Decentralized fault tolerant model for P2P Grid

Mohammed REBBAH
University of Mascara, Computer Science Department, Algeria
rebba_med@yahoo.fr

Yahya SLIMANI
University of El Manar, Tunis
yahya.slimani@fst.rnu.tn

Abdelkader BENYETTOU
Laboratoire SIMPA
Université des Sciences et Technologie d’Oran – Mohammed BOUDIAF
BP 1505 Mnaouer Oran - ALGERIE
a_benyettou@yahoo.fr

Abstract: Coupling P2P models and grid computing made the birth of a new model of grid called P2P-Grid. This model, as it inherits its advantages from these two components, it inherits their disadvantages and others related to the coupling. The volatility of its resources has weakened its quality of service. Our contribution in this paper is the proposition of a fault tolerant model for P2P Grid. It proposes a new vision on the mechanisms of fault tolerance in this type of grid, taking into account all the characteristics of the P2P grid especially the dynamicity and the scalability. The model proposed allows transforming a grid in interconnected groups, where each group is composed of SuperNode, SuperNodeDuplic and peers. We implemented a service of fault tolerance FTPGRID under the middleware Globus GT4.

Keywords: Grid computing, P2P systems, P2P grid, fault tolerance, Group Peer, Globus

INTRODUCTION

Recent years have seen the emergence of large size structures "grid computing and P2P systems" making cooperate a large number of nodes. These environments, aggregating the resources of thousands of computers and applications offer new opportunities in terms of computing power and storage capacity. Given the scalability of grid computing today, the need for self-organization and dynamic reconfiguration is becoming increasingly important. In this context, the convergence of computing between grid computing and P2P systems seems natural. However, a grid infrastructure generally consists of a federation of hierarchical clusters connected by LAN and SAN and interconnected by WANs to broadband. Currently, the standardization and the implementation of Grid systems are strongly structured in centralized or hierarchical architectures. However, the use of centralized approaches compromises both scalability and reliability of the Virtual Organization [1], especially in terms of robustness and fault tolerance, reducing the effective exploitation of the resources and their availability issues. In contrast, P2P systems are usually deployed on the Internet, a network topology usually flat. This difference can bring to wonder about the adequacy of the P2P communication mechanisms on a grid computing. A major challenge is to cope with the volatility of nodes and variances important interconnection network. It is thus very difficult to implement a fault tolerant service in a P2P grid. At this scale, errors are not exceptions but part of normal system behavior. Some studies [2-7] have explored the advantages of implementing Grid systems adopting Peer-To-Peer (P2P) models and techniques to manage the services even if critical concerns related to the decentralized control and the security issues remain. The design of the P2P grid systems, must take into account these errors to provide a stable service. Mechanisms for fault tolerance are inherently complex and expensive; they must be validated and evaluated. We propose, in this paper, a fault tolerance model for P2P grid based on the structuring the P2P grid into a set of interconnected groups and each group consists of a SuperNode (SN), SuperNodeDuplic (SND) and peers; our contribution is the reduction de redundancy in the peer group by a passive replication in the SND and we have developed FTPGRID a fault tolerance service over Globus Toolkit [2]. FTPGRID has the following properties: hybrid dynamic, it supports heterogeneity, intermittent connection peers and scalability.

The rest of the paper is organized as follows: In Section 2, we present a state of the art of the fault tolerance techniques for P2P grids. Section 3 defines our proposed model and its various actors. Fault tolerance part of this model is discussed in Section 4. Section 5 presents the architecture of FTPGRID grid service developed over Globus GT4 to validate our proposed model, and some experimental results. Conclusion and future works are presented in Section 6.

2. Related Work

Grid and P2P computing platforms aim to distribute and share computing resources for deploying large scale applications. They have the potential to deliver high performance for many applications and many users. Typically, a grid consists of an heterogeneous collection of computational resources. The characteristics of each resource may vary greatly, including processor speed, processor load, disk space, hardware configurations, variance in network bandwidths, latencies, etc. Several realizations have already been proposed, they include the Seti@Home project [8], Globus [9] and XtremWeb [10]. P2P research focuses more and more on providing
infrastructure and diversifying the set of applications: The P2P model could help to ensure Grid scalability; designers can use P2P philosophy and techniques to implement non-hierarchical decentralized Grid systems. Making use of P2P scalability and dynamic, designing resources location and discovery protocol for querying Grid resources to improve query success rate and fault tolerance of system have been already involved into network systems merging P2P and Grid [11-13]. These hybrid systems can efficiently and reasonably implement resources utilization, which provide a new platform for sharing digital resources. Moreover, since the P2P-Grid systems are composed by peers that are resident and stable service providers, they do not suffer the major drawback of structured systems, i.e. the high maintenance cost in the presence of high churn (in a typical P2P system there is a big percentage of intermittent users that join, depart and rejoin the system in a totally unpredictable way) in order to maintain their rigid structure.

