Dynamic Adaptation of Checkpoints and Rescheduling in Grid Computing

Antony Lidya Therasa.S
Sri Venkateswara college of Engineering, Sriperumbudur

Sumathi.G
Sri Venkateswara college of Engineering, Sriperumbudur

Antony Dalya.S
Aringar Anna Institute of Science and Technology, Sriperumbudur

ABSTRACT

Grid is a form distributed computing mainly to virtualilze and utilize geographically distributed idle resources. A grid is a distributed computational and storage environment often composed of heterogeneous autonomously managed subsystems. As a result varying resource availability becomes common place, often resulting in loss and delay of executing jobs. To ensure good performance fault tolerance should be taken into account. Here we address the fault tolerance in terms of resource failure. Commonly utilized techniques to achieve fault tolerance is periodic checkpointing, which periodically saves the jobs state. But an inappropriate checkpointing interval leads to delay in the job execution, and reduces the throughput. Hence in the proposed work, the strategy used to achieve fault tolerance is by dynamically adapting the checkpoints based on current status and history of failure information of the resource, which is maintained in the Information server. The Last failure time and Mean failure time based algorithm dynamically modifies the frequency of checkpoint interval, hence increases the throughput by reducing the unnecessary checkpoint overhead. In case of resource failure, the proposed Fault Index Based Rescheduling (FIBR) algorithm reschedules the job from the failed resource to some other available resource with the least Fault-index value and executes the job from the last saved checkpoint. This ensures the job to be executed within the deadline with increased throughput and helps in making the grid environment trust worthy.

Keywords: Grid Computing, Fault-Tolerance, Checkpointing.

1. INTRODUCTION

Grid computing or the use of a computational grid, is applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. Compared to other distributed environments, such as clusters, complexity of grid mainly originates from decentralized management and resource heterogeneity. These characteristics often lead to strong variations in availability, which in particular depends on resource and network failure rates, administrative policies, and fluctuations in system load. Apparently, runtime changes in system availability can significantly affect job execution. Since for a large group of time-critical or time consuming jobs delay and loss are not acceptable, fault tolerance should be taken into account. Providing fault tolerance in a distributed environment, while optimizing resource utilization and job execution times, is a challenging task. To accomplish it, we use dynamic adaptation of checkpoints technique, based on current status of the job and failure history of the resource, which overcomes the checkpoint overhead that is caused by unnecessary checkpoints incase of periodic checkpointing. And hence achieves fault tolerance with increased throughput.

Here we consider two dynamic checkpoint adaptation method: Last Failure time based Checkpoint Adaptation (LFCA), Mean Failure time based Checkpoint Adaptation (MFCA), which is mainly to change the checkpoint frequency dynamically, based on the last failure time and mean failure time of each resource. The failure time of each resource is stored in the Information Server, these failure time suggest both stability and probability of failure of each resource based on which checkpoint frequency can be varied accordingly.

In case of resource failure, the proposed Fault Index Based Rescheduling (FIBR) algorithm reschedules the job from failed resource to some other available resource with the least fault index value, and executes the job from the last saved checkpoint.

The rest of this paper is organized as follows: section 2 contains the description about the related work. Section 3 explains about the proposed work. Section 4 concludes the paper.

2. RELATED WORK

Fault tolerance is an important property in grid computing, since the resources are geographically distributed. Moreover the probability of failure is much greater than in traditional parallel systems. Therefore fault tolerance has become a crucial area of interest. A large number of research efforts have already been devoted to fault tolerance. Various aspects that have been explored include design and implementation of fault detection services as well as the development of failure prediction and recovery strategies. The latter are often implemented through job checkpointing in combination with migration and job replication. Although both methods aim to improve system performance in the presence of failure, their effectiveness largely depends on tuning runtime parameters such as the checkpointing interval and the number of replicas. Determining optimal values for these parameters is far from trivial, for it requires good knowledge of the application and the distributed system at hand. The work on Grid fault tolerance can be divided into pro-active and post-active mechanisms. In pro-active mechanisms, the failure consideration for the Grid is made before the scheduling of a job, and dispatched with hopes that the job does not fail. Whereas, Post-active mechanisms handles the job failures after it has occurred. Of those that look into these issues, many works are primarily post-active in nature and deal with failures through [8] Grid monitoring.

