ISSN 2278-7763

IMPLEMENTATION OF AUTONOMIC COMPUTING IN GRID ENVIRONMENT

Banalata Sarangi[1], Dr Laxman Sahoo[2]

[1]Research Associate, Department of Computer Science, KIIT University, Bhubaneswar, Odisha, INDIA
[2]Professor, Department of Computer Science, KIIT University, Bhubaneswar, Odisha, INDIA
Email: [1] banalata1sarangi@gmail.com, [2]laxmansahoo@hotmail.com

ABSTRACT

The advances in computing and communication technologies and software have resulted in an explosive growth in computing systems and applications that impact all aspects of our life. Computing systems are expected to be effective and serve useful purpose when they are first introduced and continue to be useful as condition changes. With increase in complexity of systems and applications, their development, configuration, and management challenges are beyond the capabilities of existing tools and methodologies. So the system becomes unmanageable and insecure. So in order to make the systems self-manageable and secure the concept of Autonomic computing is evolved. Autonomic computing offers a potential solution to these challenging research problems. The basic application area of autonomic computing is grid computing. Both autonomic computing and grid computing are proposed as innovations of IT. Autonomic computing aims to present a solution to the rapidly increasing complexity crises in IT industry, as grid computing tries to share and integrate distributed computational resources and data resources. Basic aim is to implement the autonomic computing in grid related study like autonomic task distribution and handling in grids, and autonomic resource allocation

Keywords : ***Autonomic computing, Grid computing, Real time transaction***

1 INTRODUCTION

The future Grid will be an autonomic environment that can not only assist users to share large-scale resources and accomplish collaborative tasks but also self-manage to reduce the users' interventions as much as possible. In such an autonomic Grid environment, the real-time transaction processing is a key and challenging technology to protect systems from various failures. This survey paper presents an autonomic real-time transaction service (ARTTS) that can [1] dynamically discover Grid services as participants to execute specified sub-transactions, [2] coordinate these participants to achieve the real-time and transactional requirements, and [3] assign priorities to schedule concurrent transactions. By handling the potential failures and exceptions autonomically, the ARTTS can facilitate the implementation of real-time Grid transactions and simplify the system management work, which frees users from the complex interference in the autonomic Grid environment.

2. RELATED WORK

2.1Service discovery

The first step of handling a Grid transaction is to dynamically discover services to execute sub-transactions. The Universal Description, Discovery, and integration (UDDI) define how to publish and discover Web services. Providers of Web services directly publish their services in the UDDI server. Service discovery is an important work in grid transaction, which helps in executing, sub transaction. Service here is of two types

- Transient

ISSN 2278-7763

- Persistent

The former refers to the services whose instances are created and/or destroyed in runtime and live only for a specified period. Therefore, it is impractical for the UDDI server to manage both creation and registration of millions of remote transient Grid services. This paper employs two-level registry mechanism to adapt to those transient services. Service descriptions are registered in the Undeserved while its local registry performs creation of a transient service instance.

2.2 Transaction processing

Transaction processing has three kinds of roles. Application program, Transaction manager, Resource manager. And two interface XA and TX. Real time scheduler schedule real time transaction using priority assignment policy and resolve data conflict by lock mechanism. Main issue is how to propagate deadlines from global transaction to its sub transaction and how to control concurrent execution of transaction.

2.3 Autonomic Grid computing

Optimal Grid is an autonomic Grid infrastructure developed in IBM. Using Optimal Grid; the problem owner has no need to concern the partition and deployment of the problem and to know the enlisting of computing nodes. The delivery of the code for various parts of the distributed computing, the run time management of the overall problem and dynamic rebalancing are done automatically. Transaction processing is the effective approach to recovering systems from potential failures. The autonomic Grid must be able to handle exceptions and failures in execution of reliable applications. Benefiting from above efforts, the ARTTS provides the self-protection ability through automating real-time Grid transaction processing to prevent the system from system-wide failures and maintain system consistency and real-time property without intervention of users.

