A Survey On Monitoring Model To Meet Deadlines In Scientific Workflows Within Cloud

Chitra. B,
P.G. Scholar,
Angel College of Engineering and Technology

Sreekrishna. M,
P.G. Scholar,
Angel College of Engineering and Technology

Malarvizhi.V,
P.G. Scholar,
Angel College of Engineering and Technology

Abstract

In the heterogeneous parallel and distributed computing environments like cloud there were many related approaches proposed for fault tolerant execution of scientific workflows. Many of the works involved earlier does not rely on resource failure prediction that is hard to achieve even with years of historic failure trace data of the target environment and the monitoring has not kept pace. Here Resubmission Impact (RI) that tries to establish a metric describing the impact of resubmitting a task to the overall execution time of a workflow application, and to adjust the replication size of each task accordingly is established. In this paper, to solve the software fault prediction, unavailability of the resources and monitoring problems we propose a failure prediction model that involves two different methods. In order to predict the failures we propose a method using IPMI that monitor the failure at nodes and provide the corresponding data useful for determining likely imminent failures. The other method is to predict the Unavailability from past behavior generates some initial results that indicate that nodes differently from one another and their failure is somewhat predictable and monitoring performed which intimates about the failure.

Keywords *Fault prediction, IPMI, Checkpoint, Task Replication.*

1. Introduction

To improve the fault tolerance of scientific workflow applications, which emerged in the last decade as one of the most successful paradigms for programming e-science applications in highly distributed environments such as Grids and Clouds. Currently, there are two fundamental and widely recognized techniques to support fault tolerance in distributed environments: resubmission and replication. Resubmission tries to re-execute a task after a failure which can significantly delay the overall completion time in case of multiple repeating failures. Replication submits several copies of the same task in parallel on multiple resources which suffers from potentially large resource consumption. To find a compromise balance between these two complementary techniques, we took a algorithm called Resubmission Impact (RI) that tries to establish a metric describing the impact of resubmitting a task to the overall execution time of a workflow application, and to adjust the replication size of each task accordingly.

The number of dynamic resources in the cloud system increases continuously, so fault tolerance techniques the resubmission and replication becomes a critical property for applications running on these resources. However, in traditional implementations, when a failure occurs, the whole application is shutdown and has to be restarted from the beginning. A technique to avoid restarting of the application from the beginning is rollback recovery which is based on the concept of checkpoint. Checkpoint mechanism is used to reduce the limitations imposed by the high volatility of resources. It periodically saves the application's state to stable storage. So, whenever a failure interrupts a volunteer computation, the application can be resumed from the last stable checkpoint. Checkpoint-recovery techniques make it possible for the job to resume execution from the last checkpoint instead of restarting from the beginning, whenever a failure occurs. Over provisioning techniques replicate a job in more than one resource to increase the probability of successful execution. Although these techniques address the reliability challenges to some extent, no large-scale study has been done on how effective they are when coupled with scheduling. The Resubmission Impact

(RI) mentioned before if a failure occurs it will recover from the beginning so in this paper we propose a checkpoint mechanism that is a failure prediction model which involve two methods Predicting Node failure using IPMI and Predicting Unavailability from Past Behavior.

The paper is organized as follows. Section 2 discusses related work, Section 3 introduces the workflow and infrastructure models, task replication , Resubmission Impact heuristic and scheduling for soft deadline and in Section 4 conclusion is given.

2. Related Work

Considering the faults in scientific workflows the existing techniques available are by Jia Yu,Rajkumar Buyya and Chen Khong Tham [1]minimizes the cost of execution while meeting the deadline.But on the other hand it uses run-time rescheduling to handle service agreement violations and problem occurs while scheduling dynamic pricing. The work [2] not finds traces that include mostly production workflows submitted by real users. L Guo, A S McGough, A Akram, D Colling, J Martyniak, M Krznaric [3] provide a level of QoS through resource selection from priority information along with the use of advanced reservation but which will not allow to dynamically change the execution of the workflow once deployed to the engine.

The advance reservation can have a major impact on execution time and can increase considerably predictability of a Grid environment and lacks in concentrating dynamic reservation algorithms for heavily loaded Grid environments [4]. Jayadivya S K* Jaya Nirmala S Mary Sair Bhanu S [5] proposed an approach with a prioritization of tasks that helps to meet the deadline and reduces resource wastage along with providing fault tolerance for the workflow system. And does not consider the failures like data center shutdowns, network failures that may also added to faults. The technique of query planning is proposed in [6] applications that incorporate kernel-level and user-level checkpointing could receive signals generated by chip monitoring facilities that recognize node failure is imminent, alleviating the need to take periodic checkpoints.

