ISSN XXXX XXXX © 2018 IJESC

Research Article Volume 8 Issue No.6

Checkpoint and Replication based Fault Tolerance for Map Reduce Framework in Cloud Environment

Monika Raghuwanshi, Prof. Deepak Kumar Gour
Department of Computer Science and Engineering
Technocrats Institute of Technology, Bhopal

Abstract:
This research work focuses on providing an efficient proactive fault tolerant technique for scientific application in which above mentioned fault is predicted proactively and handled efficiently such that all cloudlets finish their tasks before or on deadline and no virtual machine has been left out for execution. The used techniques for the fault tolerance comes in the category of reactive (fault is handled after it has occurred) and proactive (fault is predicted before it actually happens).Much work has been done in the area of reactive techniques.

Keywords: checkpoint-restart, cloud computing, distributed application, infrastructure-as-a-service, virtualization, scalability, self-healing,

I. Introduction

MapReduce programs do not explicitly relate to the division of work and distribution neither of data, nor to the many low-level complications that distributed execution entails. As long as the application obeys certain restrictions imposed by the programming model, the MapReduce engine ensures fault-tolerant and scalable execution. A MapReduce program may thus be developed and tested on a single machine or small cluster, and be deployed with a high degree of confidence on a much larger scale. This makes MapReduce a particularly good match for data-intensive applications executing in the cloud. Amazon's commercial cloud computing platform, Amazon Web Services, includes MapReduce support both as a separate high-level web service [12], and in the form of customized virtual machine images.

The adoption of MapReduce and similar high-level programming models was encouraged by another concurrent development: a new paradigm for software deployment known as cloud computing. The aim of cloud computing is to turn computing power into a common utility that is always available on demand and in abundant supply, much like electrical power delivered through the electricity grid. This long-held dream has recently become economically viable, with the construction and operation of extremely large-scale, commodity-computer data centers at low-cost locations.

Cloud providers allow customers to purchase on-demand access to computing power in various forms, for example as a cluster of virtual machines, provisioned from a much larger underlying cluster of physical machines. Customers may log in remotely and execute their own software on the virtual machines, and purchase access to additional machines whenever the need arises. As a result, computing clusters are now more accessible than ever: there is no need for small companies to make large up-front investments in hardware to provision for peak loads or anticipated future growth—virtual computing clusters hosted in the cloud may simply be expanded on demand, minimizing financial risks.

According to pay-as-you-go model, subscription-based services that are offered in the cloud are infrastructure, platform, and software (applications).

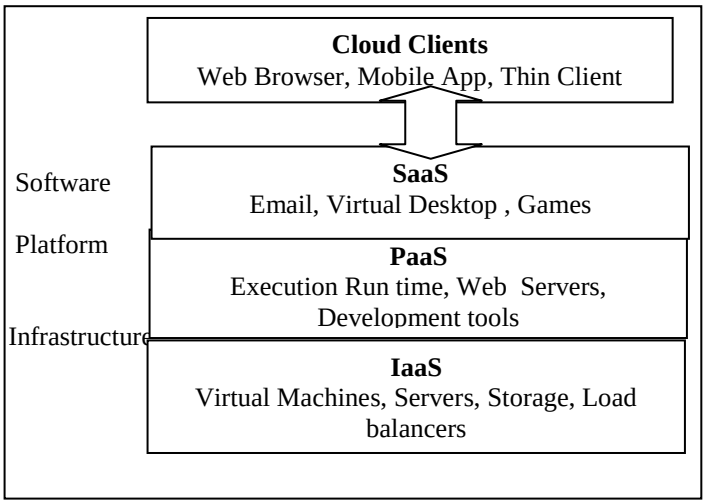

Fig.1 Different service models in cloud computing

Hence these services are provided according to the three different service models Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Depending upon the type of resource customer has requested, provider responds with the associated service model. These models further comprise of different layers essential for specific service. User can access these services anywhere and at any time with the help of internet. Details of these models are as follows and detailed figure is shown as Figure 1.

II. Fault Tolerance in Cloud Computing

Fault tolerance submits to perfect and permanent operation even when there are defective components. Continuing to work satisfactorily in the presence of obstacles is the science that builds the computing system. The efficient fault tolerant system can allow fault of different types like momentary, discontinuous or stable hardware faults, blunders in software design, operative errors, or corporal damage or externally induced upset. In a real-time application of cloud, it is performed remotely on the computing node, and the probability of error occurrence is high.