The main approaches proposed to resolve these issues are based on: (i) space filling curves (SFC [14,15]), (ii) tree based structure [16-19] and (iii) variant of SHA-1/2 hashing [20-22]. All these solutions present some drawbacks: SFC suffers from curse of space since when the number of dimensions increases, locality becomes worse. Moreover, the mapping based on SFC fails to uniformly distribute the load among the peers, when the data distribution is skewed. This last issue is overcome with the tree-based approach, which is able to distribute the load among every node in the structure, even if the root peer represents a single point of failure and load imbalance.

In [23], D. Talia et al. emphasize that Grid can benefit from the more flexible connectivity models used in P2P networks. And they explain possibility of the commonalities and synergies between two technologies in terms of the connectivity, access services, resource discovery, and fault tolerance. Some authors in [24] talk about super peer networks, and it assumes that the degree of redundancy of super peers is 2, since if system allows the increment of k-redundant super peers, then the number of open connection among super peers increases exponentially as k. In [25], Y.H. Moon et al. give a k-redundancy scheme to reduce the load of the Super-peer, but gives no information on the k-optimal for the proper functioning of the groups. In [26], Y.H. Moon et al. propose how to decide optimized redundancy level of group peers by using system cost function and grid local reliability. Moreover we discuss an effectiveness of SLA-constrained load scheduling policy with multi-probing technology in order to maintain group more stable.

All these works focuses on the load balancing of the super peer, but does not provide a mechanism for fault tolerance for this type of P2P structures. Increasing numbers of redundancy reduces the load but it endangers the groups (the risk of partitioning, security, data consistency). Despite their relatively small number, group members may be spread throughout the Internet and must be able to deal with arbitrary partitions due to network failures, congestion, and hostile attacks. We propose a group peer composed of a SN with a SND which is a passive replication (1.5 redundancy), to reduce the redundancy and the Group peer is composed from a cluster.

3. Proposed model

In our model, the grid computing is a finite set of clusters interconnected by a WAN connection; the nodes of each cluster have a LAN connection in P2P structure (see Fig. 1). The proposed grid model consists of a set of groups interconnected by a WAN link. To form a group of acceptable size, each cluster forms a group; each peer can freely join or leave a group under the observation of a SN. Users can submit jobs from any peer of a group. Once a job is submitted, the owner peer takes in charge its execution in the grid until its completion. The proposed model is supported by a transparent fault tolerance technique providing the service although the occurrence of faults at a peer.

3.1. Model actors

Our architecture is based on the following actors (see Fig. 2):

1- **Group**: is a set of peers connected by a LAN connection belongs to the same cluster, each group consists of a SuperNode (SN), SuperNodeDuplicate (SND) and peers.

2- **Repository**: is a database allowing in particular keeping
the information about peers state, jobs and lists of jobs (the update of the database is done periodically).

3-SuperNodes (SN) is a manager of a group, it contains all information concerning its peers.
1. If a peer wants to start a job remotely, it asks the SN to obtain the list of peers available to start his job.
2. The SN monitors peers, if a peer fails "Crash", it detects the fault and submits its jobs to the free peers.
3. The SN also detects the fault of SND, if the fault is "Crash," the SN selects another peer to be a new SND.
4. The SN is responsible to tolerate faults from other groups when it is impossible to tolerate them locally.

4-Peer: Upon receipt of a job, it can run it locally or it can submit it to another peer of the group. Each peer is considered an owner or an even recipient.

5-SuperNodeDuplicate (SND): is a peer group, in addition, it is responsible for monitoring the SN, if SN fails, it takes over in becoming the SN of the group and it designates a new SND.

3.2. Rules of management
1. When a Peer receives a job, it can rather execute it locally or submit it to another peer of the group, it consults the SN to send the available peers and it selects an even recipient to throw the job from afar by direct exchange and it receives the results in the same way.
2. The SN is a manager fixed beforehand, it doesn't execute any job.
3. The jobs queue of the peers has a limited size.
4. The duplication of the repository is passive.
5. The user can duplicate his jobs in the group.