Ozan Sonmez, Nezih Yigitbase, Alexandru Iosup [7] proposed methods cannot improve and adapt to computational grids where the historical runtime and wait time data considered are non-stationary. The approach mentioned in [9] developed to extend detection to lower levels, such as hardware and job execution faults. The performance analysis of list scheduling [10] is done and the total execution of the program is minimized and even though it does not degrade but not improve the worse case and average case performance.

This method has the advantage of applicability to new and unknown environments, however, it often leads to unnecessarily large resource consumption and to large differences between the expected execution time (as promised to the end user) and the real execution time. Our work belongs to this third category and brings two advantages over existing methods: 1) it reduces the resource consumption, and 2) it offers improved QoS by meeting soft deadlines.

3. Proposed System

3.1. The Model

Let A Set= {A1; . . .;An} denote the set of n activities or tasks in a workflow and DSet={(A_i;A_j; $Data_{ij}$)|A_i;A_j ∈ ASet} the set of control and data flow dependencies, where $Data_{ij}$ denotes the amount (i.e., bits) of data that needs to be transferred from A_i to A_j before A_j can start its execution (if $Data_{ij}$=0, the dependency becomes control flow only). Let w_i ∈ Work denote the amount of work (e.g., number of instructions) that each task A_i ∈ ASet requires in order to be completely processed. A workflow W_f is defined as a triplet: W_f={ASet;DSet;Work} representing a directed graph of computational tasks. We use pred : ASet→2^{ASet}, pred(A1)={A2 : ∀(A2;A1; $Data_{ij}$) ∈ DSet} to denote the set of predecessors and succ : ASet →2^{ASet}, succ={ A2 : ∀A1;A2; $Data_{ij}$∈DSet} to denote the set of successors of a task A1→ASet.

Our resource model consists of a set R of heterogeneous processing cores, referred in the following as resources. A schedule of a workflow is defined as a function sched :ASet →2^R that assigns to each task A 2^{ASet} a subset r ∈R of resources which simultaneously execute |r| replicas of A, where |r| denotes the cardinality of r and r≠ ∅ . We denote the schedule of task Ai ∈ ASet as: sched(Ai) = r. We assume a non preemptive scheduling model meaning that individual tasks cannot be rescheduled unless they are cancelled and then restarted. We further assume the availability of the execution time TAi for all tasks Ai ∈ ASet and for every resource p ∈R, which also includes the time for transferring data from the predecessor activities to p. This information is available from previous executions, or provided by an own prediction service based on regression and similarity methods

3.2. Task Replication Technique

In our work, the technique replication submits several copies of the same task in parallel on multiple resources which suffers from potentially large resource consumption. The idea behind task replication is that a replication of size r can tolerate r -1 failed tasks while keeping the impact on the execution time minimal. We call r the replication size. While this technique can help to successfully complete time-critical tasks, its downside lies in the large resource consumption. Algorithm 1 presents a trivial baseline heuristic called REPLICATEALL that schedules a workflow by simply replicating each task a fixed amount of times indicated by a maximum replication size rep_{max} input parameter. This heuristic builds first a replication vector RV defining the replication $_{sizes}$ $rep_i \in RV$ for each task $A_i \in A_{Set}$, and initializes it with the maximum replication size parameter rep_{max} (given by the user) for all tasks in the workflow Then, it schedules the workflow including the replicas onto the available resources R by invoking HEFT_REPLICATION that replicates each task according to the input replication vector RV and schedules the resulting workflow according to an extension of the HEFT algorithm.

Algorithm 1 REPLICATEALL heuristic

Wf = (ASet,DSet,Work): scientific workflow;
Require: R: resource set;
rep_{max}: maximum replication size;
Ensure: $|sched(A)| = rep_{max} : \forall A \in A_{Set}$

1: function REPLICATEALL(Wf, R, rep_{max})
2: RV$\leftarrow \bigcup_{i=1}^{Aset} \{repmax\}$
3: return HEFT_REPLICATION(Wf,R,RV)
4: end function

Algorithm 2 HEFT algorithm with task replication

Wf = (ASet,DSet,Work): scientific workflow;
Require: R: resource set;
RV : replication vector;
Ensure $|sched(A)| = rep_i : \forall A \in A_{Set} \wedge rep_i \in RV$

1: function HEFT_REPLICATION(Wf, R, RV)
2: ranks$\leftarrow$ B-RANKS(Wf)
3: sRanks $\leftarrow$SORT(ASet; ranks)
4: for all A $\in$ sRanks do
5: sched(A)$\leftarrow$ MAP ECT(A;R)
6: for all $A_i \in$ pred(A) do
7: if $sched(A_{succ}) \neq \emptyset : \forall A_{succ} \in succ(A_i)$ then
8: $C[1 : rep_i] \leftarrow$REPLICATE(A_i, rep_i)
9: $sched(C_j) \leftarrow$ MAP_ECT(C_j;R): $\forall j \in [1; rep_i]$
10: end if
11: end for
12: end for
13: return sched(A) : $\forall A \in A_{Set}$

3.3. Resubmission impact heurisitic

The resubmission tries to re-execute a task after a failure which can significantly delay the overall completion time in case of multiple repeating failures. In addition to the workflow application Wf and the set of resources R, the RI heuristic receives as input the maximum replication count rep_{max} and the maximum resubmission count res_{max}, which have to be given by the user. The RI heuristic consists of two phases.