These error happening events enhance the desire for fault tolerance technology to achieve the efficiency of real-time computing in cloud environment. If a fault occurs in either the component (hardware) or the design method (software), an error will occur. These events enhance the need for fault tolerance technology.

II.1. Types of Fault Tolerance

Fault tolerance is mainly divided into two types. Fault tolerance of hardware and software.

A. Hardware fault tolerance

Mostly computer systems nowadays get recovered by themselves from failures caused due to hardware components. In systems, each component has a back up with protected duplication or redundancy. If one component fails, the other components can perform the function. Example: Mirroring technology to repair faults in storage media. A common hardware fault tolerance approach is fault masking and dynamic recovery.

i. Fault Masking:

It is a kind of structural duplication method that absolutely identifies the errors within a block of similar components. All identical components perform the same function, their output is ranked, and errors caused by the defective module are removed. Triple module redundancy (TMR) is a very form of fault masking in which components are tripled and their output is voted. Voting process for selecting redundant components can also be tripled such that voting process can rectify the failure of personage voters. If two components in the redundant triplet get failed and the vote is no longer valid, then TMR technique fails. Hybrid redundancy is an extension of TMR where triplexes components are always backed up with extra components and are used to replace failing components that can handle more faults. A voting system requires more than three times the hardware of a non redundant system, but it has the advantage of continuing the calculation without interruption in the event of a failure, and you can use an existing operating system.

ii. Dynamic recovery:

Dynamic recovery techniques are used when only single copy of a task or any calculation is executed at one time. This technology performs self-healing method. Also it performs additional backup operations using additional spare components (proactive redundancy) along with fault masking technique. This requires a special mechanism of recovery to detect faults in the module. There is no need of switching to a redundant component in this technique because faults in modules or components are parsed and recovered by performing tasks such as rollback, initialization, retry, restart , restore operation and continue the process.

B. Software fault tolerance

Software failures can be handled by using static as well as dynamic approaches just like hardware fault tolerance approach. In N version programming, static redundancy is used in which a separately written program is used that executes the same operation. Each module in this technique is developed with up to N different implementations or algorithms. Every programmer does the similar task. Every version submits its output to the voter or decision-maker. The voter or the decision maker then decides the correct output and returns it by calling it as the result of the module. Other dynamic approach is a recovery block method. In this method, a program is separated into blocks, and the acceptance test is executed after the execution of each block. In case the acceptance test fails, then redundant code block is used for the execution. In the devise diversity approach, merging of both hardware and software fault tolerance is done to make a failure free computer system using different hardware and software on redundant channels. Main goal of this technology is to withstand both hardware and software failures.

III. Proposed Methodology

In proposed work when cloudlets arrive to the processor then the best fit approach will be applied. The best fit approach here refers to the process of allocation of suitable VM to the cloudlet which will execute it in minimum time. In case, if the best fit is not found or if any of the machine is currently not matching with the job's specifications then the proposed algorithm would eliminate this failure by processing that cloudlet in two ways, either it will split the cloudlet in two parts i.e dividing the cloudlet into two parts on the basis of its size or it will make a request for the new virtual machine. The splitting of the cloudlet is a method in which the cloudlet is divided into two parts and both parts are processed simultaneously which consumes less time and also works efficiently. In case of generating a request for the new virtual machine, memory requirement for that VM is checked. If there is no sufficient memory for the new VM, then that cloudlet will be considered as fail, otherwise a new VM is created for that cloudlet.