4. Fault tolerance
To ensure greater reliability, the faults must be tolerated in a transparent manner and without increasing the load of SN. We proposed to tolerate crash and disconnection faults by a fault tolerance mechanism based on passive replication, which is responsible to tolerate peer faults in the peer group or outside the group if it fails to submit the group

4.1 Faults type
Our system is able to detect and recover crash faults and disconnection faults.

Crash fault: in this case, the entity stops abruptly and it can not deliver any service (stop process), so it will no longer be accessible. It may be a lack of node.

Disconnection Fault: In this type of fault, there is a lack of communication medium. This type of fault occurs when there is an error in communication management between the different nodes of the grid such as fault in the DNS Manager, a connection problem (wiring) or problem in system files.

4.2. Fault tolerance at the SN level

4.2.1. Peer crash fault

a- Tolerance inside the group (Toler_In_Group): When a SN detects a fault crash of any peer, it tolerates the fault by attempting to distribute the related jobs on the others peers of its group according to the availability of peers and their ability to support other jobs (i.e. the queue of jobs is full) (see Fig. 3).

b- Tolerance outside of the group (Toler_Out_Group): In case of the saturation of the peers of the group, the SN tolerates peers fault outside the group by sending a request to neighboring groups SN’s to send them more jobs. The SN starts by attempting to tolerate the fault inside the group by launching Toler_In_Group and if there are still jobs to tolerate, it tries with the neighboring groups by launching Toler_Out_Group (see Fig. 4).
4.2.2 Peer disconnection fault

When a peer is disconnected from its SN, it tries to re-establish the connection with the SN through another peer group; it requests the peers group to allow it to communicate with SN through one of them and once a peer answers positively, it opens a communication channel toward the SN.

4.2.3 SND crash fault

When SN detects a SND fault crash, it tolerates its jobs as a peer of a group and designates among the group peers a new SND.

4.2.4. SN crash fault

When SND detects a SN fault crash; it becomes the SN of the group, it updates all the group peers, then designates one of them as a New SND.

4.3. Algorithms

1. Detect_Fault: At the SN, we use two data sets of tables type: Peers_inf that contains all the information of the peers and table_jobs that contains all the information of the jobs to run.

```plaintext
BEGIN
1. While (True) do
2.   Peers_Infs.first()
3.  While (Not Peers_Infs.eof()) do
4.    IF (Ping (Peers_Infs.@IP)=false)) then
5.       AddressIP=Peers_infs.@IP
6.       Host_name=Peers_Infs.Peer
7.       nb_Peer=Get_Peers();
8.       For i=1 to nb_Peer then
9.         Send(Msg_verify_Stat_Peer(AdressIP),Peer[i])
10.     EndFor
11.    State_Peer=Receive(Get_stat_Peer())
12.   IF (State_Peer="Connected") then
```

Fig. 3: Peer crash fault tolerance inside the group

Fig. 4: Peer crash fault tolerance outside the group
13. elif Peers_Infs.Fault_Type="Disconnection"
14.     Else Peers_Infs.Fault_Type="Crash"
15.     EndIF
16.     X=Select SND From Peers_Infs where Peer=Hote_name
17.     IF (X.SND=True) then
18.         Listener=Get Listener();
19.         Toler_Fault SND(Hote_name,Fault_Type,Listener)
20.     Else
21.         Toler_Fault_Peer(Hote_name,Fault_Type,Listener)
22.     EndIF
23.     EndIF
25.     EndDO
26.     EndWhile

END.

2. Toler_Fault_Peer: This algorithm tolerates the peer faults depending on the type of fault (crash or disconnection).

If this is a fault crash:
1-Toler_In_Group: if the size is greater than or equal to the number of jobs to tolerate, we tolerate all the jobs locally.
2-Toler_Out_Group: if the size available within the group is insufficient, we tolerate out of the group.

Toler_Fault_Peer(Hote_name,Fault_Type, Listener)
BEGIN
1. IF(Fault_Type="Disconnection") then
2.     Determine Path(Listener)
3. Else //Fault_Type="Crash"
4.     List_Job_Tolerer=[]
5.     Jobs_Peer_Crash=Select(Idjob,Prop,Job)From Table_Jobs where (Peer=Hote_name)
6. For i=1 to (Jobs_Peer_crash).lenght do
7.     Verify_DuplicNb_Duplic (Idjob, prop)
8.     IF(nb_Duplic=0) then
9.         ADD (Idjob,Prop,Job,List_Job_tolere)
10.     EndFor
11.     EndIF
12. Nb_Job=List_Job_Tolerer.lenght
13. Toler_Fault(List_Job_Tolerer)
END.