In an agent oriented, pro-active fault tolerant grid framework was used in which faults are divided into six classes: (a) Hardware faults, (b) Application and operating system faults, (c) Network faults, (d) Software faults, (e) Response faults and (f)

Timeout faults. These are further divided into different sub-classes, where agents deal with individual faults proactively. Agents maintain information about hardware conditions, executing process memory consumption, available resources, network conditions and component mean time to failure. Based on this information and critical states, [7] agent enables the grid system to tolerate faults.

In Failure-Aware Grid Resource Management system [6] the Virtual Resource Manager (VRM) which supports QoS by means of SLAs. In this work it addresses the problem of remapping reservation to other resources when the originally selected resource fails. It mainly focuses on those jobs that are scheduled to a failed resource and not yet started its execution, which is the so called in-active jobs. Instead of dealing with fault tolerance of active jobs which usually requires checkpointing and migration. It computes a remapping interval during which it remaps those jobs that are assigned to a faulty resource and are inactive to some other resource in advance before it begins its execution. A min-max checkpoint placement method [1] is introduced that determines the suboptimal checkpoint sequence under uncertain circumstances in terms of the system failure time distribution. However, even if the (sub)optimal checkpointing interval is computed beforehand, the distributed system or application parameters upon which the interval is based will presumably change over time. Therefore, new forms of checkpointing optimization were recently considered in literature. One of them is the so-called cooperative checkpointing concept, which addresses system performance and robustness issues by allowing the application programmer, the compiler and the runtime system to jointly decide on the necessity of each checkpoint. The checkpointing algorithms used in this paper are based on this concept and thus are cooperative (adaptive) heuristics.

3. PROPOSED WORK

In the proposed work fault tolerance is achieved by dynamically adapting the frequency of checkpoints mainly to increase the throughput, and in case of resource failure, the proposed FIBR Algorithm reschedules the job from failed resource to some other available resource based on fault occurrence history, and then the job is executed from the last saved checkpoint. The grid model (Figure 1) considered in this paper consists of [1]: geographically distributed computational sites with many computational resources (r) at each site. The latter include a user interface (UI) through which the jobs are submitted into the system; a Resource Broker (RB) which is used to identify all the available resources, a scheduler(S) to schedule the job to the available resources. A checkpoint server (CS) where Checkpointing data is made persistent. An Information server (IS) which collects the job and resource status information required by the scheduler and checkpoint server. It maintains the history of information about each and every resource. Assume the scheduler, Information server, checkpoint server are protected against failure and only the computational resources are unstable.

