ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 235 – 238 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

Using Checkpoint For Implementing Fault-Tolerant On Desktop Grids

M.Udaya Prakash Reddy[1]
Assistant Professor[1]
Department Of Computer Science and Engineering ,
Malla Reddy Institute Of Engineering and Technology
email- udhayreddi@gmail.com

P.Mangala Tulasi[2]
Assistant Professor[2]
Department Of Computer Science and Engineering ,
Malla Reddy Institute Of Engineering and Technology
email- tulasipuppala@gmail.com

Abstract:

Grid computing uses massive power of idle cycles of PC's .Desktop grids is nothing using the idle cycles of desktop PC's for computing large scale applications. There are many fields which requires large scale massive power such as scientific fields to handle complex and demanding problems. Desktop grids uses widely distributed grids the probability of occurring fault is more. Desktop Grids are being increasingly used as the execution platform for a variety of applications that can be structured as Bag-of-Tasks (BoT)[1].
In this paper we proposes a fault tolerant fair scheduler for bag of tasks application on desktop grid, which ensures error free transmission of data and performing tasks by using resources and also fair sharing of resources. Fault tolerant is implemented on desktop grid by using check point.

Keywords - Scheduler, Grid Information System, checkpoint

INTRODUCTION

Almost every college, office and every member have computers. In fact modern world is incomplete without Computers. Desktop Grids are computational grids formed by using resources of idle desktop machines. Most of the computers in offices and personal computers are used only for certain time and are idle for most of the time and also they are not using the whole storage of the system .Grid computing combines all the machines that are idle and form as Virtual Group and uses the group for computing large-scale applications. There are two types of desktop grids one is local and other is individual computers . Local computers are the group of computers in an organization and educational institutes . Second is the individual computers which are used by citizens . This offers the opportunity to resolve the increasing need for computational resources. As most desktop systems are idle for significant periods of time, it should be possible to harvest their idle CPU cycles or other unused resources and apply them towards projects in need of such resources. Apart from providing huge computational power and storage capacity desktop grids also have several challenges in using this volatile and shared platform effectively. The usefulness of desktop grid computing is not limited to major high throughput public computing projects. Many institutions, ranging from academics to enterprises, hold vast number of desktop machines and could benefit from exploiting the idle cycles of their local machines[2] . Important examples of desktop grids are SETI@home and PrimeGrid, Almere Grid, Condor based grids[4], the WISDOM project is using grid computing to speed the search for a cure for malaria, a disease that effects millions of people all over the developed world, MammoGrid is building a grid for hospitals to share and analyse mammograms in an effort to improve breast cancer treatment.

Desktop grids faces many challenges and the most important challenges are platform is volatile, since users may reclaim their computer at any time, which makes centralized schedulers inappropriate. Second, desktop grids are likely to be shared among several users, thus we must be particularly careful to ensure a fair sharing of the resources. Fair sharing of resources means there should be balanced share of resource no resource should get more load and no resource should get less instead there should be balanced sharing to the resources. The solution to the fair sharing of resources is already give by fair decentralized scheduler[1]. The main aim of our model is provide fault tolerant by using check points. Fault tolerance is an important property in Grid computing as the dependability of individual Grid resources may not be able to be guaranteed; also as resources are used outside of organizational boundaries, it becomes increasingly difficult to guarantee that a resource being used is not malicious in some way[3].

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 235 – 238 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

PROBLEM DESCRIPTION

In this section, we formally define the problem we target. Our goal is to design a fault-tolerant for fair scheduler on desktop grid. Our main objective while scheduling tasks of concurrent applications is to ensure fairness among users. There is more chance of occurring of resources failures in grid computing because grids are in distributed environment. Desktop Grids is forming Virtual Group by using many computers around world and using computer power and storage so at any time their is a chance of reclaiming the computer at any time from the virtual Group.In the desktop grids it is more because even and individual computer also comes under this category.

