International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Fault Tolerant Techniques in Mobile Grid Computing: A Survey

Amit Savyanavar[1], Pranav Ghate[2]

[1, 2]Department of Computer and engineering, MITCOE, Kothrud, Pune University, Pune, India

Abstract: ***Computational grid systems are the systems, where a task is distributed in various geographically distributed nodes. An important property of the grid system is Fault tolerance; it is a property of the system which enables a system to operate properly in case of failures. Checkpointing and replication are the two commonly used techniques for fault tolerance. Furthermore, these techniques mitigate the lost work, they commonly migrates a snapshot of the latest correct state of the failed node or share information between the resources to ensure consistency between them. However, these techniques are not full proof as they may introduce some kind of runtime overheads. Increasing fault tolerance means making your system more reliable and available to the users. It is also essential to satisfy the QoS requirement of the computational grids. Reliability and availability are the two greatest challenges in grid due to the unreliable nature of its resources.***

Keywords: Fault tolerance, Computational Grids, Checkpointing, mobile grid systems, QoS.

1. Introduction

A system that coordinates resources which are not bound to a centralized control using some general purpose protocols and some significant Quality of Service (QoS) is called a *grid.* They consist of nodes that are geographically distributed to execute a specific task. The probability of failure increases with increase of grid components or load. This may give rise to process failure, node failures, machine crashes, link failure etc. Together giving rise to failure to complete the job in given deadline, denial of services violating service level agreements. Consequently, expected QoS degrades. [1]

Grid systems are widely used in the areas like academics, industry etc. A new kind of grids called *Mobile* Grids is receiving growing heed and is becoming an important part in computational grids. It consists of mobile nodes which provide computing resources and facilitate user access to the grid [2]. Similarly mobile grids can aid in utilization of any unutilized resources on the device. On the contrary they provide a chance for mobile nodes to save their own resources and use the resources of the grid.

While executing a task consider a failure occurs, that is a component fails, then previous work will be lost and the application has to be restarted and continue from scratch resulting in time and resource wastage. We need a technique which can prevent or detect these faults. The property of the grid which enables the system to continue operating smoothly while the failure has occurred is termed as *Fault Tolerance.* A complementary but a different approach to increase the dependability of the system is *Fault prevention.* This property includes techniques like inspecting the nodes or the resources and eliminates the cause by which a failure or a fault may occur.

When do we say a failure has occurred? It is the time where the system behaves differently than the specified behaviour. The seed of occurrence of a failure is called an error, and a fault is a root cause of a failure. Nevertheless a fault may not certainly generate an error, regardless the same fault may evolve in multiple failures. Likewise a single error may cause multiple failures.

The paper is organised as follows, Section 2 briefly describes some of the recent fault tolerance technique. Section 3 describes some of the findings and Section 4 presents the conclusion.

2. Fault Tolerance Methodologies

As we know fault tolerance is an important property of mobile grid systems, as faults which may occur through dynamic links, less reliable links etc may cause fatal node failure intern failure of the whole system. However some techniques or methods are used to overcome these faults. Some of them are listed below.

2.1 Reliability Arrangements

Paul J darbyIII and nian-Feng Tzeng have proposed a reliability middleware driven checkpointing arrangement [1] where the nodes act as providers and consumers. An arrangement is stable when a consumer or provider does not prefer any other provider or consumer, respectively, to its current partner in the relationship. The relationship is defined as:

$$MH_i \rightarrow MH_j$$

Where, MH_i is a mobile host working as a provider and MH_j is a mobile host working as a dedicated Consumer. Above relationship is defined as, each provider in the arrangement has a dedicated consumer. If stable arrangement which is formed say, $\prod_i$ holds the relationship $R_i \geq R_{th}$ where R_i is the reliability of some checkpointing arrangement and R_{th} is the reliability of the stable checkpointing arrangement. This arrangement is called $\prod_{ith}$. The reliability is found using. [12]

$$\prod_{c=1}^{n-1}[1-(1-\omega c)\,(1-\rho c \varphi c b)]$$

Where,

$\prod_i$ – is some unique checkpointing arrangement. Which is a directed graph consisting of a unique collection $MH_k \rightarrow MH_l$ relationship on MH_i, where

Licensed Under Creative Commons Attribution CC BY

$k \neq l$.
ω_k – Connectivity of host k, functions as a checkpointing node.
ρ_l – connectivity of host l, is functioning as a task executer.
φ_{cb} – wireless link reliability.

