ISSN:0975-9646

Efficient Algorithm for Fault Tolerance in Cloud Computing

[1]Jasbir Kaur, [2]Supriya Kinger

Department of Computer Science and Engineering,
SGGSWU, Fatehgarh Sahib, India,
Punjab (140406)

Abstract-Fault tolerance in cloud computing platforms and applications is a crucial issue. This issue is especially difficult since cloud computing relies by nature on a complex splitting into many layers. This paper analyses the implementation of fault tolerance in such a complex cloud computing environment with a focus on FCFS and SJF along with MPIL method with fault tolerance property. The proposed algorithm works for reactive fault tolerance among the servers and reallocating the faulty servers task to the new server which has minimum load at the instant of the fault. We illustrate this discussion with experiments where exclusive and collaborative fault tolerance solutions are implemented in an autonomic cloud infrastructure that we prototyped. It also includes algorithm comparison between MPI and MPIL.

Keywords-Fault tolerance, FCFS, SJF, MPI, MPIL

1. INTRODUCTION

Cloud computing is an emerging model for business computing. The basic principles of cloud computing are that data entered by the user is not stored locally, but is stored in data center of internet. The need of services to the lowest level is in demand. Nowadays everybody is not ready to purchase the devices that provide the services. The users rather purchase the services provided by the devices at the big servers. The infrastructure of pay-per-use is highly in demand. The users from different locations just like to have the services and pay for the time being they are availing the services. Cloud computing enables convenient and on-demand network access to shared pool of computing resources that needs to be managed. The companies that can provide cloud computing services manage and maintain the normal operation of these data centers, ensure strong computing power and large storage space for users, then users only at any time, and any place use any terminal equipment that is connected to the internet to access these services without having to think of the position of cloud that are stored. It is a large scale computing using virtual resources. Its popularity is increasing as a cost effective alternative and also High Performance Computing for supercomputers. There have been different clouds releases until now Eucalyptus, Hadoop, CloudSim and Nimbus etc [1].

Resource scheduling is the basic and key process for clouds in Infrastructure as a Service (IaaS) as the need of the request processing is must in the cloud. Every server has limited resources so jobs/requests needs to be scheduled. Each application in the cloud computing is designed as a business processes including a set of abstract processes. To allocate the resources to the tasks there is need to schedule the resources as well as tasks coming to the resources, there needs to be a Service Level Agreements (SLAs) for Quality of Service (QoS). Till now no algorithm has been introduced which considers both reliability and availability together? According to the paradigm of cloud there has been a lot of task scheduling algorithms, some are being fetched on the basics of scheduling done on the operating system. The basics of operating system job scheduling is taken and applied to the resources being installed in the cloud environment [2].

There is enormous need for the cloud services to schedule the resources as this scheduling will further be followed by the job/task scheduling inside of the resources. There may be many instances of the single resource that they can be run at the same time. There is need of checking of availability and reliability and also the load must be balanced among the resources of the same type. For the above parameters there need for a procedure or function that could check them and allocation should be done in the best and optimal way.

2. RELATED WORK

There has been lot of work done in the field of fault tolerance but we will only take the techniques that have come into action a few years back i.e. recent ones

Various fault tolerance techniques are currently prevalent in clouds [3-7]:-

Check pointing – It is an efficient task level fault tolerance technique for long running and big applications. In this scenario after doing every change in system a check pointing is done. When a task fails, rather than from the beginning it is allowed to be restarted that job from the recently checked pointed state.

Job Migration – Sometimes it happens that due to some reason a job cannot be completely executed on a particular machine. At the time of failure of any task, task can be migrated to another machine. Using HA-Proxy job migration can be implemented.

Replication - Replication means copy. Various tasks are replicated and they are run on different resources, for the successful execution and for getting the desired result. Using tools like HA-Proxy, Hadoop and AmazonEc2 replication can be implemented.

Self- Healing - A big task can divided into parts. These multiplications done for better performance. When various instances of an application are running on various virtual instances.