ARCHITECTURE

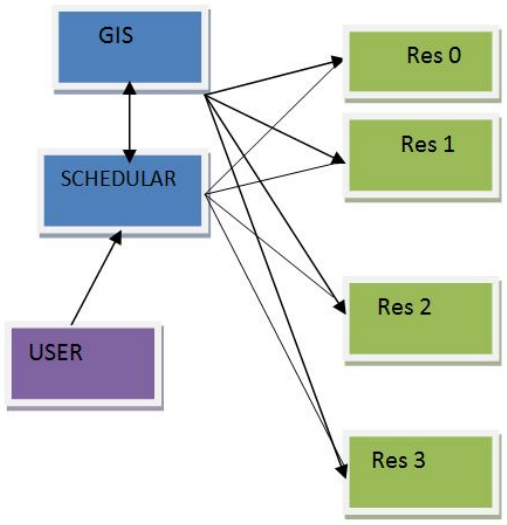

Figure 1:Architecture of Scheduler and GIS

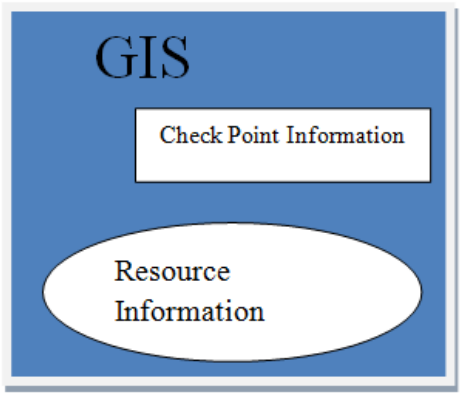

Figure 2: GIS architecture

Algorithm 1 Resource Failure Detection Algorithm used by GIS.

---

repeat
send the messages to all checkpoints up to resource from GIS
if a checkpoint does not respond sending acknowledge back **then**
remove it from the list
update GIS entities about the failure
end if
wait for Time GIS seconds
until simulation is over

Algorithm 2 Resource Failure Detection Algorithm used by Scheduler .

---

repeat
push the messages to all resources which are running jobs
if a resource does not respond by sending acknowledge **then**
Scheduler ask the GIS for a list of resources
choose one of them
resubmit the jobs
end if
wait for T seconds
until all my jobs have been successfully executed.

GIS: maintains an up-to-date list of available resources. GIS collects the information from all the checkpoints of all the resources and maintains an up-to-date list.

According to Algorithm 1 :
Message will be send to all the checkpoints to all the resources and if message is not send back from the then remove the resource from the resource information maintained in the GIS. Contact a GIS

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 235 – 238 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

entity for a list of available resources in order to know where to run their jobs. The functionality of this entity can be summarized in Algorithm 2. For enabling an efficient pushing mechanism, User Datagram Protocol (UDP) is used by these entities. This is due to the fact that UDP requires a less significant network latency in comparison with a Transmission Control Protocol (TCP), although UDP does not provide retransmission of lost packets. The sequential steps are shown in a box with a number inside. Here is list of steps showing the working of the model .First, *R1,R2,R3* and *R4 resources* register to *GIS* (step 1). Then, *GIS* creates a list of available resources. In order to keep that list up-to-date, *GIS* push the messages to the resources periodically (step 2). When *User* wants to run a job, he/she contacts scheduler and scheduler contacts *GIS* in order to get a list of available resources (step 3). Upon receiving the scheduler's request, *GIS* returns its list. In that moment, scheduler will choose *R1* for example, based on the features of the resource and the job scheduling. When *User* has chosen the resource, he/she submits the job to *R1* and starts a regular pushing mechanism.

In the event of a failure affecting *R1*, *GIS* is able to detect this problem due to the pushing mechanism in place (step 4). Hence, *GIS* removes the failed resource from the list. During a routine push, GIS discovers that *R1* has failed. As a result, *scheduler* ask *GIS* for a list of resources (step 5). When *R1* recovers, it registers itself again to *GIS* (step 6). With this approach, *GIS* is able to maintain an up-to-date list of available resources. If the failure only affects some of the machines in a resource, what happens next depends on the allocation policy of this resource. If the resource runs a *space-shared* allocation policy, the jobs that are currently running on the failed machines will be terminated and sent back to users. However, when the resource runs a *time-shared* (round-robin) allocation policy, no jobs will be failed, as their execution will continue in the remaining machines of the resource. For both allocation policies, the remaining machines are responsible for responding to pushing requests from users and GIS. Moreover, they are required to inform the GIS about such failure. This way, the GIS can have accurate information on the current status of the resource.

USING GridSim FOR RESOURCE FAILURES

GridSim:

GridSim allows modeling and simulation of entities in parallel and distributed computing systems such as users, applications, resources, and resource brokers/schedulers for design and evaluation of scheduling algorithms.