Reliability driven middleware (ReD) are QoS aware as they are provided with the information via wireless measurements. As the distance increases the link reliability decreases, resulting to failures. In the algorithm proposed in this paper each provider gets a request message from a consumer which is in the proximity range of the provider. If the provider already has a dedicated consumer then gain of each relationship is checked. And a pair is formed of those two nodes having highest pairing gain amongst two relationships.
The pairing gain is calculated by: [12]

$$"\partial c \,||\varphi cb \times \rho b" \,/"\partial c \; alone"$$

Where,
∂c - Connectivity of host C, ie, $\partial c = \{\varphi c_1, \varphi c_2, \varphi c_3 \ldots \varphi c_n\}$, 1 … n are the task execution nodes.
φ_{cb} – Wireless link reliability, node C to node B, $\varphi_{cb} = e^{(-\partial i \Delta ti).}$
ρ_b – Connectivity of the node B, is serving as Checkpointing node.
Δt_i – Minimum checkpoint data transmission interval.
$\Delta t_i = \Delta t_{bc} = \Delta t_{cb}$
Δt_{bc} – Minimum checkpoint data transmission time.
Δt_{cb} – Minimum recovery data transmission time.

In [1] Parmeet Kumar and Pritee Parwekar has proposed a fuzzy rule based fault tolerance in mobile grids where the checkpointing arrangement is derived without accessing any wired network or a static host. In this paper weights for the nodes in the system are derived using a Fuzzy rule system (FRS), where the weights are availability of stable storage and battery power. If a node has sufficient storage for saving the checkpoint that particular node qualifies to be a checkpointing storage node (CSN). Furthermore the nodes which are assigned high weights by the FRS are classified as CSNs who will accept the checkpoints of neighboring nodes.

Stable storage and battery level together decide the fitness of fuzzy set. Memory capacity and crisp values in the range of (0%, 100%) are taken as the parameters of stable storage and battery power respectively. Reprehensive values for membership function of stable storage are used from [9], [10] and are assumed to be 1MB to 10MB from storage and checkpoint or recovery message is of size few bytes. The weights are calculated based on Fuzzy rule depicted in below Table 1.

Table 1: Fuzzy Rule Base Table

Rule	Stable Storage	Battery Power	Weight
1	Low	Low	Low
2	Low	Intermediate	Low
3	Low	High	Intermediate
4	Intermediate	Low	Low
5	Intermediate	Intermediate	Intermediate
6	Intermediate	High	High
7	High	Low	Intermediate
8	High	Intermediate	High
9	High	High	High

The proposed protocol constructs a limited size rule base of size n^2 where *n* is number of values used for ruling. In this case those are High, Low and Intermediate. So it will have a rule size of 3^2.

2.2 Checkpointing Algorithms

In [21] an antecedent graph approach for checkpointing is proposed which Antecedent Graphs (AGs) are checkpointed onto a stable storage periodically so that the size of the graph piggybacked on the message does not reduce the efficiency. We see how the graphs are formed considering 3 mobile agents. In the beginning each agent is at the state S_{A0}, S_{B0}, S_{C0} respectively. The intervals are determined by the messages that are received. When agent B receives message *m1* from C, the deterministic interval and AG of state interval S_{B1} provides information about the previous condition.

Ex. Formation of AG of graph for agent A. message *m2* is received by agent A from agent B. A combines the antecedence graph received from B to its own graph for the formation of the event S_{A1}. B sends a message m0 to C. After this, the message m1 is received by B from C; with the difference of antecedence graph of C (if a message from C has been previously received). B combines the AG received from C to its own graph for the formation of the event SB1. C receives a message m0 from B for the formation of the event S_{C1}. After this, the message m1 is sent to B from C. After this, C receives the message m3 from A, with the antecedence graph of A. C combines the antecedence graph received from A to its own graph for the formation of the event S_{C2}.

