IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 1 (May. - Jun. 2013), PP 31-38
www.iosrjournals.org

Adaptive check-pointing and replication strategy to tolerate faults in computational grid

Subash Thapa[1], Abhijit bhowmick[2]
[1]*(computer Science, NIT Kurukshetra, INDIA)*
[2]*(computer Science, NIT Kurukshetra, INDIA)*

Abstract: *We have considered both adaptive resource replication and adaptive check-pointing. Complete resource failure has been considered which was ignored in most of the previous works. We have proposed an efficient solution for scheduling as well as a fault tolerant method based on maintaining a balance between the overheads of replication and check-pointing. The selection of resource from available resource is done such that no partiality is done between jobs of equal priority. The fault tolerance method, i.e., resource failure has been resolved in our proposed solution with the help of replication, three replicas for each task and number of replicas will be increased by one on each permanent failure to maintain total replicas three and new replica start its execution from the last saved checkpoint .*

Keywords: *Grid, fault,tolerance, redundancy,replication,adaptive,check pointing, Grid Computing*

I. Introduction

.The main characteristics of grid are:
 Multiple administrative domains Since, it spawns into multiple administrative domains, the policies and autonomy of different domains needs to be maintained.
Heterogeneity: Grid contains different types of resources like personal computers to super computers to specialized devices like telescopes. Grid provides seamless way to success different resources.

Scalability: A grid might grow from few integrated resources to millions. The performance might be degraded as the size of grid increases.

1.3 Types of Grid Systems

The types of grid are mentioned below:
Computational Grid: A computational Grid is a system that aims at achieving higher aggregate computational power than any single constituent machine.
Data Grid: A Data Grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data.

1.4.1 Resource Allocation

Available resources should be used maximally or optimally in order to balance the distribution of tasks over the network.

If node failure or communication link failure occurs, it is required to re-allocate and manage the tasks and resources from start.

1.4.3 FAULT TOLERANCE

Fault-tolerance or graceful degradation is the property that enables a system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components.

II. Previous Work

In previous works, several mechanisms were proposed for grid or distributed computing systems. However, some of them used only space redundancy (hardware replication), and others used only time redundancy (check-pointing and rollback).In 2007, Sangho Yi, Derrick Kondo has published a paper for reducing a number of replicas by using check-pointing and rollback and also analyzed the probability of successful execution within the given deadline and reliability requirement of each task for selecting replicas. In that paper major drawback analyzed by us are:
  In this paper, author assumes both permanent and recoverable failures occur according to the Poisson process with rate λfp and λfr, respectively for calculation of no replicas but If the failures occur with non-

exponential distribution, (e.g. Gamma, Weibull, etc.), it is very hard to use the existing probabilistic distribution for the same.

In this paper, check-point interval is not major issue, does not talk about this and each recoverable failure on each replica required to resume from its last saved check-point.

. Previous works done by Babar Nazir, Kalim Qureshi and Paul Manuel in 2008 maintained fault index of grid resources which doesn't keep track of permanent failure and recoverable failure. But in this paper major drawback was:

The fault tolerance method used by the author in this paper is based only on check-pointing.

The priority (deadline) of task has not been considered in this paper. As it is a economy based system , task of different priority from user is a certainty.

III. Problem formulation:

Grid users submit their jobs to the grid resource broker (GRB) by specifying their QoS requirements, i.e., deadline in which users want their jobs to be executed and the budget which users have for the completion of jobs.

Grid resource broker schedules user jobs on the best available resource by optimizing time.

Result of the job is submitted to user upon successful completion of the job.

Such an economy based environment has following two major draw backs:

In the economy based grid environments, there are resources that fulfill the criteria of deadline and budget constraints (QoS requirements), but they have a tendency toward faults. In such a scenario, the grid resource broker goes ahead to select the same resource for the mere reason that the grid resource promises to meet QoS requirements of the grid jobs. This eventually results in compromising the user's QoS parameters in order to complete the task.

In this paper, in order to address the first problem, we use a task check-pointing heuristic to enable the economy based grid to tolerate faults gracefully, as we are able to restore the partially completed task from the last checkpoint. And we also use replication on at most three best available resources on the basis of a probability value of permanent resource failure.

In order to address the second problem, we make our check-pointing strategy adaptive by maintaining a probability value of recoverable resource failure. In this way, we are able to introduce checkpoint mostly when it is necessary.

IV. System Model and Assumption:

We have considered an economy based computational grid computing environment. In this system we assume the resources R1, R2, R3……………………………..……… Rn with following characteristics value.