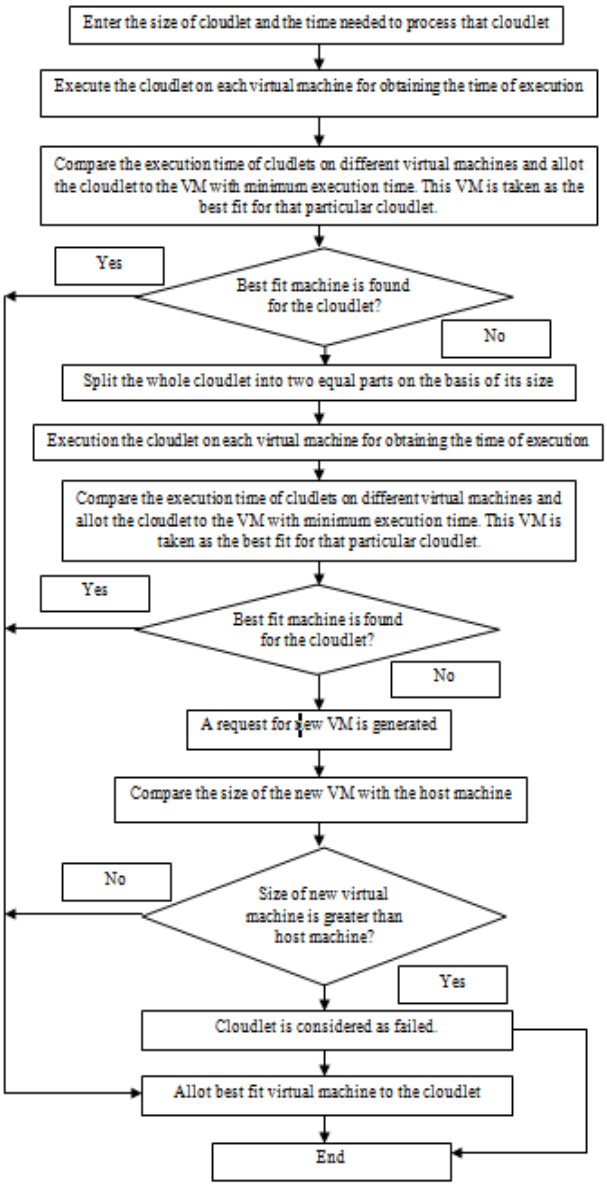

Fig.2 Flow of proposed work

i. Enter the size of the cloudlet (taken from scientific application -Montage) and the time needed to process that cloudlet.

ii. Execute the cloudlet on each virtual machine for obtaining the time of execution.

iii. Compare the execution time of cloudlets on different virtual machines and allot the cloudlet to the VM with minimum execution time. This VM is taken as the best fit for that particular cloudlet.

iv. If the best fit VM is not found for the execution for a particular cloudlet, then split the whole cloudlet into two equal parts on the basis of its size and then again repeat the step iii.

vi. In case after splitting of cloudlet into the new cloudlets, if any of the cloudlet does not get best fit VM, then a request for new VM is generated.

vii. Compare the size of the new VM with the host machine. If size of new VM is less than the host machine then new VM is created and allocated to the cloudlet otherwise cloudlet is considered as failed.

III.1. Distributed algorithm

The problem above can be solved by highly efficient approximation algorithms, e.g., branch-and-bound, and fast off-the-shelf solvers, e.g., CPLEX, for moderate-sized input. An additional challenge arises in dealing with the MapReduce job for big data. In such a job, there are hundreds or even thousands of keys, each of which is associated with a set of variables (e.g., xpij and ypk) and constraints in our formulation, leading to a large-scale optimization problem that is hardly handled by existing algorithms and solvers in practice. In this section, we develop a distributed algorithm to solve the problem on multiple machines in a parallel manner. Our basic idea is to decompose the original large-scale problem into several distributively solvable subproblems that are coordinated by a high-level master problem.

III.2. Online algorithm

Until now, we take the data size and data aggregation ratio as input of our algorithms. In order to get their values, we need to wait all mappers to finish before starting reduce tasks, or conduct estimation via profiling on a small set of data. In practice, map and reduce tasks may partially overlap in execution to increase system throughput, and it is difficult to estimate system parameters at a high accuracy for big data applications. These motivate us to design an online algorithm to dynamically adjust data partition and aggregation during the execution of map and reduce tasks. In this section, we divide the execution of a Map Reduce job into several time slots with a length of several minutes or an hour.

IV. Results

In proposed work when cloudlets arrive to the processor then the best fit approach will be applied. The best fit approach here refers to the process of allocation of suitable VM to the cloudlet which will execute it in minimum time. The simulation results are shown below:

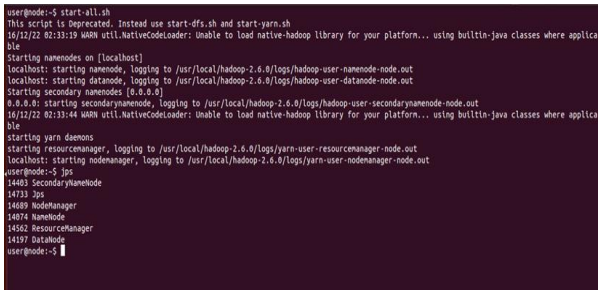

Fig.3 Hadoop Desktop window

Fig.3shows hadoop desktop window. In this window write the code for the proposed research work. In this specified the data node, node manager, Jps, resource manager etc.