They have also used a parallel checkpointing algorithm where each checkpoint is given a unique sequence number, which increases monotonically. When a node sends a request it attaches checkpoint request and numeric weight value in parallel the dependent agents (DA) make AGs of events which occur during the checkpointing stage. However all the agents specified in the AG receives the inquiry message from checkpointing agent (CA). If DA agrees then CA sends the weight indicating positive response to starting agent (SA). If the SA receives such responses from all DAs then it checkpoints the info in Base station (BS) and informs other respectively. The maximum length graph for these agents is then constructed and stored in stable storage by BA.

Jack Dongarra, Thomas Herault and Yves Robert in [20] revisited a Double Checkpointing and proposed a Triple checkpointing algorithm. In double checkpointing to avoid crash of the whole application by a single failure local checkpoints are replicated. This algorithm states that once checkpointing is done locally and again the same checkpoint

Licensed Under Creative Commons Attribution CC BY

is replicated remotely. In case of application failures the checkpointed data may be recovers from two points, therefore if the node fails the data can be recovers from local or a paired node increasing the reliability. The non blocking algorithm is given below:

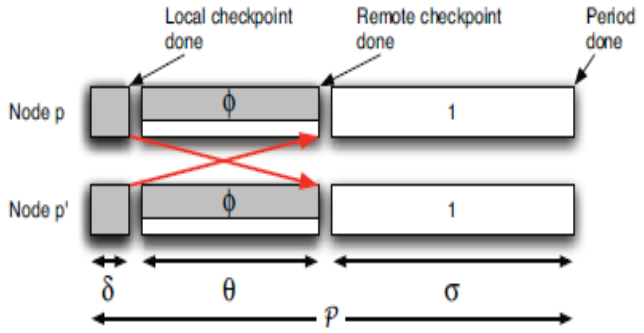

Figure 1: Non blocking Checkpointing Algorithm

- Periodic checkpoints are taken, with period [20]

P=$\delta + \theta + \sigma$.

Where,

δ – Period of checkpointing locally and no application work is performed.

θ – Period of checkpointing remotely. Each node checkpoints remotely, that is it exchanges its checkpointed data with its paired node.

σ– Period of application executing at full speed.

The weight, which is the amount of checkpointing data, is given by [20]

$$W = (\theta - \phi) + \sigma = \mathcal{P} - \delta - \phi$$

Where,

$\emptyset$ - is some communication overhead.

In triple checkpointing algorithm the nodes triples rather than pairs. It is more reliable when a fixed amount of memory is available for checkpointing. When the failure occurs the local checkpoint is lost and the checkpoint has to be recovered from its buddy. Triple checkpointing algorithm works just the same as double checkpointing but here we add another secondary node for checkpointing one more time. This reduces the overhead occurred in double checkpointing. Here 1st node checkpoints on primary node 2 and secondary node 3. Similarly node 2 has 2 buddies' node 1 and node 3 and node 3 has again 2 buddies' node 1 and node 2.

3. Findings

Here we find that in check pointing arrangements the reliability middleware is used to maintain the reliability of the check pointing arrangement where as in other paper fuzzy logic is used to do the same. Reliability middleware makes a stable pairs where as other algorithm makes the arrangement depending on fuzzy rules both are way better then Random Checkpoint arrangement (RCA) algorithm, shown below.