Some common notations and functions used throughout paper are:

Np: Denotes count of permanent failure.

Nr: Denotes count of Recoverable failure.

Ns: Denotes count of successful execution of task.

Pp: Probability value of permanent failure.

Such that $Pp = Np / (Ns+Np+Nr)$

□ □ □ Pr: Probability value of recoverable failure.

Such that $Pr = Nr / (Ns+Np+Nr)$

□ □ □ Pt: Probability value of Total failure.

Such that $Pt = Pr + Pp$

□ □ □ C(i), $i = 1,2, \ldots ,N$, are the percentage of task completed such that

$0 <= C(i) <= 100$ and $C(1) > C(2) > \cdots > C(N)$

F(i), i = 0, 1,2, . . . ,N, are integers such that F(0) < F(1) < ……… <F(N) and represent index range of fault index.

Proposed Idea:
t_r denote the work requirement time of a task

$T(t_r)$ denote expected total execution time such that in the absence of any failures $T(t_r) = t_r$

The $T(t_r)$ can expected execution time, $T_i(t_i,c_i)$, where the t_i and c_i be calculated as the sum of all intervals are the work requirement time and check-pointing overhead of an ith interval.

t_d represents the deadline for a task.
P_s is a user-defined probability of successful execution of a task within the deadline.
$P(t_r, t_d)$ is probability of successful execution of a task which has tr as the work . requirement time and td as the deadline
He assume that the permanent and recoverable failures occur according to *Poisson* process at rate λf p and λf r , respectively

Theorem 1
If we assume the recoverable failures occur according to the Poisson process with rate λ_{fr} and that failures can occur during checkpointing, the expected execution time of the ith interval with check- pointing, $T_{ri}(t_i, c_i)$ is given by

$$T_{r_i}(t_i, c_i) = \frac{(e^{\lambda_{fr}(t_i+c_i)} - 1)(1 + \lambda_{fr}r) - \lambda_{fr}(t_i + c_i)}{\lambda_{fr}}$$

Theorem 2
If we assume both permanent and recover able failures occur according to the Poisson process with rate λ_{fp} and λ_{fr}, respectively, the probability for successful execution for the time requirement t_r within the deadline td , $P(t_r, t_d)$ is given by

$$P(t_r, t_d) = e^{-\lambda_{fp} t_d} \cdot \left(1 - e^{-\frac{t_d - \sum_{i=1}^{n_c}(t_i + c_i)}{T_r(t_r)}}\right)$$

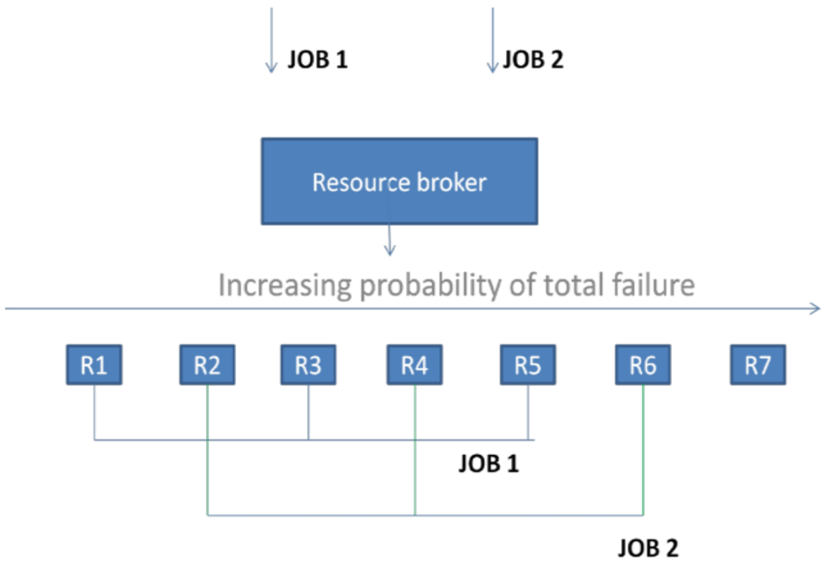
V. Adaptive fault tolerant job scheduling strategy

Grid faults affect the performance of resource management strategy. As our proposed strategy considers fault tolerance in an economy based computational grid environment, the aim is to optimize user-centric metrics in the presence of faults

We propose to maintain and update probability value of permanent or recoverable resource failure of all available resources of the grid. The probability value of the grid resource will suggest its vulnerability to faults (i.e. higher the probability value is higher the failure rate).

