the relief of load for storage space organization, worldwide data contact with location independence and the avoidance of capital disbursement on hardware, software and the personnel maintenances, etc.,

B. Cloud Management Challenges

Cloud computing presents a figure of organization challenges. Companies are using public clouds do not have tenure of the tools hosting the cloud setting and because of the environment is not having within their individual networks, public cloud customers don't have full visibility or control. Users of the public cloud services must also contain integrate with an planning distinct by the cloud provider and using its exact parameters for running with cloud mechanism. Arrangement includes the cloud APIs for configuring IP addresses, firewalls, subnets and data service functions for the storage. Because power of these functions is based on the cloud provider's communications and services public cloud users must combine among the cloud infrastructure organization.

Hybrid cloud which merge public and private cloud services, occasionally with conventional transportation essentials, present their possess set of administration challenges. These include protection concerns if sensitive data lands on public cloud servers, finances concerns around over use of the storage or bandwidth and the large number of mismanaged images. Managing the information flow in hybrid cloud environment is also a significant dispute. The clouds must share in sequence with applications hosted off-premises by public cloud providers and this information may change continually.

II. SYSTEM ANALYSIS

A. Existing System

The FIFO algorithm is a default scheduling algorithm provided by Hadoop MRv1. It follows a strict job submission order to schedule each map task of a job and meanwhile attempts to schedule a map task to an idle node that is close to the corresponding map-input block. However, the FIFO algorithm only focuses on map-task scheduling, rather than reduce-task scheduling. Hence, when FIFO is adopted in a virtual MapReduce cluster, its low reduce-data locality might cause a long job turnaround time. Besides, FIFO is designed to achieve node locality and rack locality in conventional

ISSN No: - 2456 - 2165

MapReduce clusters, rather than achieving the VPS-locality and Cen-locality in a virtual MapReduce cluster. Consequently, the map-data locality of FIFO might be low in a virtual MapReduce cluster.

MapReduce task scheduling presented the delay scheduling algorithm to improve data locality by following the FIFO algorithm but relaxing the strict FIFO job order. If the scheduling heuristic cannot schedule a local map task, it postpones the execution of the corresponding job and searches for another local map task from pending jobs. A similar but improved approach. However, similar to FIFO, this approach did not provide reduce-task scheduling. The Balance-Reduce (BAR) algorithm, which produces an initial task allocation for all map tasks of a job and then takes network state and cluster workload into consideration to interactively adjust the task allocation to reduce job turnaround time. In order to simplify BAR, the authors assumed that all local map tasks spend identical execution time. But this assumption is not realistic since the map-task execution time fluctuates even though when the processed input size is the same. Besides, reduce-task scheduling was not addressed by BAR.

MapReduce workload prediction mechanism to classify MapReduce workloads into three categories based on their CPU and I/O utilizations and then proposed a Triple-Queue Scheduler to improve the usage of both CPU and disk I/O resources under heterogeneous workloads. An optimal map-task scheduling algorithm, which converts a task assignment problem into a Linear Sum Assignment Problem so as to find the optimal assignment. Nevertheless, applying this algorithm to real-world MapReduce.

Clusters needs to carefully determine an appropriate time point to conduct the algorithm since slaves might become idle at different time points. Introduced a co-scheduler called LiPS, which utilizes linear programming to simultaneously co-schedule map-input data and map tasks to nodes such that dollar cost can be minimized.

B. Drawbacks

- Less average implementation time of jobs when the number of tasks is less than the cluster size.
- Tasks progress at a constant rate throughout time.
- Each task Tracker has a fixed number of map slots and reduce slots, configured by the administrator in advance.
- It organizes jobs into pools and shares resources fairly across all pools based on max-min fairness.
- Each pool consists of two parts: map-phase pool and reduce-phase pool. Within each map/reduce-phase pool, the fair sharing is worn to map/reduce slots between the management jobs at all phase. Pools can also be weights to distribute the cluster non proportionally in the pattern file.
- There is no cost to launching a speculative task on a node that would otherwise have an idle slot.
- A task's progress score is representative of fraction of its total work that it has done. particularly, in reduce task, copy and diminish phases each take about 1/3 of the total time.

III. PROPOSED SYSTEM

In the presence of node failures, it re-schedules tasks to run on other nodes that hold the replicas. However, the scenario becomes different for erasure-coded storage, where MapReduce tasks must issue degraded reads to download data from other surviving nodes. Such degraded tasks are typically scheduled to launch after all local tasks are completed, and when they launch, they compete for network resources to download data from surviving nodes. This can significantly increase the overall runtime of a MapReduce job.

Traditional MapReduce scheduling emphasizes locality, and implements locality-first scheduling by first scheduling local tasks that run on the nodes holding the input data for the tasks. MapReduce is designed with replication-based storage in mind. In the presence of node failures, it re-schedules tasks to run on other nodes that hold the replicas. However, the scenario becomes different for erasure-coded storage, where MapReduce tasks must issue degraded reads to download data from other surviving nodes. Such degraded tasks are typically scheduled to launch after all local tasks are completed, and when they launch, they compete for network resources to download data from surviving nodes. This can significantly increase the overall runtime of a MapReduce job. Thus, a key motivation of this work is to customize MapReduce scheduling for erasure coded storage in failure mode.