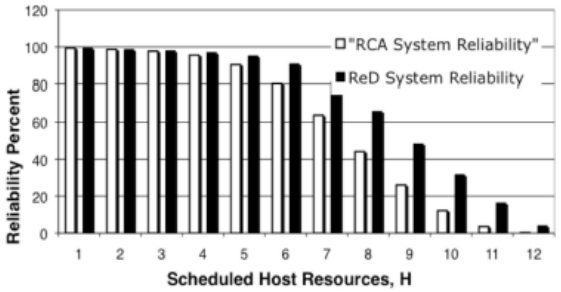

Figure 2: RCA vs. Reliability driven middleware algorithm [1]

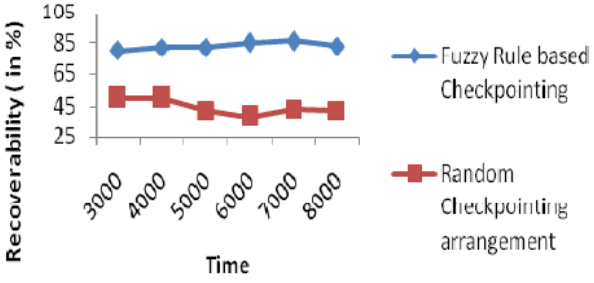

Figure 3: Fuzzy rule VS RCA [7]

In antecedent graph approach a base agent is used which increases the dependability and if the base agent fails the whole system crashes. Where as in double checkpointing there is no base agent and nodes depend on the neighboring buddy. Here the checkpoints are saved on neighbor. However they generate decent communication and other overheads which may create some problems. Whereas triple checkpointing has one more node where the data is checkpointed and reduces these overheads.

4. Conclusion

From above survey we can conclude that ReD middleware and fuzzy algorithms are efficient and reliable then the RCA. Also, the checkpointing algorithms which does not include the base agent are more reliable then the algorithms including the base agent.

5. Future Scope

In future we may merge these techniques and try to make a hybrid algorithm which provides a reliable checkpointing arrangement, which is decentralized and with low communicational overhead.

References

[1] Parmeet Kaur and Pritee Parwekar "Fuzzy Rule based Checkpointing Arrangement for Fault Tolerance in Mobile Grids" 2014 IEEE
[2] H. Kurdi, M. Li, and H. Al-Raweshidy, " A Classification of Emerging and Traditional Grid Systems", IEEE Distributed Systems Online, vol. 9, no. 3, pp.1,1, 2008
[3] S.P. Ahuja, and J.R. Myers, "A Survey on Wireless Grid Computing", J. Supercomputer. Vol 37, 1, pp. 3-21, 2006.
[4] P.J Darby and Nian-Feng Tzeng, "Decentralized QoS-Aware Checkpointing Arrangement in Mobile Grid