VI. Components of proposed scheduling strategy:

Grid resource:
Grid Information Server: Grid users register themselves to the Grid Information Server (GIS) of a grid by specifying the QoS requirements
Grid Resource Broker (GRB): Each grid job (composed of gridlets) is first submitted to its broker. When GRB receives a grid job from a user, it gets the contact **information** of available grid resources from the GIS and then requests the resources to send their current work load condition.



```
ALGORITHM A
Select the resource according to QoS from available resource.
Arrange them in increasing probability value of total failure.
Select the three replicas for equal priority job such that no partiality is done at time t as shown in fig.

F: probability value of the selected grid resource

F(i), i = 0, 1, 2, . . . ,N, are integers such that $F(0) < F(1) < \cdots < F(N)$

C(i), i= 1, 2, . . . ,N, are the percentage of task completed such that

$<= C(i) <= 100$ and $C(1) > C(2) > \cdots > C(N)$

IF $F(0) <= F < F(1)$ THEN

The task is queued to that resource with a checkpoint at C(1) (It indicates that the grid resource has to send the result of the task completion to the CPManager when C(0)% task is completed.)

GO TO Step 6.
END IF

IF $F(1) <= F < F(2)$ THEN

The task is queued to that resource with a checkpoint at C(2).

GO TO Step 6.

END IF

IF $F(2) <= F < F(3)$ THEN

The task is queued to that resource with a checkpoint at C(3).

GO TO Step 6.

END IF

IF $F(N- 1) <= F < F(N)$ THEN
```

The task is queued to that resource with a checkpoint at C(N).

GO TO Step 6.

END IF

IF F(N) <=F THEN
Remove resource from the available resources list and label it as unavailable resource (i.e., no job is assigned) to that resource until:
It has passed ALIVE time successfully. (ALIVE time is the time interval for which grid resource must be up so that it can be again included in the grid available resource list.) END IF

Add the job to the unassigned job list and reapply the time optimization algorithm for this task on available resources.

EXIT
Fault Tolerant Schedule Manager (FTScheduleManager): we assume that a fault occurs when a fault detector is unable to receive any signal or response from grid resource in pull model / push model. Then a fault is detected by a fault detector, the fault occurrence information about the grid resource is updated in Fault Tolerant Schedule Manager that would be both recoverable failure information and permanent failure information. Task successful execution information is also send by resource broker to Fault Tolerant Schedule Manager.
Thus fault probability information is used, while making a job allocation decision to the grid resource. It uses following algorithm i.e. Algorithm B:

ALGORITHM B:
IF gridlet receptor receives the task completion result from the CPManager THEN a. IF resource fault index >= 1 THEN
Send a message to FTScheduleManager to increment Ns END IF

b. GOTO Step3 END IF
IF the to FTScheduleManager receives the task failure message from fault detector, THEN a. IF resource is permanent failure THEN
Send a message to FTScheduleManager to increment Np

Send a message to CPManager, whether there is any checkpoint result of

this task.

iii. IF checkpoint result of the task exists in CPManager’s task checkpointing table, THEN Inform the resource broker with that check point value to replicate on another resource.

GOTO Step 3. END IF

iv. IF checkpoint result of the task does not exist in CPManager’s task check-pointing table, THEN

Inform the resource broker with that check point value to replicate on another resource.