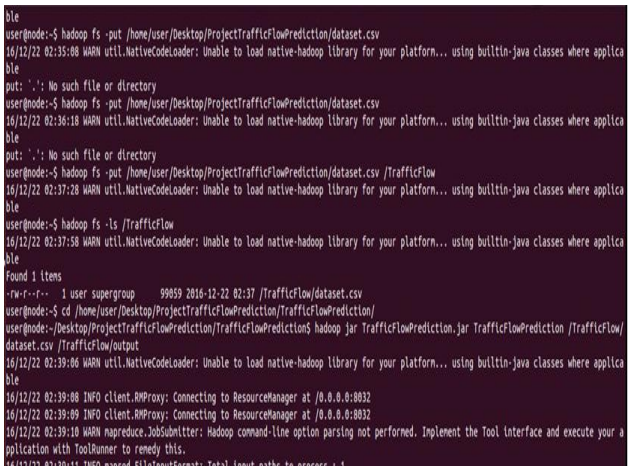

Fig.4 Create Directory

Fig.4 shows create directory for process of remove replication files. In this research work create directory in hadoop distributed filling system at given path for remove replication files. The directory is specified the path for giving the data and obtain the data.

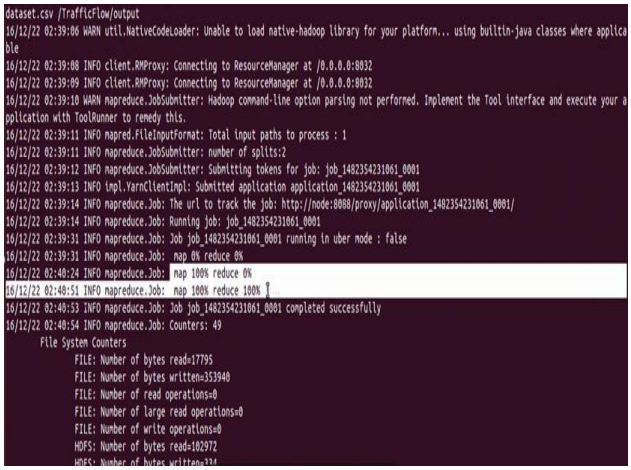

Fig.5 Mapreduce shows in Command window

Fig.5 shows mapreduce. In this command window shows different commands for map reduce. MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

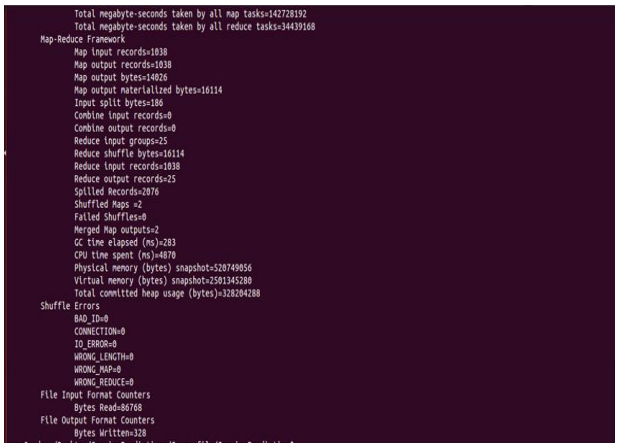

Fig.6 Mapreduce framework

Fig.6 shows Mapreduce framework. In this figure shows total megabyte seconds taken by all map tasks and total megabyte seconds taken by all reduce. MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity in a reliable manner.

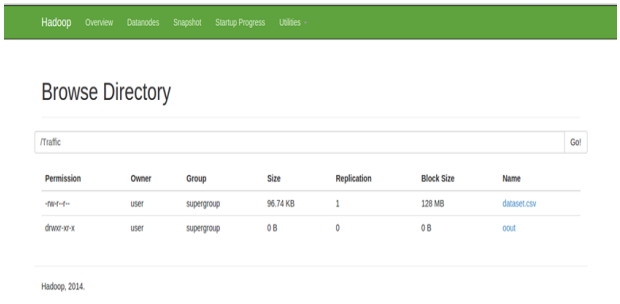

Fig.7 shows the traffic data

Fig.7 shows the traffic data. In this shows block size, file name, group name and no. of replication file. In this show two files browse.