Safety-bag checks - In this case the blocking of commands is done which are not meeting the safety properties [4].
S-Guard- It is less turbulent to normal stream processing. S-Guard is based on rollback recovery. S-Guard can be implemented in HADOOP, Amazon EC2.
Retry- In this case we implement a task again and gain. It is the simplest technique that retries the failed task on the same resource.
Task Resubmission- A job may fail now whenever a failed task is detected, In this case at runtime the task is resubmitted either to the same or to a different resource for execution.
Timing check: This is done by watch dog. This is a supervision technique with time of critical function [4].
Rescue workflow- This technique allows the workflow to persist until it becomes unimaginable to move forward without catering the failed task.
Software Rejuvenation-It is a technique that designs the system for periodic reboots. It restarts the system with clean state and helps to fresh start.
Preemptive Migration- Preemptive Migration count on a feedback-loop control mechanism.
The application is constantly monitored and analyzed.
Masking: After employment of error recovery the new state needs to be identified as a transformed state. Now if this process applied systematically even in the absence of effective error provide the user error masking [5].
Reconfiguration: In this procedure we eliminate the faulty component from the system.
Resource Co-allocation: This is the process of allocating resources for further execution of task.
User specific (defined) exception handling- In this case user defines the particular treatment for a task on its failure.
Several models are implemented based on these types of techniques..
“**AFTRC**” a fault tolerance model for real time cloud computing based on the fact that a real time system can take advantage the computing capacity, and scalable virtualized environment of cloud computing for better implement of real time application. In this proposed model the system tolerates the fault proactively and makes the diction on the basis of reliability of the processing nodes [8].
“**LLFT**” is a propose model which contains a low latency fault tolerance (LLFT) middleware for providing fault tolerance for distributed applications deployed with in the cloud computing environment as a service offered by the owners of the cloud. This model is based on the fact that one of the main challenges of cloud computing is to ensure that the application which are running on the cloud without a hiatus in the service they provided to the user. This middleware replicates application by the using of semi-active replication or semi-passive replication process to protect the application against various types of faults [9].
“**FTWS**” is a proposed model which contains a fault tolerant work flow scheduling algorithm for providing fault tolerance by using replication and resubmission of tasks based on the priority of the tasks in a heuristic matric. This model is based on the fact that work flow is a set of tasks processed in some order based on data and control dependency. Scheduling the workflow included with the task failure consideration in a cloud environment is very challenging. FTWS replicates and schedule the tasks to meet the deadline [10].
“**FTM**” is a proposed model to overcome the limitation of existing methodologies of the on-demand service. To achieve the reliability and resilience they propose an innovative perspective on creating and managing fault tolerance .By this particular methodology user can specify and apply the desire level of fault tolerance without requiring any knowledge about its implementation. FTM architecture this can primarily be viewed as an assemblage of several web services components, each with a specific functionality [11].
”**Candy**” is a component base availability modeling frame work, which constructs a comprehensive availability model semi automatically from system specification describe by systems modeling language. This model is based on the fact that high availability assurance of cloud service is one of the main characteristic of cloud service and also one of the main critical and challenging issues for cloud service provider [12].
“**Vega-warden**” is a uniform user management system which supplies a global user space for different virtual infrastructure and application services in cloud computing environment. This model is constructed for virtual cluster base cloud computing environment to overcome the 2 problems: usability and security arise from sharing of infrastructure [13].

3. PROPOSED WORK

After study, we found that the main fault tolerance issues in cloud computing are detection and recovery. To combat with these problems, many fault tolerance techniques have been designed to reduce the faults. But due to virtualization and internet based service providing behavior fault tolerance in cloud computing is still a big challenge. Our proposed model is not only to tolerate faults but also to reduce the chance of future faults.

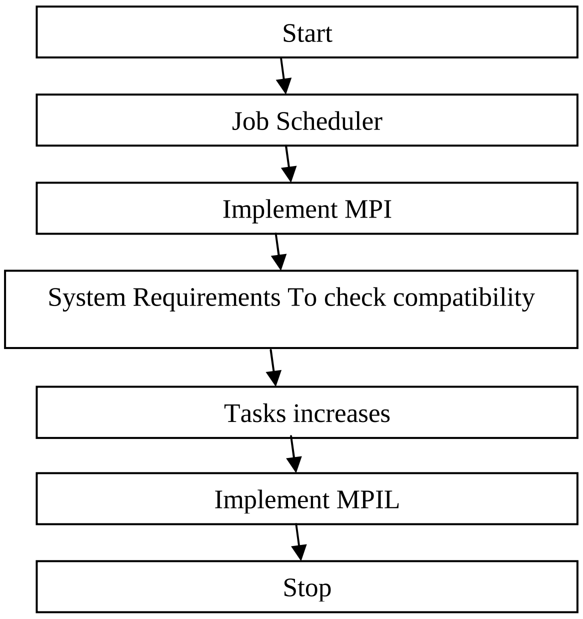

Figure.1 Flowchart of work

The flow chart (fig. 1) shows that implementation of new proposed technique. To obtain the result following steps has to be achieved:

1. There are N number of tasks and J is the Job scheduler. The use of the job scheduler is to execute the jobs.
2. When a scheduler executes the jobs it sends the MPI in the form of message to check the compatibility of the system so that , execution of the jobs takes place.
   Following are the system requirements that are checked to check the compatibility of the system.
   - CPU Type
   - Speed(MHz/GHz)
   - Processor Bus (MHz)
   - Cache (KB)
   - RAM size (MB/GB)
   - SCSI Controller
   - Disk type
   - Disk size (GB)
3. There are number of systems that are not compatible according to the above requirements, so various jobs remain unexecuted.
   E.g there are 10 tasks that are unexecuted due to the incompatibility of the system, so system will check MPI many times to check the reason of un execution , so it is very time consuming process to check all the time.
4. So MPI with look up tables is used in this case, Look up table is the table that contains the response of the system each time in accordance with the tasks. This table is updated number of times s that it can be known that which are the new target systems for job execution. The main difference between the MPI and MPIL is that MPIL is checked only for one time for unexecution of the job while MPI is checked number of time to check the execution of tasks. There is one more term called checkpoints, it is used to reduce the checking of the execution tasks offenly.

Proposed Algorithm (MPI with Lookup)

```
1) START
2) INITIALIZE Job_cnt=1;
3) Configure.System.Model=true
4) Look_up_value=0;
5) Count=selected.job.count
6) If count>1
7) Allocate.job.specified.system
8) Allocation.parameter=configured.parameter
9) If allocation.process==ok
10) Process and update .value.system selection
11) If value.exceeds.system.capabilites--broadcast system.list
12) Update.loopuptable ;
13) if broadcast.count==1
14) Create.newlookuptable=true
15) Else update.lookup=true
16) End.
```

4. TOOLS and RESULTS

Here Visual Studio (2008) is used as a front end and SQL-Server (2005) is used as a back end means for the database purpose.

Tools Used	
Platform	Windows
Operating System	Windows -7
Framework	.NET Framework
Front End Tool	ASP.net with C#
Editing Tool	Microsoft Visual Studio 2008

Table.1 Tools Used

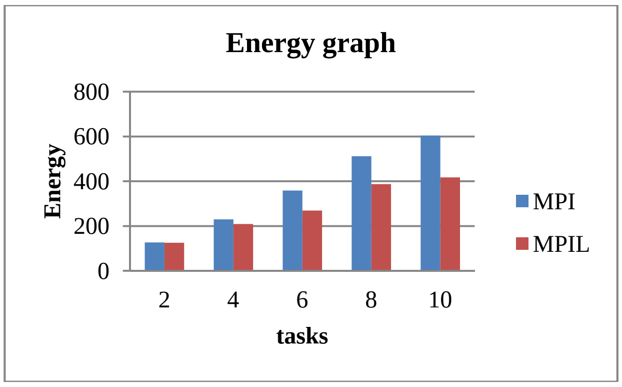

Figure.2 Energy consumed versus task assigned of MPI and MPIL

In fig. 2 energy graph of MPI and MPIL is drawn for the given no. of tasks. This result shows that how much energy is consumed for the particular number of tasks using MPI and MPIL both. It shows that MPIL consume less energy as compare to MPI.