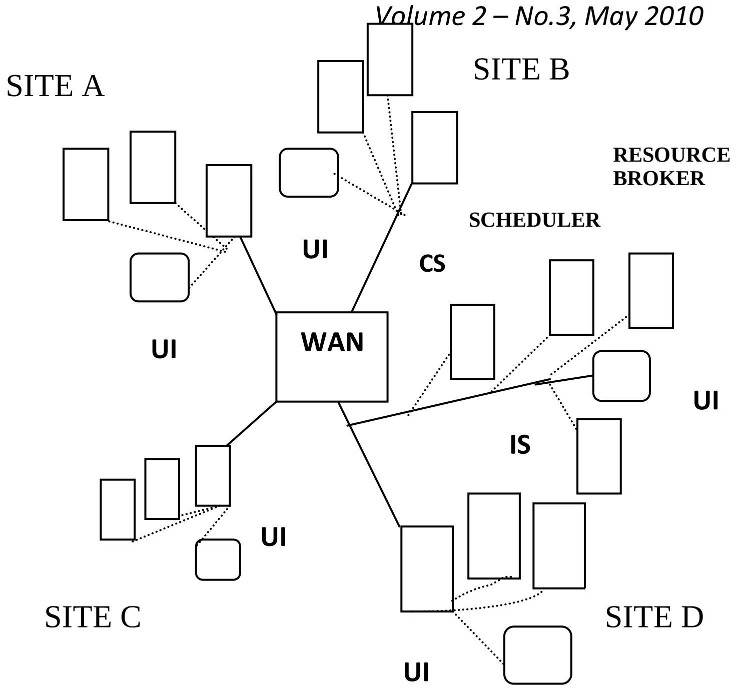

Figure 1: Grid Architecture: CS(Checkpoint Server),UI (User Interface),WAN (Wide Area Network), IS (Information Server). Each site with 32 computational resources r.

The work in this paper can be broadly divided into two parts: 1. Dynamic adaptation of checkpoints, 2.Rescheduling in case of resource failure.

3.1 Dynamic Adaptation of Checkpoints

This approach mainly concentrates on achieving fault tolerance by reducing unnecessary checkpoint overhead which will reduce the job throughput. Hence to reduce the unnecessary checkpoints the two algorithms differentiates the checkpointing interval based on the history of failure frequency of the resource and current status of a particular job. Here we consider two parameters: the last failure time of a resource, and the mean failure time of resource. These parameters suggest the stability of the resource based on which the checkpointing interval is omitted or modified.

By dynamically changing the checkpoint frequency [1], we will, on one hand, eliminate unnecessary checkpoints and, on the other hand, introduce extra job state savings, where the danger of failure is considered to be severe. More specifically, the optimal checkpointing interval for a job j (I^j_{opt}) running on the computational node r depends on the following parameters: E^j_r is the execution time of j on the resource r. F_r is the average time between failures of r. Additionally, the value of I^j_{opt} should satisfy the inequality $C < I^j_{min} < I^j_{opt}$ to be sure that jobs make execution progress despite of periodic checkpointing. C is the runtime overhead which is the time delay resulting from interruption of job execution to perform checkpointing. I^j_{min} is the minimum checkpointing interval of j, which should be initialized with a default value, for example, a small percentage of E^j_r. Here we assume the total execution time of the job is exactly determined in advance.

3.2 Last Failure Time Based Checkpoint Adaptation

The main aim of this Last Failure time based Checkpoint Adaptation (LFCA) algorithm[1] is to omit unnecessary checkpoint in-order to reduce the checkpoint overhead on a relatively stable resource. This unnecessary checkpoints are omitted mainly to reduce the overhead and to increase the job throughput. This algorithm considers the last failure time (Lf_r) of the resource, which is one of the parameter that suggests the stability of a resource. The operation of this algorithm on a resource, executing a single job is diagrammatically represented in Figure 2 which is explained below:

Step1. The job is submitted to the resource (r), and after an execution interval I, the job running on an active resource generates a checkpoint request.

Step 2. For each resource the algorithm gets the last failure time (Lf_r) from the information server, when no failure has occurred, the Lf_r is initiated with the system start time.

Step3. The checkpoint request generated by the job is evaluated by the scheduler (S) and it is allowed only if the comparison of $tc\text{-}Lf_r <= E^j_r$ evaluates to true, where tc is the current system time. If the $tc–Lf_r > E^j_r$ it is assumed that the resource is stable and the checkpoint is omitted to avoid the overhead.

Step 4: To prevent too many checkpoint omissions, a maximum number of omission limit should be defined. Thus this approach reduces the checkpoint overhead by omitting the unnecessary checkpoint.

