ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

Improving the QoS of Cloud Data Storage Services

Prof.J.Gowrishankar[1], S.P.Kausalya[2], N.Nagapriya[3]

[1]Assistant Professor, Knowledge Institute of Technology, Salem, Tamil Nadu, India

[2&3]PG Student, Knowledge Institute of Technology, Salem, Tamil Nadu, India

ABSTRACT: The cloud computing is an emerging technology containing multiple features and uses. Hence the interest on cloud technologies also increased among the technologists and also it is still having its peak in the market. In this paper we discussed on MapReduce which is an emerging feature in cloud computing. It has a highly scalable programming paradigm that enables high-throughput data-intensive processing as a cloud service. The performance of MapReduce depends on the storage service of cloud. In this paper, we discussed on the MapReduce deployments which are backed up by the real cloud middleware. A stable throughput can be maintained at the storage level, so that the I/O operations can be easily monitored. This in turn increases the efficiency of scheduling algorithms and reveals new options for management at the upper layers of cloud. This is main aim of this paper.

KEYWORDS: Map Reduce, storage service, stable throughput

I. INTRODUCTION

The cloud computing is an emerging technology with lots of developments day by day. By this technology, even a limited financial users too can invest on it. Because of its simplicity and flexibility, it has become the most wanted tool today. According to this technology, we do not want to have any hardware or physically huge components to have it but still we need some more knowledge about cloud to efficiently use it. People can use it as like temporary storage for their provisional purpose. That is, we can rent some resources which is needed by us and we can only pay for what we have used. Cloud technologies have lots of platforms with their replaceable need.

But the needs of these expertises are increasing daily. In order to make these resources available to all the needies, it has been given a new paradigm called MapReduce. This feature has become more familiar in the field of cloud as soon as it was implemented. MapReduce applications are mostly data-intensive and are also critical components of cloud infrastructure. It is the underlying storage device which needs to deliver a highly aggregated throughput even under a heavy data access concurrency generated when the service runs for a very large number of clients. The data storage device should not only able to support high aggregated throughput for heavy access of data but also to maintain a stable aggregate throughput for each individual access of the data. This paper deals with this MapReduce, to improve the data storage with the stable aggregate throughput though it is a difficult task on cloud.

First, MapReduce frameworks run on infrastructures that comprising thousands of commodity hardware components. Since faults cause drops in performance, fault tolerance becomes a critical aspect of throughput stability. Second, complex data access patterns are generated that are likely to combine periods of intensive I/O activity with periods of relative less intensive I/O activity throughout runtime. This has a negative impact on throughput stability, thus adaptation to access pattern is a crucial issue as well.
Really it is a difficult task to find the non trivial solutions manually. Hence we have to make it automated.

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

II. PROPOSAL

A) Overview

We propose a general methodology to improve the stability of data through stable throughputs to discover and characterize the situations that are responsible for the variations of data during data transfer. The methodologies are:

1. Component monitoring
2. Application-side feedback
3. Behavior pattern analysis

Component monitoring:

The component monitoring is one of the important tasks to be performed to determine where the failures may occur. In large scale infrastructures with thousands of machines, it is impossible to save any of the component from failure if it seems to fail. So every component should be monitored by characterizing each component with large number of parameters. Hence the failures in any component can be easily identified and replaced efficiently.

Application-side feedback:

Though we can find the state of the system at a specific moment in time by using extensive monitoring, we unable to locate where the fluctuation took place at that time. This is because the quality of service from the application point of view is not known to all. For example, in the context of MapReduce applications, monitoring the network traffic is not enough to infer the throughput achieved for each task, because the division of each job into tasks remains unknown to the storage service. Therefore, it is crucial to gather dynamically feedback from the upper layers that rely on the storage service in order to decide when the system performs satisfactory and when it does not.