The degraded-first scheduling, whose main idea is to move part of degraded tasks to the earlier stage of the map phase. The advantages are two-fold. First, the degraded tasks can take advantage of the unused network resources while the local tasks are running. Second, it avoids the network resource competition among degraded tasks at the end of the map phase. The first present the basic version of degraded-first scheduling. The conduct mathematical analysis to show the improvement of degraded-first scheduling over the default locality-first scheduling in Hadoop. Finally, present the enhanced version of degraded-first scheduling that takes into account the topological configuration of the cluster.

IV. MODULES DESCRIPTION

A. Map Reduce Model

Hadoop cluster composed of multiple nodes (or servers) that are grouped into different racks. Typical clusters connect all nodes via a hierarchy of switches. Without loss of generality, we consider a simplified two level case where

ISSN No: - 2456 - 2165

nodes within each rack are connected via a top-of-rack switch, and all the racks are connected via a core switch. Hadoop runs on a distributed file system HDFS for reliable storage. HDFS divides a file into fixed-size blocks, which form the basic units for read and write operations. Since node failures are common, HDFS uses replication to maintain data availability, such that each block is replicated into multiple (by default, three) copies and distributed across different nodes.

B. Erasure Coding

To reduce the redundancy overhead due to replication, erasure coding can be used. An erasure code is defined by parameters (n, k), such that k original blocks (termed native blocks) are encoded to form n−k parity blocks, and any k out of the n blocks can recover the original k native blocks. We call the collection of the n blocks a stripe. Examples of erasure codes include Reed-Solomon codes and Cauchy Reed-Solomon codes. Hadoop's authors propose a middleware layer called HDFS-RAID, which operates on HDFS and transforms block replicas into erasure-coded blocks. HDFS-RAID divides a stream of native blocks into groups of k blocks, and encodes each group independently into a stripe according to the parameters (n, k).

C. Degraded First Scheduling

The design goal is to evenly spread the launch of degraded tasks among the whole map phase. This design goal follows two intuitions.

- Finish running all degraded tasks before all local tasks. If some degraded tasks are not yet finished after all local tasks are finished, they will be launched together and compete for network resources for degraded reads.
- Keep degraded tasks separate. If two or more degraded tasks run almost at the same time, they may compete for network resources for degraded reads.

The key challenge here is how to determine the right timing for launching degraded tasks, so that they are evenly spread among the whole map phase. One possible solution is to predict the overall running time of the whole map phase and launch degraded tasks evenly within the predicted time interval. However, this approach is difficult to realize for two reasons. First, different MapReduce jobs may have highly varying processing time of a map task. Thus, it is difficult to accurately predict how long the whole map phase would be. Second, even if we can make accurate predictions, it is possible that no free map slots are available when a degraded task is ready to launch. Thus, the launch of some degraded tasks may be delayed, defeating the original purpose of evenly spreading the degraded tasks.

D. Locality Preservation &Rack Awareness

The default locality-first scheduling achieves high locality by first launching local tasks whenever they are available. On the other hand, Algorithm 2 may break the locality. Specifically, if we first assign degraded tasks to a node, then the node may not have enough map slots to process its local tasks. The master may instead assign some of the local tasks of the node to other nodes of different racks, and these tasks become remote tasks. Having additional remote tasks is clearly undesirable as they compete for network resources as degraded tasks do. We provide a function ASSIGNTOSLAVE to determine whether to launch a degraded task to a specific slave. The point out that our implementation also works for heterogeneous settings, where some slaves may have better processing power than others in the same cluster. If we estimate the processing time for the local map tasks based on not only the number of local map tasks, but also the computing power of the slave node, then we allow the more powerful slaves to process a degraded task while processing more local map tasks.

V. CONCLUSION

The proposed system explores the feasibility of running data analytics in erasure-coded clustered storage systems. It present degraded-first scheduling, a new MapReduce scheduling scheme designed for improving MapReduce performance in erasure-coded clustered storage systems that run in failure mode. It shows that the default locality-first scheduling launches degraded tasks at the end, thereby making them compete for network resources. Degraded-first scheduling launches degraded tasks earlier to take advantage of the unused network resources. It also proposes heuristics that leverage topological information of the storage system to improve the robustness of degraded-first scheduling. It conducts simple mathematical analysis and discrete event simulation to show the performance gains of degraded-first scheduling.

VI. FUTURE ENHANCEMENT

Minimal Interference Maximal Productivity (MIMP) system, which enhances both the hypervisor's scheduler and the Hadoop job scheduler to better manage their performance. Our primary contributions include:

- A new priority level built into Xen's Credit Scheduler that prevents batch processing VMs from hurting interactive VM performance.
- Task affinity models that match each Hadoop task to the dedicated or shared VM that will provide it the most benefit.

- A deadline and progress aware Hadoop job scheduler that allocates resources to jobs in order to meet performance goals and maximize the efficiency of a hybrid cluster.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2] S. Chen and S. Schlosser, "Map-Reduce meets wider varieties of applications," Intel Res., Santa Clara, CA, USA, Tech. Rep. IRPTR-08-05, 2008.

[3] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in Proc. IEEE 4th Int. Conf. eScience, Dec. 2008, pp. 222–229.

[4] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in mapreduce," in Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., May 2012, pp. 419–426.

[5] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in Proc. 5th Eur. Conf. Comput. Syst., Apr. 2010, pp. 265–278.