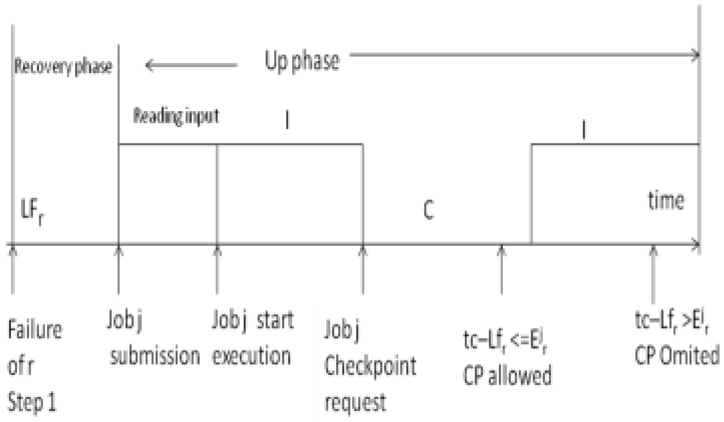

Figure 2: Operation of LFCA on a resource running a single job.

3.3 Mean Failure Time Based Checkpoint Adaptation

The Mean Failure based Checkpoint Adaptation (MFCA) algorithm [1] dynamically modifies the checkpointing frequency and deal with inappropriate checkpointing intervals. The checkpointing frequency is modified based on the Remaining job execution time (RE^j_r) and mean failure interval of the resource (Mf_r) where r is the resource and j is the job assigned to that resource.

The use of mean failure time instead of last failure time, reduces the effect of individual failure event. The operation of this algorithm on a resource running on a single job is diagrammatically represented in Figure 3 which is explained below:

Step 1:Once the job starts its execution and after an execution interval t_i, the job j issues a checkpoint request.

Step 2: If $RE^j_r < Mf_r$ and $I^j_r < \alpha * E^j_r$, where $\alpha<1$, then the frequency of checkpointing will be reduced by increasing the checkpointing interval, $I^j_r new=I^j_r old+I$.

Where RE^j_r is the remaining execution time of the job, I^j_r is the customized checkpoint interval, I is the time interval that is added to increase or decrease the checkpoint interval. The first inequality in the condition ensures that either r is sufficiently stable or the job is almost finished, while the second limits the excessive growth of I^j_r compared to the job length. The latter can particularly be important for short jobs, for which the first condition almost always evaluates to true.

Step 3: Else the checkpointing frequency is increased by reducing the checkpoint interval, $I^j_r new=I^j_r old-I$. while reducing the checkpoint interval, the following constraint should be taken into account: $C<Imin<=I^j_r new$.

This ensures the time between the consecutive checkpoints is never less than time overhead added by each checkpoint. This reduces the unnecessary checkpoints by increasing the checkpoint interval for relatively stable resource.

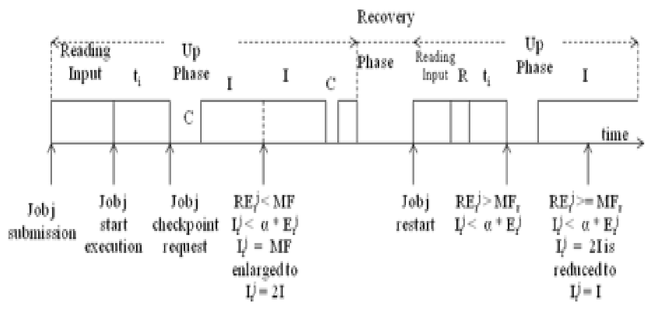

Figure 3: Operation of MFCA on a resource running a single job.

3.4. Rescheduling

The job running on a resource is rescheduled to some other resource in case of resource failure. The proposed Fault Index Based Rescheduling (FIBR) algorithm is explained below:

Step 1: The user submits the job with its deadline, and estimated execution time. After allocating the job to the resource, the Resource Broker expects a response of job execution within a time interval. This time interval is a function of the speed of a resource and communication latency between Resource Broker and the resource.

Step 2: If the resource could not get the result of execution within that time interval as specified by the grid manager, it realizes the fault has occurred, and increments the fault index of that resource by 1, or decrements by 1 on successful completion. This value is updated and stored in the Information Server.