Fig.8 Line graph of traffic flow

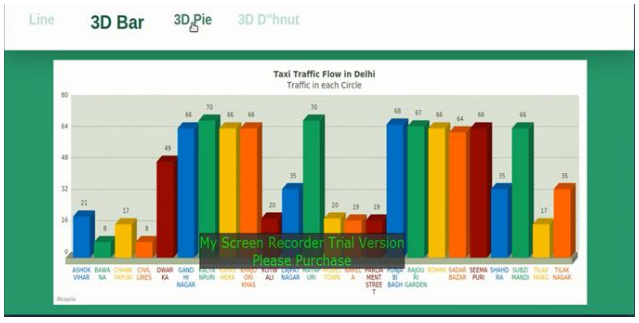

Fig.9 3D Bar graph of traffic flow

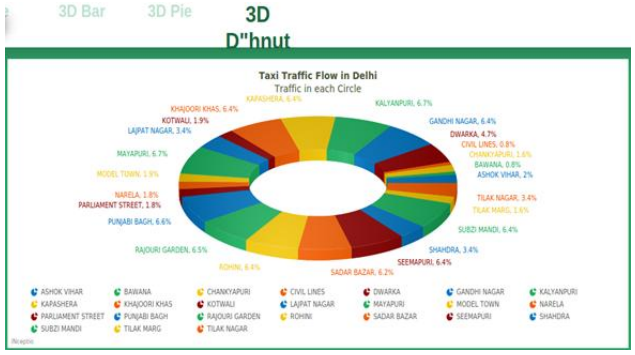

Fig.10 3D D" hunt graph of traffic flow

Fig.8 shows Line graph of traffic flow. In this figure x axis show circles and y axis show number of taxies. In this figure circles shows the traffic. There is different traffic in different places shows in this figure.

Fig.9 shows 3D Bar graph of traffic flow. In this figure x axis show circles and y axis show number of taxies. In this figure traffic shows in 3D bar chart in different places.

Fig.10 shows 3D D''hunt graph of traffic flow. In this figure traffic shows in percentage in different places. This is pie chart.

V. Conclusion

The main goal of this analysis work was to propose an economical proactive fault tolerance technique for scientific applications in Cloud. The planned technique satisfies all the analysis gaps found within the literature survey of fault tolerance techniques. The planned technique handles the fault of allocation of virtual machines proactively specified all cloudlets complete their tasks on or before point and there's successful allocation of virtual machines to all or any the cloudlets. Likelihood of failure within the planned system is less because the fault occurred within the system has been tolerated proactively. The projected work assures the reliability and the efficiency of the system where all the virtual machines are dedicated to their task and no VM has been left out.

References

[1] Cao, Jiajun, et al. "Checkpointing as a service in heterogeneous cloud environments." Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. IEEE, 2015.

[2] Liu, Yang, and Wei Wei. "A replication-based mechanism for fault tolerance in mapreduce framework." Mathematical Problems in Engineering 2015 (2015).

[3] Di, Sheng, et al. "Optimization of cloud task processing with checkpoint-restart mechanism." High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for. IEEE, 2013.

[4] Egwutuoha, Ifeanyi P., et al. "A fault tolerance framework for high performance computing in cloud." Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012.

[5] Egwutuoha, Ifeanyi P., et al. "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems." The Journal of Supercomputing 65.3 (2013): 1302-1326.

[6] Liu, Jing, et al. "An intelligent job scheduling system for web service in cloud computing." Indonesian Journal of Electrical Engineering and Computer Science 11.6 (2013): 2956-2961.

[7] J. Ansel, K. Arya, and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," in 23rd IEEE International Symposium on Parallel and Distributed Processing (IPDPS-09). IEEE, 2009, pp. 1–12.

[8] J. Cao, G. Kerr, K. Arya, and G. Cooperman, "Transparent checkpointrestart over InfiniBand," in ACM Symposium on High Performance Parallel and and Distributed Computing (HPDC'14). ACM Press, 2009.

[9] D. Borthakur, The Hadoop Distributed File System: Architecture and Design, http://hadoop.apache.org/docs/r0.18.0/hdfs design.pdf.

[10] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 29–43, 2003.

[11] B. Selic, "Fault tolerance techniques for distributed systems," http://zh.scribd.com/doc/37243421/Fault-Tolerance-chniquesfor-Distributed-Systems.

[12] F. Wang, J.Qiu, J.Yang, B. Dong,X. Li, andY. Li, "Hadoop high availability through metadata replication," in Proceedings of the 1st International Workshop on Cloud Data Management (CloudDB '09), pp. 37–44,ACM,November 2009.