3. REAL-TIME GRID TRANSACTION

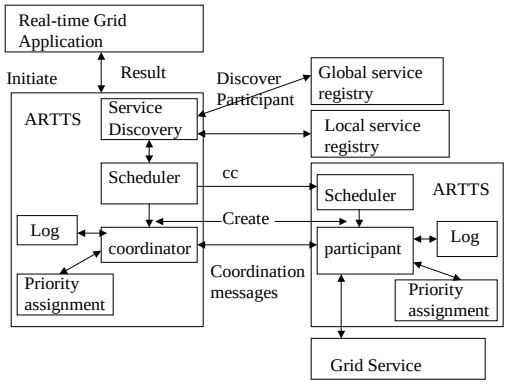
Tang et al. (2003a) have discussed coordination of different activities in Grid computing and presented corresponding coordination algorithms for two types of transactions, atomic transaction (AT) and cohesion transaction (CT). The AT, served as coordinating the short-lived transaction, consists of a set of atomic sub transactions that have to commit synchronously. The CT, consisting of atomic sub-transactions or cohesion sub-transactions, allows some sub-transactions to commit while others fail in order to coordinate the long-lived transaction. The real-time Grid transaction is an extension of the above work in the autonomic Grid environment.

Transaction processing for the autonomic Grid environment focuses on how to coordinate sub-transactions rather than processing of each individual sub-transaction. The real-time Grid transaction mainly concerns with participant discovery, coordination algorithms, and policies of deadline and priority assignment.

3.1 Flow of the real-time Grid transaction processing

In the autonomic Grid environment, a typical real-time transaction processing includes following steps, as shown

1. The initial ARTTS initiates a global transaction for a Grid application, discovers and selects satisfactory Grid services to serve as participants, using the Service Discovery module
2. Its Scheduler creates a Coordinator and broadcasts the Coordination Context (CC) messages to all selected remote participants, which create Participant locally and return Response messages to the Coordinator.
3. The created Coordinator and Participants interact to control the transaction execution, including correct completion and failure recovery.



The flow the real time Grid transaction processing

3.2 Discovery of participants

In the Grid service environment, any network entity is encapsulated into a service, which is identified by the Grid service handle(s) and reference(s). The goal of discovering participants is to dynamically find the references of service instances. A service discovery model with

ISSN 2278-7763

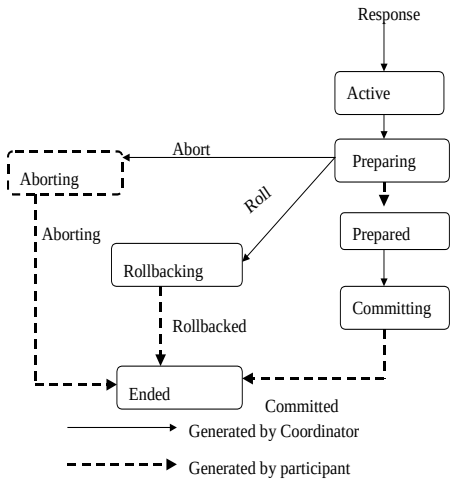
a two-level registry has been proposed in Tang et al. (2003b). Handles and references of persistent Grid service instances (or handles of factory for transient Grid services) are registered with the Local Service Registry, which publishes service descriptions in the Global Service Registry. The basic steps of service discovery can be described as follows.

- Query the Global Service Registry to obtain the service description and the handle of their home Local Service Registry.
- Select desirable services based on some policies such as the lowest cost.
- Discover the references of the selected services in the Local Service Registry. For a transient Grid service, its factory service creates the service instances and returns the initial service references.

3.3 Coordination of participants

The real-time Grid transaction aims at time-critical Situation, where the most important goal is to maximize the number of transactions that can finish before their deadlines rather than the system throughput. To improve the successful rate, a few functional alternative services are organized into a functional alternative service group (FASG) to execute the same sub-transaction in parallel. If one member of a FASG can successfully complete before its deadline and reports a Committable message, the FASG is considered committable and other members are aborted. In the preparation phase, each functional alternative service executes a specified sub-transaction in its private work area (PWA). When a service completes the sub transaction successfully before its deadline, it returns a Committable message. Any actual commit occurs only when the global transaction commits. On receipt of an Abort message, the service rollbacks operations taken previously by releasing the PWA. In the commit phase, the Commit message enables the committable participants, which have reported the Committable messages, to commit sub-transactions.

The bellow figure illustrates the state conversion diagram of the Real-time Grid transaction. Solid rectangles indicate the states of both Coordinator and Participants while the dashed rectangle only denotes the state of Participants.