The first phase establishes the RI metric. First, we make a copy Work 1 of the set Work defining the work of each task in the workflow. Then, we enlarge the amount of work wi in task Ai by multiplying it with the resubmission count maximum resubmission count, and define a new workflow Wf containing the new work amount for each task. Afterwards, we compute the difference in expected execution time and the real time by scheduling both of them using the HEFT algorithm. We repeat these steps for every task in the activities. Finally, we compute the RI for every task in Activity by normalizing the differences against the maximum of all the tasks.

In the second phase, the replicate function is again called and thus the tasks with a higher RI will have a larger amount of replicas to avoid expensive resubmission.

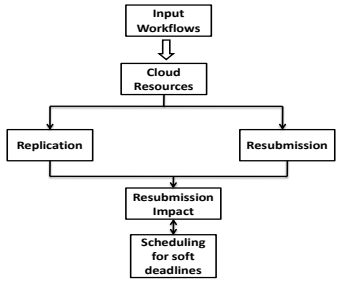

Figure 1. System Flow Diagram

From the figure 1, Resubmission Impact (RI) that tries to establish a metric describing the impact of resubmitting a task to the overall execution time of a

workflow application, and to adjust the replication size of each task accordingly. Real-life users typically want to know an estimation of the execution time of their application before deciding to have it executed. In many cases, this estimation can be considered to be a soft deadline that shall be satisfied with some probability We propose on top of the RI heuristic a method that first proposes to the end user a realistic soft deadline and then monitors and dynamically reschedules the workflow to meet the deadline in the absence of failure models.

3.4. Predicting the Failure Using Intelligent Platform Management Interface (IPMI)

To help reduce the amount of checkpoint data, and to aid in the overall maintenance and monitoring of a large computational cluster, an accurate yet light-weight tool must monitor for failed nodes and provide data useful for determining likely imminent failures. Such a monitoring tool would allow an application to decide at runtime whether a failure is likely to occur and consequently check whether a checkpoint is necessary.

We have to generate some initial results that indicate that nodes fail differently from one another, and that their failure is somewhat predictable. We track their movement between 5 different availability states (Available, User Present, CPU Threshold Exceeded, Unavailable, and Becoming Unavailable), and classify resources based on their behavior in terms of these states, over time. A predictor is used after this.

4. Simulation Results

To evaluate the performance of our fault tolerant system we have simulated the different structures and workflows by using CloudSim 3.0.

CloudSim is a framework for simulation and modeling of cloud computing environments mainly used for resource allocation and scheduling. It includes data centers, Virtual machines, brokers, cloudlets and hosts.

Here in the below table we have taken 5 virtual machines and its corresponding parameters.

Table 1. Virtual machine with its parameters

VMid	Mips	CPU	RAM	BW
0	350	1	512	1250
1	350	2	2048	700
2	500	3	1024	1500
3	400	1	3056	1000
4	350	3	512	800

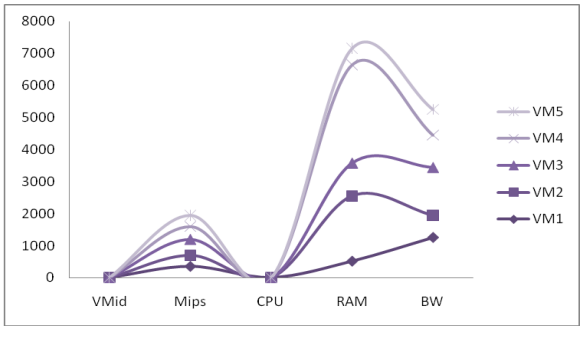

Figure 2. Workflows with Earlier Completion Time

Here 20 workflows are considered with 5 Virtual Machines and the Earlier Completion Time calculated with the above specified algorithm. The earlier completion time is also calculated for the respective tasks with the table given below.