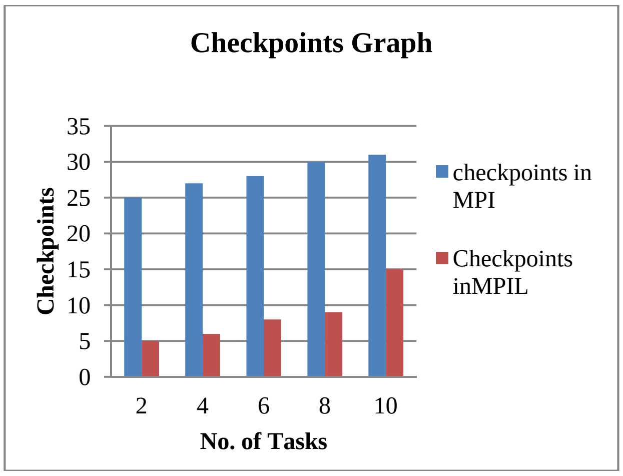

Figure.3 Checkpoints Performance of MPI and MPIL

In fig.3 Checkpoints graph of MPI and MPIL is drawn for the given no. of tasks. This result shows that how many numbers of checkpoints are used for the particular number of tasks using MPI and MPIL. It shows that MPIL uses less number of checkpoints as compare to MPI.

From above it is concluded that MPIL is far better than the MPI because it consumes less energy and lesser number of checkpoints as compare to MPI.

4. Conclusion And Future Scope

Fault tolerance refers to correct and continuous operation even in the presence of faulty components. In most of the real time cloud applications, processing on computing nodes is done remotely. So there are more chances of errors. So there is an increased requirement for fault tolerance to achieve reliability for the real time computing on cloud infrastructure. Fault tolerance is carried out by error processing which have two constituent phases. The phases are "effective error processing" which aimed at bringing the effective error back to a dormant state, i.e. before the occurrence of error and "latent error processing" aimed at ensuring that the error does not become effective again. In the end it is concluded that the performance of MPIL is better than the MPI in terms of both energy consumption and checkpoints required as seen from the above results.

References

[1] Sun Microsystems, Inc. "Introduction to Cloud Computing Architecture" White Paper 1st Edition, June 2009

[2] Mladen A. Vouk, "Cloud Computing – Issues, Research and Implementations", Department of Computer Science, North Carolina State University, Raleigh, North Carolina, USA,Journal of Computing and Information Technology - CIT 16, 2008, 4, 235–246doi:10.2498 /cit.1001391

[3] AnjuBala, InderveerChana," Fault Tolerance- Challenges, Techniques and Implementation in Cloud Science Issues, Vol. 9, Issue 1, No 1, January 2012 ISSN (Online): 1694-0814 www.IJCSI.org

[4] Benjamin Lussier, Alexandre Lampe, Raja Chatila, JérémieGuiochet, Félix Ingrand, Marc-Olivier Killijian, David Powell, "Fault Tolerance in Autonomous Systems: How and How Much?" LAAS-CNRS 7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 04, France {firstname.lastname}@laas.fr

[5] Jean-clandeLaprie "Dependable computing and fault tolerance: concepts and terminology" LAAS-CNRS 7 Avenue du Colonel Roche, 31400 Toulouse, France

[6] GolamMoktaderNayeem , Mohammad Jahangir Alam," Analysis of Different Software Fault Tolerance Techniques", 2006.

[7] GeoffroyVallee, KulathepCharoenpornwattana, Christian Engelmann, AnandTikotekar, Stephen L. Scott," A Framework for Proactive Fault Tolerance".

[8] Sheheryar MalikandFabriceHuet "Adaptive Fault Tolerance in Real Time Cloud Computing" 2011 IEEE World Congress on Service

[9] Wenbing Zhao, P.M. Melliar and L.E. Mose" Fault Tolerance Middleware for Cloud Computing" 2010 IEEE 3rd International Conference on Cloud Computing.

[10] Jayadivya S K, JayaNirmala S, Mary SairaBhanus"Fault Tolerance Workflow Scheduling Based on Replication and Resubmission of Tasks in Cloud Computing" International Journal on Computer Science and Engineering (IJCSE)

[11] Ravi Jhawar, Vincenzo Piuri and Marco Santambrogio"A Comprehensive Conceptual System level Approach to Fault Tolerance in Cloud Computing" IEEE

[12] Fumio Machida, Ermeson Andrade, Dong SeongKim and Kishor S. Trivedi"Candy: Component-based Availability Modeling Framework for Cloud Service Management Using Sys-ML" 2011 30th IEEE International Symposium on Reliable Distributed Systems.

[13] Jianlin, Xiaoyi Lu, Lin Yu, YongqiangZou and Li Zha"Vega Warden: A Uniform User Management System for Cloud Applications "2010 Fifth IEEE International Conference on Networking, Architecture, and Storage.