The state conversion diagram of the real-time Grid transaction

Note that the transaction enters the prepared state only after the Coordinator receives a Committable message from each FASG before deadline d (T). Otherwise, the Coordinator sends Rollback messages to all participants.

4. DEADLINE CALCULATION

Deadline refers to the time by which the transaction must finish or else undesirable results may occur.

Tasks are of two types:

Local task - The task that are executed only at the originating node.

Global task - it consist of series of sub transaction.

The objective is to determine the priorities of the sub transaction so that the percentage of missed deadline is kept as low as possible.

Dead line of local transaction:

$$d(T) = ri + sti + wei$$

d (T)=deadline

r i= arrival time

Sti = slack

Wei = Execution time

Deadline of sub transaction:

A global task is in the form of T= [T1, T2…Tm]

There are basically 4 methods

1. UD-Ultimate deadline

ISSN 2278-7763

$$dl(Ti) = dl(T)$$

2. ED-Effective Deadline

$$dl(Ti) = dl(T) - \sum_{j=j+1}^{m} Pex(Tj)$$

3. EQS-Equal slack

$$dl(Ti) = ar(Ti) + [dl(T) - ar(Ti) - \sum_{j=i}^{m} pex(Tj)]/(m - i + 1)$$

4. EQF-Equal Flexibility first

$$dl(Ti) = ar(Ti) + pex(Ti) + [(dl(T) - ar(Ti) - \sum_{j=i}^{m} pex(Tj))] * [pex(Ti) / \sum_{j=i}^{m} pex(Tj)]$$

Let X is a transaction

Ar (X)—arrival time

d l(X)---deadline

sl(X)—slack

ex (X)—real execution time

Pex (X)—Predicted execution time

dl (Ti)—deadline of sub transaction

Earliest deadline first

Highest priority is given to the transaction with closest deadline

5. SIMULATION

Let a model contain k nodes, each node services their tasks according to some real-time scheduling algorithms. Example-EDF.

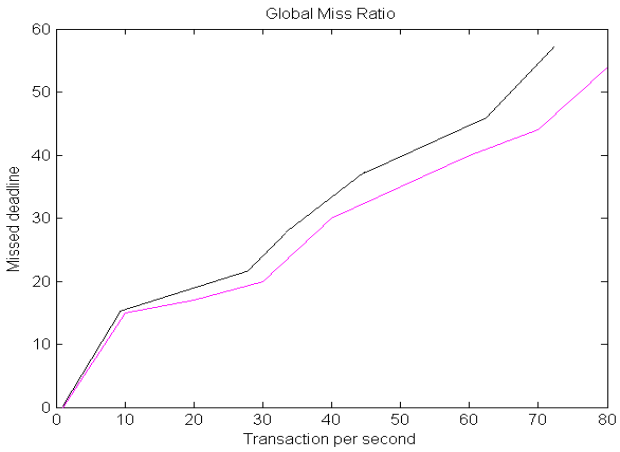
The transaction manager generates local and global transactions with an independent Poisson stream with an arrival rate varying from 1 to 80 transactions per second. Evaluation time, deadline of transaction follows a uniform distribution. No of transaction resources access is atleast one main memory database is taken for simplicity. First step of simulation consisted in measuring the global miss ratio for different scheduling algorithm, let EDF and FCFS.GMR is defined as The total number of transaction that miss their deadlines compared to the total number of transactions accepted by the algorithm.

System Parameter

Parameter	Description	Value
We_i	Execution time	30 to 300 ms
sf_i	Slack factor: $st_i = sf_i * We_i$ $d_i = r_i + We_i\ (1 + sf_i)$	3 to 5
λ	Transaction arrival rate per second	1 to 80 transaction per second
NR	Number of resources	20