Licensed Under Creative Commons Attribution CC BY

Computing", IEEE Transactions on Mobile Computing, vol.9, no.8, pp.1173-1186, 2010.
[5] Z. Wang, Q. Chen and C. Gao, " Implementing Grid Computing over Mobile Ad-Hoc Networks based on Mobile Agent". In Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops, 321-326, 2006.
[6] I. Rao, N. Imran, P. Woo Lee, E.Huh and T. Chung, "A proxy based efficient checkpointing scheme for fault recovery in mobile grid system", In Proc. of the 13th International conf. on High Performance Computing , pp 448-459, 2006.
[7] E. N Elnozahy, L. Alvisi, Y.M. Wang and D.B. Johnson, "A survey of rollback-recovery protocols in message-passing systems", ACM Computing Surveys (CSUR), v.34 n.3, p.375-408, September 2002
[8] R.Prakash and M.Singhal, "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 10, 1035-1048, 1996.
[9] Q. Jiang and D. Manivannan,"An optimistic checkpointing and selective message logging approach for consistent global checkpoint collection in distributed systems", in Proc. IEEE International Parallel and Distributed Processing Symposium, pp. 1-10. 2007
[10] T. Park, N. Woo and H.Y. Yeom, "An efficient optimistic message logging scheme for recoverable mobile computing systems", IEEE Transactions on Mobile Computing, vol.1, no.4, pp.265, 277, 2002.
[11] G. Cao, and M. Singhal," Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems", IEEE Trans. Parallel Distributed Systems 12, 2, 157-172, 2001.
[12] Paul J. Darby III and Nian-Feng Tzeng, "Decentralized QOS aware Checkpointing Arrangements in Mobile Grid Computing", *IEEE transaction on mobile computing, vol. 9. no. 8, Aug. 2010*
[13] W. Woerndl and R. Eigner, "Collaborative, Context-Aware Applications for Inter-Networked Cars," Proc. 16th Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '07), pp. 180-185, June 2007.
[14] W. O'Neil, "The Cooperative Engagement Capability (CEC) Transforming Naval Anti-Air Warfare," Case Studies in National Security Transformation, Center for Technology and National Security Policy, Number 11, Aug. 2007.
[15] P PEO Soldier, "Documentary Highlights Success of Land Warrior System," http://www.army.mil/-news/2009/05/27/21680- documentary-highlights-success-of-land-warrior-system, May 2009.
[16] SUN Microsystems, "Sun Grid Compute Utility," http://www. sun.com/service/sungrid, 2006.
[17] Hewlett-Packard Development Company, L.P., "Grid-Computing —Extending the Boundaries of Distributed IT," http:// h71028.www7.hp.com/ERC/downloads/4AA03675EN W.pdf? Jumpid=reg_R1002_USEN, Jan. 2007.
[18] "IBM Grid Computing," http://www-1.ibm.com/grid/about_ grid/what_is.shtml, Jan. 2007.
[19] S. Wesner et al., "Mobile Collaborative Business Grids—A Short Overview of the Akogrimo Project," white paper, Akogrimo Consortium, 2006.
[20] Computerworld, "HP Promises Global Wireless for Notebook PCs," http://www.computerworld.com/mobiletopics/mobile/ story/0, 10801, 110218, 00.html? Source=NLT_AM&nid=110218, Apr. 2006.
[21] Rajwinder Singh and Mayank Dave "Antecedence Graph Approach to Checkpointing for Fault Tolerance in Mobile Agent Systems" IEEE Transaction on computers Vol. 62, No. 2, February 2013
[22] A. Agbaria and W.H. Sanders, "Distributed Snapshots for Mobile Computing Systems," Proc. IEEE Second Int'l Conf. Pervasive Computing and Comm., pp. 177-188, 2004.
[23] T. Osman, W. Wagealla, and A. Bargiela, "An Approach to Rollback Recovery of Collaborating Mobile Agents," IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Rev., vol. 34, no. 1, pp. 48-57, Feb. 2004.
[24] J. Yang, J. Cao, and W. Wu, "CIC: An Integrated Approach to Checkpointing in Mobile Agent Systems," Proc. IEEE Second Int'l Conf. Semantics, Knowledge and Grid, 2006.
[25] P.S. Mandal and K. Mukhopadhayaya, "Checkpointing Using Mobile Agents in Distributed Systems," Proc. IEEE Int'l Conf. Computing, Theory and Applications, 2007.
[26] Jack Dongarra, Thomas Herault and Yves Robert "Revisiting the double checkpointing algorithm" *IEEE DOI 10.1109/IPDPSW 2013.11, IEEE International Symposium on parallel and Distributed Processing Workshop and PhD Forum 201.*
[27] J. Dongarra, P. Beckman, P. Aerts, F. Cappello, T. Lippert, S. Matsuoka, P. Messina, T. Moore, R. Stevens, A. Trefethen, and M. Valero, "The international exascale software project: a call to cooperative action by the global high-performance community," *Int. J. High Perform. Compute. Appl.*, vol. 23, no. 4, pp. 309–322, 2009.
[28] F. Cappello, A. Geist, B. Group, L. Kale, B. Kramer, and M. Snir, "Toward Exascale Resilience," *Int. J. High Perform. Compute. Appl.*, vol. 23, no. 4, pp. 374–388, 2009.
[29] A. Bouteiller, T. Herault, G. Krawezik, P. Lemarinier, and F. Cappello, "MPICH-V: a multiprotocol fault tolerant MPI," *IJHPCA*, vol. 20, no. 3, pp. 319–333, 2006.
[30] X. Ni, E. Meneses, and L. V. Kal´e, "Hiding checkpoint overhead in HPC applications with a semi-blocking algorithm," in *Proc. 2012 IEEE Int. Conf. Cluster computing*. IEEE Computer Society, 2012.

Licensed Under Creative Commons Attribution CC BY