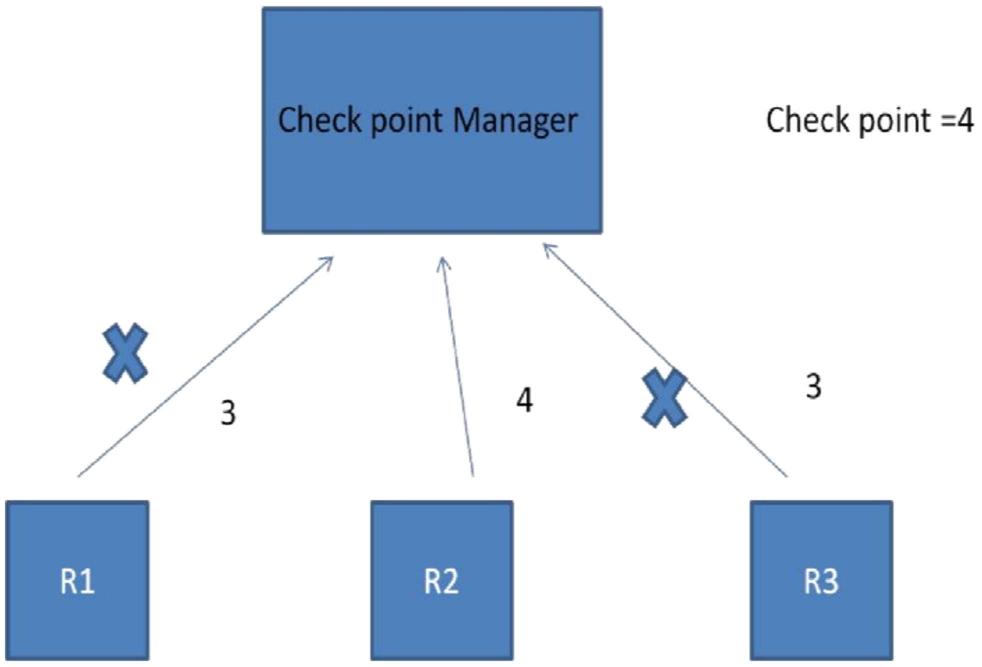
GOTO Step 3. END IF

END IF

EXIT

Checkpoint Manager (CPManager): It receives the partially executed result of a

task from a grid resource in the intervals specified by the grid resource broker based on the checkpoint. It maintains grid tasks and their checkpoint table which contains information of partially executed tasks by the grid resources. CPManager also receives and responds to the task completion and task failure message from grid resources. For a particular task, the CPManager discards the result of the previous checkpoint when a new value of checkpoint result is received with later value. For a particular task, if CPManager receives the task completion message from resource, it removes its entity from the task checkpoint result table. In our proposed algorithm it takes maximum of available check-point from replication.



Grid dispatcher: It dispatches the tasks one by one to the respective grid resource as listed in the queue by GRB. A small grid task is called a gridlet.
Gridlet receptor: It receives the task (gridlet) execution result from the grid resource where the task is dispatched by the dispatcher.

WORKING:
Resource entity registers themselves to the Grid Information Service (GIS) entity.
Fault Tolerant Schedule Manager (FTScheduleManager) sends a synchronous event to the GIS entities to send the list of resource available.
The GIS entity returns a list of registered resources to FTScheduleManager. FTScheduleManager then initializes the resource fault occurrence history table
that contain the fault index (suggest resource vulnerability toward failures) of the resources available.

IV. The GIS entity returns a list of registered resources and their contact details to the broker.
The broker entity finds the current work load condition of the resources by sending events to resource entities requesting them to send their resource configuration and properties.
VI. Resource entities respond to this event by sending their dynamic information that includes resources cost, capability, availability, load, and other configuration parameters.
VII. The broker entity sends synchronous events to FTScheduleManager to get the probability value of total resource failure.
VIII. The broker entity, after receiving probability value of total resource failure index list, dispatches gridlet to replicate on three resource entities if available such that all job of equal priority will get equal efficiency resource entities at a time t.
Fault Tolerance method:
I. Resource failure has been resolved in our proposed solution with the help of replication, 3 replica for each task.