Step 3: When there is a resource failure, the job executed on the failed resource is rescheduled by checking the fault index value of the available resources from the information server. The fault index value suggests the rate of tendency of resource failure. Lesser the fault index value, lesser is the failure rate of the resource

Step 4: Based on the fault index value the job is rescheduled to some other available resource with least fault index value and executed from the last saved checkpoint. Thus increases the percentage of job execution.

On combining the checkponting method with FIBR rescheduling, and when we compare the two methods, the MFCA along with FIBR proves to be effective than LFCA with FIBR.

4. PERFORMANCE ANALYSIS

When comparing the two dynamic checkpoint adaptation techniques: The Last failure time based adaptation(LFCA), Mean failure time based adaptation (MFCA). The Mean failure time based adaptation of checkpoints proves to be effective in terms of number of number of successful job execution, average number of checkpoints and average job execution time.

When comparing LFCA and MFCA in terms of Average job execution time Figure 4 the average job execution time of MFCA is less when compared to LFCA, which inturn increases the job throughput.

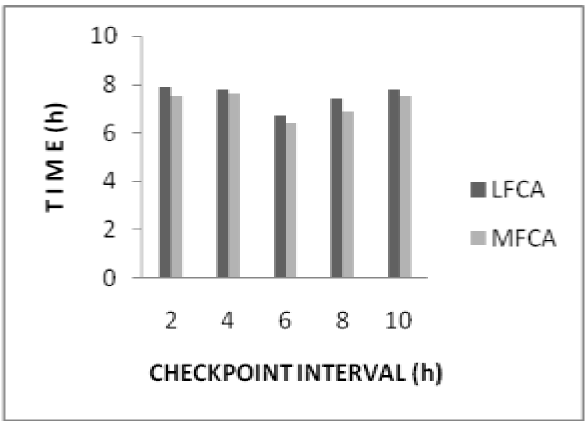

Figure 4: Average Job Execution Time.

When comparing LFCA and MFCA in terms of number of successful job execution,Figure 5, MFCA has higher number of successful job execution when compared to LFCA

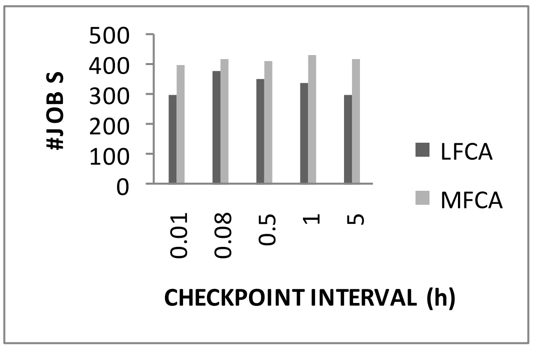

Figure 5: Jobs successfully executed.

When comparing the average number of checkpoints in both LFCA and MFCA Figure 6 as the checkpoint interval increases the average number of checkpoints in MFCA is comparatively higher than LFCA.

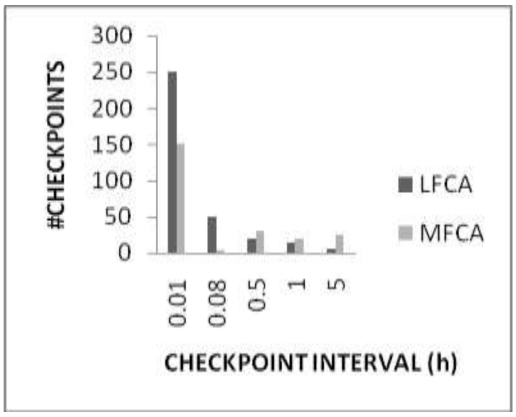

Figure 6: Average number of checkpoints.