3. Toler_Fault : This algorithm distributes the jobs of a failed peer on these peers within the group (if there are peers available), or outside the group through the SN if the group is overload.

Toler_Fault(List_Job_Tolerer)
BEGIN
1. Nb_Job=List_Job_Tolerer.lenght
2. Nb_Job_Notolerer=List_Job_Tolerer.lenght
3. While (nb_Job_Tolerer!=0) do
4.     Nb_Max=Select SUM (Taille_Disp) from Peers_Infs where stat_peer="Connected"
5.     IF (Nb_Max >= Nb_Job) then
6.         Toler_Local_Group (List_JobNotolerer)// Local fault toleranceTolerer
7.     Else
8.         List_S=List_Job_Notolerer
10.        Toler_In_Group (List_S)
11.        Toler_Out_Group (List_C)
12.     EndIF
END.

5. Experimentation

We present a new hybrid approach of fault tolerance in a P2P grid to illustrate the adaptability of the hybrid method. Our approach is implemented on Globus GT4.0.1 middleware [9] based on the algorithms of fault tolerance introduced in section 4. The architecture of GT4 is based on a set of components and tools that makes the design and deployment of grid services efficient, robust and secure.

5.1. Grid service Architecture

We delivered our model by developing a service Grid (non web service), which was deployed in the middleware Globus GT4 and was called FTPGRID: Fault Tolerance
P2P GRID". It is composed of two interfaces: one is responsible of the management of peers and the other is designed for SN (see Fig. 5).

**Peer interface:** Through this interface, the user can launch a job locally or through a file already prepared by inputting parameters of starting a job in the grid. A peer can start a job at another peer group with the opportunity of submitting a number of replicas and the status of jobs run in the peer (see Fig. 6).

**SN interface:** We can find all the information from the peer group (peer address, type of peer, IP address, state of pair, type of fault, number of jobs, maximum size of the queue and available size) (see Fig. 7) in the window job monitoring, we can find all information about each job status (peer-id, type, peer IP, type of faults, the peer tolerating the job, size of the queue, the queue space) (see Fig. 8) and Fig. 9 archives all information from pairs of jobs (even, job-id, job status, peer owner, job tolerate, time of submission, execution time).

---

**Fig. 5:** FTPGRID service grid architecture

**Fig. 6:** Peer interface
5.2. Results

We have tested our fault tolerance service (FTPGRID) to evaluate the performance of our application. Our experiment was conducted on the average size of the queue of each peer is equal to 5 has increased the number of jobs in the grid from 5 to 50, were conducted 6 tests for each experiment. The number of messages transmitted between the different actors in the model are presented in Fig. 10. The messages of SN is slightly superior than SND and peers, the SND is slightly superior than every peer of the group, the difference comes of the updates with the SN. More the number of peers in the group become important, more the SN manages to tolerate the maximum of faults
inside the group but the load of the group increases (see Fig. 11). It is difficult to make a compromise between the size of the group and the level of tolerance. The reduction of the size of the group increases tolerances out group which is in general expensive (due to the WAN connection) than the annification of the size of the groups to preserve tolerance inside it, but this last overloads the group which may damage the performances and can even generate some other faults. The overload of the group is generally related to the number of jobs started in the group directly, that affects the level of tolerance (ingroup or outgroup). If the number of jobs launched in the group increases the rate of tolerance outside also increases, which requires a good load balancing in the group to maximize the tolerance in the group (see Fig. 12).

6. Conclusion

In this paper, we proposed a fault tolerance model for a P2P grid based on a set of interconnected groups, each group is composed of SN, SND and peers. This model offers a new vision on the mechanisms of fault tolerance, taking into account all the characteristics of the particular P2P Grid like dynamicity and scalability. We proposed a passive replication in the SND, which reduces the k-redundancy to 1.5. We have implemented a service fault tolerance FTPGRID over Globus GT4, which can manage in a transparent fault tolerance in the P2P grid. We have treated
the crash and disconnection faults. As a solution, we implemented a technique for fault tolerance based on replication (for jobs submitted into the grid) at the SN and SND. The integration of technical fault tolerance in such a P2P grid causes an enormous load in the group with the duplication at the SN and SND by it makes the service provided by the model grid more reliable. For a continuation of our work, several perspectives can be figured: Take into account other types of faults or to form dynamic groups in order to optimize the fault tolerance outside groups.

References


[8] SETI@home http://setiathome.ssl.berkeley.edu