Behavior pattern analysis:

By using extensive monitoring we come to know the overall behavior of the storage service in time. And also by using the application feedback we can identify when it is worked satisfactorily and when it is worked poorly. The behavior of the storage service can be easily monitored. But the challenge however is to describe the behavior patterns in such a way that they provide meaningful insight with respect to throughput stability, thus making healing mechanisms easy to implement.

Hence we again proposed four methods to stabilize the throughput of the storage service. They are:

- Monitor the storage service
- Identify behavior patterns
- Classify behavior patterns according to feedback:
- Predict and prevent undesired behavior patterns:

Monitor the storage service:

The large number of parameters that are used to illustrate the state of the each component in the large scale infrastructures is periodically collected to monitor the long period of service-up time. An important aspect in this context is fault detection, as information on when and how long individual faults last is crucial in the identification of behavior patterns.

Identify behavior patterns:

After monitoring the parameters, the whole behavior of the storage service is divided into small individual patterns. These patterns must be divided on the basis that if a pattern given at any moment of time should be matched with the pattern of the storage service used. The critical part of this step is to extract meaningful information with respect to throughput stability that describes the behavior pattern.

Classify behavior patterns according to feedback:

The relationship between the behavior patterns and the storage service of throughput is known by analyzing the perceived quality of service. After the classification of behavior patterns, the transitions from desirable state to the undesirable states are analyzed. This is examined to find the reason for the unstable throughput of the storage service. This step involves user-level interpretation and depends on the quality of the generated behavior model.

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

Predict and prevent undesired behavior patterns:

Finally, understanding the reasons for each undesired behavior enables the implementation of prediction and prevention mechanisms accordingly. More specifically, for each undesirable state, the preconditions that trigger a transition to it are determined. Using this information, a corresponding policy is enabled, capable of executing a special set of rules while these preconditions are satisfied. These rules are designed to prevent this transition to occur or, if this is not possible, to mitigate the effects of the undesired state. It is assumed that the application is monitored throughout its execution and the storage service has accessed to the monitoring information.

B. GloBeM: Global Behavior Modeling

In order to identify and describe the behavior patterns of the storage service approach to model the global behavior of large-scale distributed systems(from now on it will be named GloBeM). Its main objective is to build an abstract, descriptive model of the global system state. This enables the model to implicitly explain the interactions between entities, which has the potential to expose non-trivial dependencies essential for the description of the behavior, which otherwise would have gone unobserved. GloBeM obeys a set of methods in order to build such a model, starting from monitoring information that corresponds to the monitored behavior. These basic monitoring data are then aggregated into global monitoring parameters, representative of the global system behavior instead of each single resource individually. This aggregation can be performed in different ways, but it normally consists in calculating global statistic descriptors (mean, standard deviation, skewness, kurtosis, etc.) values of each basic monitoring parameter for all resources present. This ensures that global monitoring metrics are still understandable from a human perspective. This global information undergoes a complex analysis process in order to produce a global behavior representation. This process is sturdily based on machine learning and other knowledge discovery techniques, such as virtual representation of information systems. A behavior model presents the following characteristics: Finite state machine: The model can be expressed as a finite state machine, with specific states and transitions. The number of states is generally small (between 3 and 8). State characterization based on monitoring parameters: The different system states are expressed in terms of the original monitoring parameters. This ensures that its characteristics can be understood and directly used for management purposes. Extended statistical information: The model is completed with additional statistic metrics, further expanding the state characterization.