The Table 2 and Figure 3 follows the Comparision Analysis of Various Parameters of Different Virtual Machines for our work

Table 2. Tasks with Earlier Completion Time

Task	ECT
8	122.22
3	499.99
5	309.98
10	428.56
16	99.54

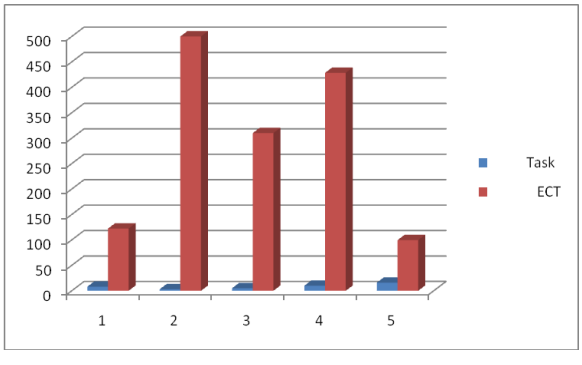

Figure 3. Performance Analysis of Tasks Scheduled Vs. Earlier Completion Time

5. Conclusion

In this paper a new heuristic called RI to increase fault tolerance of scientific workflows in highly distributed environments such as computational Grids in the absence of failure models is used. The heuristic is based on a combination of task replication and task resubmission using a new RI metric that describes the impact of task resubmission on the overall workflow makespan and we have a proposed a checkpoint mechanism that is a failure prediction model which involve Predicting Node failure using IPMI and Predicting Unavailability from Past Behavior.

Finally in the future, we have an idea of designing and testing a simple prediction-based scheduler, which chooses resources based on their predicted failure rate during the application's execution interval, their current CPU load, and their CPU speed. We compared results against two other schedulers, (i) a Condor-like scheduler, which chooses the available resource with the highest CPU speed, and (ii) a semi optimal scheduler that given oracle knowledge about the future availabilities of machines, chooses the machine with the fastest CPU speed that will complete the application without becoming unavailable.

REFERENCES

[1] Tharam Dillon, Chen Wu and Elizabeth Chang, "*Cloud Computing: Issues and Challenges*", 2010 24th IEEE International Conference on Advanced Information Networking and Applications.

[2] G.Kandaswamy, A. Mandal, and D.A. Reed, "*Fault Tolerance and Recovery of Scientific Workflows on Computational Grids*," Proc. IEEE Eighth Int'l Symp. Cluster Computing and the Grid (CGDRID '08), pp. 777-782, 2008.

[3] K. Plankensteiner, R. Prodan, T. Fahringer, A. Kertesz, and P.Kacsuk, "*Fault-Tolerant Behavior in State-of-the-Art Grid Worklow Management Systems*," Technical Report TR-0091, Inst. On Grid Information, Resource and Worklow Monitoring Services,CoreGRID—Network of Excellence, Oct. 2007.

[4]] M. Wieczorek, M. Siddiqui, A. Villazon, R. Prodan, and T. Fahringer, "*Applying Advance Reservation to Increase Predictability of Workflow Execution on the Grid*," Proc. IEEE Second Int'l Conf. e-Science and Grid Computing (E-SCIENCE '06), 2006.

[5]L.Guo, A.McGough,A.Akram,D.Colling, and J.Martyniak, "*Qos for Service Based Workflow on Grid*," Proc.Conf.UK e-Science 2007 All Hands Meeting, January 2007.

[6] Y.Zhang, D. Wong, and W. Zheng, "*User-level Checkpoint and Recovery for LAM/MPI*," SIGOPS Oper. Syst. Rev., vol. 39, no. 3, pp. 72–81, 2005.

[7] J. Yu and R. Buyya, "*A Taxonomy of Scientific Workflow Systems for Grid Computing*," ACM SIGMOD Record, vol. 34, no. 3, pp. 44- 49, 2005

[8] Seoko Son and Kwang Mong Sim ,"*A price and time slot Negotiation mechanism for Cloud Service Reservations*" in IEEE Transactions on Systems,Man, Cybernetics, June 2012

[9] Bernd Grobauer, Siemens CERT Tobias Walloschek, Siemens IT Solutions and Services Elmar Stöcker, Siemens IT Solutions and Services, "*Understanding cloud computing vulnerabilities*", IEEE Security and Privacy.

[10] S. Ostermann, R. Prodan, T. Fahringer, A. Iosup, and D. Epema, "*A Trace-Based Investigation of the Characteristics of Grid Workflows*," From Grids to Service and Pervasive Computing, pp. 191-203, Springer, http://www.springerlink.com/content/ x21m42878m456338/fulltext.pdf, Aug. 2008

[11] K.M.Sim and B.Shi, "*Concurrent negotiation and coordination for Grid resource coallocation*" IEEE Trans.Syst.,Man,Cybern.B,Cybern.,vol.40,no.3, pp.753-766,May2010

[12] J.Yu,T.Buyya and chen Khong Tham, "*Qos- Based Scheduling of Workflow Application on Service Grids*," Proc.IEEE First Int'l Conference e-Science GridComputing (eScience'05), Jan.2005