6. Results

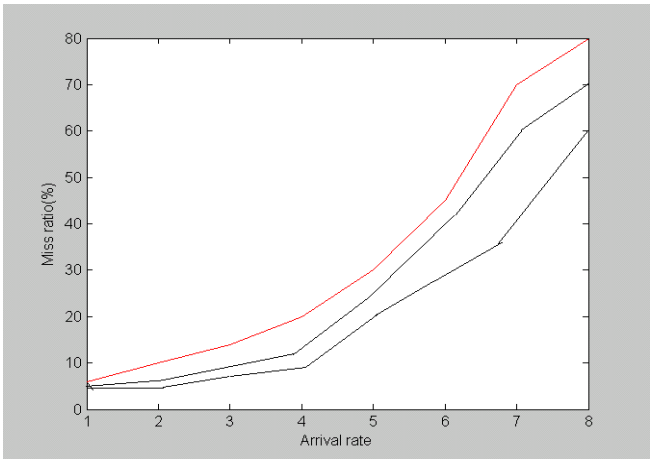
We found that the EDF algorithm have less global miss ratio in compare to FCFS.both the algorithm performs almost identically until number of transaction per second grows to 30.following that point EDF misses fewer deadlines than FCFS.



The reason was each sub transaction acted as a unit to compete for resources so that more workload the more system resource they consumed. So more transaction missed their deadlines, as they could not get enough re-

ISSN 2278-7763

source in time



7. CONCLUSION AND FUTURE WORK

In this paper we basically concentrate on the autonomic systems. We discuss about autonomic real time transaction service. How this service dynamically discover grid services to execute specified sub-transactions. Dynamically assign priorities for scheduling concurrent transaction. As a result, by handling the entire transaction process on behalf of users automatically it facilitate the implementation of real-time and transactional grid application to provide self-protection function and simplify the management work.

The future work will be to make the autonomic application more practical and satisfy all the four properties for solving various issues in the autonomic grid environment. The future work will focus on combining security measure with our work's the overall grid environment will get authentication, authorization and communication protection.

REFERENCES

[1] Feilong Tang, Minglu Li, and Joshua Zhexue Huang. "Real-time transaction processing for au Tonomic Grid Applications,"Engineering Applica tion of ArtificialIntelligence 17(2004), pp.799-807, China, 2004.

[2] Xiaolong Jin, and Jiming Liu, "Characterizing au tonomic task distribution and Handling in grids," Engineering Application of Artificial Intelligence 17(2004), Pp.809-823, Hong Kong, 2004.

[3] RainerUnland, and Huaglory Tianfield, Towardsautonomic computing Systems," Engineering Application of Artificial Intelligence 17(2004), pp.689-699, Germany, 2004.

[4] Eser Kandogan, John Bailey Rob Barrett, and Paul p.Maglio, "Usable autonomic computing Systems: the system administrators' perspec tive," In Advance Engineering Informatics *19*, Pp.213-221, Nov 2005.

[5] Roy Sterritt, and Dave Bustard, "Towards an Auto nomic computing Environment," Proceeding of IEEE14th international workshop on Database and Expert systems applications (DEXA '03), 2003.

[6] Ricardo M.Bastos, Flavio M.de Oliveira, and Jase Palazzo M.de Oliveira, "Autonomic computing approach for resource allocation,"Expert systems with applications 28(2005), pp.9-19, Brazil, 2005.

[7] Andrea Baldini, Alfredo Benso, and Paolo Prinetto. "A dependable autonomic computing environment for self-testing of complex heterogeneous systems,"Electronics notes in Theoretical Computer science 116(2005), pp.45-57, Italy, 2005.

[8] Jeffrey O.Kephart, and David M.chess, "The vi Sion of Autonomic Computing,"IEEE computer Society, Jan 2003, pp.41-49.

[9] Paul Lin, Alexander MacArthur, and John Leaney, "Defining Autonomic Computing: A software engineering prospective,"Proceedings of the 2005 IEEE Australian Software engineering conference (ASWEC '05), 2005.

[10] Steve R.white, James E.Hanson, Ian Whally, David M.chess, and Jeffrey O.Kephart, "An Architectural Approach to Autonomic Computing," Proceedings of IEEE International conference on Autonomic Computing (ICAC'04), 2004.

[11] E.Grishikashvili Pereira, R.Pereira, and A.Talb-Bendiav, "Performance evaluation for self healing Distributed services and fault detection mechanisms," Journals of Computer and System Science, 2006.

[12] Yuan-Shan Dai, "Autonomic Computing and Reliability Improvement," Proceedings of the 8th IEEE International Symposium on Object-Oriented

[13] Shimon Whiteson, and Peterstone, "Towards Autonomic Computing: Adaptive network routing and scheduling," Proceeding of IEEE International Conference Autonomic Computing, 2004.

ISSN 2278-7763