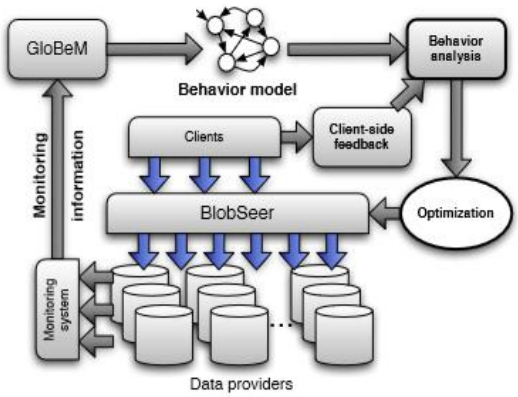

Fig: Stabilizing throughput in Blobseer

C. BlobSeer:

BlobSeer is an efficient distributed data management service particularly designed to deliver a high throughput under heavy access concurrency of data from the storage service. Data are summarized in BlobSeer as the large sequences of bytes called BLOBs. Each BLOB is influenced through a simple versioning access interface that enables fine grained reads, writes and appends of subsequences of bytes from/to the BLOB. Three key design factors enable BlobSeer to achieve a high throughput under heavy access concurrency. They are: data striping, distributed metadata management and versioning-based concurrency control. Data striping: Each BLOB is splitted into chunks and are distributed among the data providers, which are responsible to hold the chunks. To maintain data availability inspite of failures, each chunk is replicated on multiple individual providers. A configurable chunk distribution strategy is

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

allowed when writes and appends are issued in order to optimize chunk placement in such a way that accesses to different chunks are as much as possible handled by different machines, effectively distributing the I/O workload.
Metadata decentralization: BlobSeer uses a distributed metadata management scheme to avoid the bottleneck of accessing the same centralized node and prevent a single point of failure.
Versioning-based concurrency control: Writes and appends to a BLOB never modify its contents, but rather generate a new snapshot of it that looks and acts like the original BLOB save for the applied update. Only the difference is physically stored, with unmodified parts shared. This approach enables the concurrency control to isolate updates in their own snapshot, thus avoiding the need for synchronization significantly better than lock-based approaches, which greatly improves achieved throughput.

III. MAPREDUCE FRAMEWORK FEATURES

The foremost features of MapReduce framework are

1. Scheduling
2. Synchronization
3. Error and fault handling

1. Scheduling:

This feature ensures that in a cluster of data multiple tasks are being executed from multiple jobs. It also ensures optimization in other words called speculative execution which is implemented by Map Reduce framework. It enables additional instance of the same task, if a particular task takes too long to execute.

2. Synchronization:

Between the two phases of Map and Reduce processing, Map Reduce execution enables synchronization since the Reduce phase can start only when all the pairs of map' key or value are emitted.

3.Error and fault handling:

On account of establishing job execution in which errors and faults are the norm, job tracker enables restart execution of failed tasks.

IV. CONCLUSION

In order to enhance the MapReduce distributed data- intensive applications on the cloud, the underlying data storage backend has to ensure besides a high aggregated throughput under heavy access concurrency also a stable throughput for each individual data transfer, which is a quality of service constraint. But the complexity of the system's behavior makes reasoning about quality of service a difficult problem, because a lot of distinct factors affect the behavior simultaneously: highly-concurrent data access patterns, long periods of service uptime, failures of physical components, the highly distributed nature of the storage service itself, etc. This paper proposes a new approach to improve QoS in a distributed storage system, based on component monitoring, application-side feedback and global behavior modeling that can be combined to deduce useful knowledge about potential bottlenecks, making reasoning about potential improvements are very much easier.

In near future, we will be working on the development of stable throughput by implementing hadoop feature with different scheduling algorithms and trying to find the most efficient algorithm for stable throughput for storage service on cloud.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a clouddefinition," SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 50–55, 2009.
[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," SIGOPS Oper. Syst. Rev., vol. 41, no. 3, pp. 59–72, 2007.
[4] D. A. Patterson, "Technical perspective: the data center is the computer," Commun. ACM, vol. 51, no. 1, pp. 105–105, 2008.
[5] "Amazon Elastic Map Reduce," http://aws.amazon.com/ elasticmapreduce/.
[6] B. Nicolae, G. Antoniu, L. Boug´e, D. Moise, and A. Carpen-Amarie, "Blobseer: Next generation data management for large scale infrastructures," Journal of Parallel and Distributed Computing, 2010, to appear.009.