5.CONCLUSION

Fault tolerance forms an important problem in all distributed environments. Here we address the problem of fault tolerance in terms of resource failure. Thus the proposed work achieves fault tolerance by dynamically adapting the checkpoint frequency, based on history of failure information and job execution time, which reduces checkpoint overhead, and increases the throughput. And in case of resource failure, the proposed Fault Index Based Rescheduling (FIBR) algorithm effectively reschedules the job from failed resource to some other available resource with least fault index value based on the history of fault occurrence information which is available in the information server. Hence the Dynamic adaptation of checkpoint method, dynamically varies the checkpoint frequency, increases the job throughput, and thus makes the grid environment trustworthy.

6. REFERENCES

[1]. Chtepen, M.; Claeys, F.H.A.; Dhoedt, B.; De Turck, F.; Demeester, P.; Vanrolleghem, P.A. Adaptive Task CHECKPOINTING and Replication: Toward Efficient Fault-Tolerant Grids Parallel and Distributed Systems, IEEE Transactions on Volume 20, Issue 2, Feb. 2009 Page(s):180 – 190 Digital Object Identifier 10.1109/TPDS.2008.93

[2]. Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, Eucalyptus: A Technical Report on an Elastic Utility Computing architecture Linking Your Programs To Useful Systems.UCSB computer science technical report number 2008-2010

[3] Favarim, F.; da Silva Fraga, J.; Lung Lau Cheuk; Correia, M. .GRIDTS: A New Approach for Fault- Tolerant Scheduling in Grid Computing Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on Volume ,Issue,12-14 July 2007 Page(s):187–194 Digital ObjectIdentifier 10.1109/NCA.2007.27

[4]. Fangpeng Dong and Selim G. Akl January 2006 Scheduling Algorithms for Grid Computing:State of the Art and Open Problems. Technical Report No. 2006-504 School of Computing, Queen's University Kingston, Ontario

[5] Foster,I.; Yong Zhao; Raicu,I.; Lu,S; Grid computing and Grid computing 360-degree compared. Grid computing environments workshop,2008.GCE'08 12-16 Nov.2008 pages:1-10.

[6]Lars-Olof Burchard, C´esar A. F. De Rose, Hans Ulrich Heiss, Barry Linnert and J¨org Schneider. VRM: A Failure-Aware Grid Resource Management System. Proc. of the 17th Intl: Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05). IEEE. 2005

[7]Mohammad Tanvir Huda, Heinz W. Schmidt and Ian D. Peake. An Agent Oriented Proactive Fault tolerant Framework for Grid Computing. First International Conference on e-Science and Grid Computing (e-Science'05).IEEE. 2005

[8]R. Medeiros, W. Cirne, F. Brasileiro and J. Sauve, .Faults in Grids: Why are they so bad and What can be done abut it? in the proceedings of the Fourth Intl: Workshop on Grid Computing (GRID'03), 2003.

[9] Nazir, B.; Khan, T.Fault Tolerant Job Scheduling in Computational Grid. Emerging Technologies, 2006. ICET apos;06. International Conference on Volume , Issue, 13-14 Nov.2006 Page(s):708–713 Digital Object Identifier 10.1109/ICET.2006.335930

[10] D. Feitelson, Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/, 2008

[11] Jang-uk In,Paul Avery, Richard Cavanaugh. SPHINIX:A fault tolerant system for scheduling in dynamic environments,proceedings of the 19th IEEE international parallel and distributed processing symposim.

[12]www.gridbus.org/gridsim/

[13] grid simulator.http://www.buyya.com/gridbus/gridsim/, released on Apr 08, 2009

[14] S. Agarwal, R. Garg, M. Gupta, and J. Moreira, "Adaptive Incremental Checkpointing for Massively Parallel Systems," Proc.18th Ann. Int'l Conf. Supercomputing (SC '04), Nov. 2004.

[15] A. Subbiah and D. Blough, "Distributed Diagnosis in Dynamic Fault Environments," Parallel and Distributed Systems, vol. 15, no. 5,pp. 453-467, 2004.