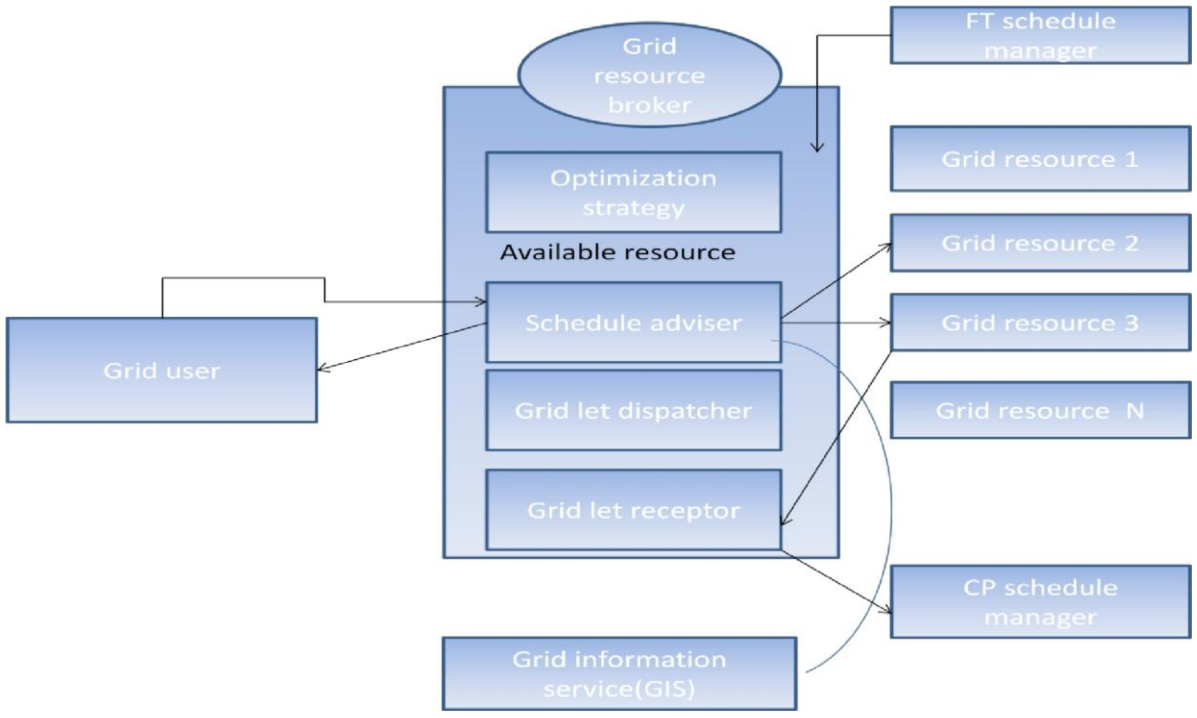
When the resource entity fails to process the gridlet, detected by fault detector then fault detector sends gridlet failure event to the CPManager entity and to FTScheduleManager.

III. The CPManager entity on the receipt of gridlet failure event from resource entity sends an asynchronous gridlet failure event to broker entity .

IV. The checkpoint at the check point manager is adaptively updated by taking the maximum of the checkpoints from the 3 replicated resource periodically.

On permanent failure of resource, maximum of the last saved check points is taken from the check point manager to replicate task on another resource by resource broker but on recoverable failure restart the process on same resource with last saved check-point. Thus taking advantage of the different reliability and computation speed of the replicated resources.

Page 22 of 27
VI. Task is resumed from this check point on a new replicated resource .Thus in this way maximum number of replicas for eack task is fixed at 3. i.e resource wastage is minimized.



VII. Conclusion

We have proposed an efficient solution for scheduling as well as an adaptive fault tolerant method based on maintaining a balance between the overheads of replication and check-pointing. The case of complete resource failure has been considered. Our proposed solution minimizes the resource wastage due to replication by taking an optimal number of replicas. At any time the check point manager maintains a unique check point which it keep on updating periodically , by taking the maximum of the received check points from the three replicas. This utilizes the different computational speed of the replicated resources. Use of probability value instead of index value further takes into account the case of permanent failure while scheduling and while deciding the check point intervals.

References

[1] "Adaptive checkpointing strategy to tolerate faults in economy based grid" Babar Nazir · Kalim Qureshi · Paul Manuel Published online: 16 October 2008 © Springer Science+Business Media, LLC 2008
[2] "Using Replication and Checkpointing for Reliable Task Management in Computational Grids" Sangho Yi, Derrick Kondo INRIA, France {sangho.yi, derrick.kondo}@inria.fr
[3] H. Z. E. Tsiakkouri, R. Sakellariou and M. D. Dikaiakos, "Scheduling Workflows with Budget Constraints," in Proceedings of the CoreGRID Workshop "Integrated research in Grid Computing", S. Gorlatch andM. Danelutto, Eds., Nov. 2009, pp. 347-357.
[4] H. Topcuouglu, S. Hariri, and Min-you Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260-274, 2010.

[5] M. Wieczorek, A. Hoheisel, and R. Prodan, "Taxonomies of the Multi-criteria Grid Workflow Scheduling Problem," in Proceedings of the CoreGRID Workshop on Grid Middleware. Dresden, Germany: Springer-Verlag, June 2009.

[6] J. Yu and R. Buyya , "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms, "Scientific Programming Journal, vol. 14, no. 1, 2009.

[7] J. Yu, R. Buyya, and C. K. Tham, "Cost-based Scheduling of Scientific Workflow Applications on Utility Grids" , in Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), IEEE. Melbourne, Australia: IEEE CS Press, Dec. 2009.

[8] J. Yu, M. Kirley, and R. Buyya, "Multi-objective Planning for Workflow Execution on Grids," in Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), IEEE. Los Alamitos, CA, USA: IEEE CS Press, June 2010.