Overview of GridSim functionalities:

Incorporates failures of Grid resources during runtime. New allocation policy can be made and integrated into the GridSim Toolkit, by extending from Alloc class. Has the infrastructure or framework to support advance reservation of a grid system. Incorporates a functionality that reads workload traces taken from supercomputers for simulating a realistic grid environment. Incorporates an auction model into GridSim. Incorporates a datagrid extension into GridSim. Incorporates a network extension into GridSim. Now, resources and other entities can be linked in a network topology. Incorporates a background network traffic functionality based on a probabilistic distribution. This is useful for simulating over a public network where the network is congested. Incorporates multiple regional GridInformationService(GIS) entities connected in a network topology. Hence, you can simulate an experiment with multiple Virtual Organizations (VOs).Adds ant build file to compile GridSim source files[10].

Along with GridSim classes we are using many new classes for implementing resource failures.

UserFailure: as its name suggests, this class implements the behavior of the users of our grid environment. Its functionality can be summarized as follows: (1)creation of jobs; (2) submission of jobs to resources; (3)push the resources used to run its jobs; (4) on the failure of a job, choose another resource and re-submit the failed job to it; (5) receive successful jobs.

ResourceFailure: based on Grid-Sim's GridResource class, this class interacts with RegionalGISWithFailure to set machines as failed or working. It also interacts with classes implementing AllocPolicyWithFailure to set jobs as failed.

AllocWithFailure: it is an interface class, which provides some functions to deal with resource failures. Each allocation policy implementing this interface will have a different behaviour with regard to the failures.

AvailabilityInfo: This class is used to implement the pushing mechanism. The user and GIS send objects of this class to resources, which in turn send them back, as mentioned previously. When a resource still has some working machines left, it will send these objects back with no delay. However, when all machines are out of order, the resource sends these objects back with some delay with a

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 235 – 238 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

special tag. This is done to simulate a situation, where if a resource does not reply to the given push before a specified time out, then it is interpreted as not available. This method is used to overcome the same problem in GridSim, i.e. waiting for events that never arrive.

Resource failure statistics

Virtual Group	Resources	Failed Resources
V1	100	5
V2	20	1
V3	60	30
V4	23	23

Job failures

	Jobs	Failed Jobs
1	200	52
2	89	8
3	100	65
4	10	10

CONCLUSION AND FUTURE WORK

Grid computing is the hot topic now-a-days which is used in many fields such as scientific, global warming etc. GridSim is the simulation tool used for simulation. By using Gridsim we model failure detection and handling the failures by using checkpoints. Failure model is implemented by using Gridsim, by this model researchers can be able to develop more realistic Grid models.In this paper we handled exception handling by using Grid sim.

As for future work, we are planning to use the improved simulation tool to carry out research aimed at providing more security and QoS in Grids.

References

[1]: Javier Celaya, Loris Marchal ” A Fair Decentralized Scheduler for Bag-of-tasks Applications on Desktop Grids"

[2]: Derrick Kondo and Filipe Araujo"Characterizing Result Errors in Internet Desktop Grids".

[3]: Domingues, P.Sch. of Technol. & Manage., Polytech. Inst. of Leiria, Portugal.Silva, J.G. ; Silva, L.Sharing Checkpoints to Improve Turnaround Time in Desktop Grid Computing

[4]Universiteit Antwerpen

[5] Paul Townend and Jie Xu"Fault Tolerance within a Grid Environment"Department of Computer ScienceUniversity of Durham, DH1 3LE, United Kingdomp.m.townend@dur.ac.uk jie.xu@dur.ac.uk

[6] Agust´_n Caminero and Anthony Sulistio "Extending GridSim with an Architecture for Failure Detection" Department of Computing Systems 2Dept. of Computer Sc. & Software Eng.
The University of Castilla La Mancha, Spain The University of Melbourne, Australia
fagustin, blanca, carmeng@dsi.uclm.es fanthony, rajg@csse.unimelb.edu.au 2007

[7] N. Hayashibara, A. Cherif, and T. Katayama. "Failure detectors for large-scale distributed systems" In *Proc. of the 21th Symp. on Reliable Distributed Systems, (SRDS)*, Japan, 2002.

[8] Nikolaos D. Doulamis"Fair Scheduling Algorithms in Grids"IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, NOV 2007

[9] Fangpeng Dong and Selim G. Akl "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems"January 2006.

[10] http://www.cloudbus.org/gridsim/