ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

Task Scheduling in Grid Computing Systems Constrained by Resource Availability

Chitra, Dr.Prashanth C.S.R

Abstract*— In this paper, a survey of some of the approaches to task scheduling in grid computing systems based on resource availability is discussed. To improve task scheduling, resource availability can be predicted by multistate availability predictors. Execution time of each application is modeled by a random variable. Each application is associated with a set of probability mass function (pmf), one pmf for each machine in the Heterogeneous computing suite, describing the probability of all possible execution times for that application. Stochastic completion times require the summation of random variables. The MaxRobust heuristic applies a simple greedy approach to maximizing the stochastic robustness of the resource allocation. A 3 state Markovian model is used for scheduling iterative master worker applications on processors that can fail or be temporarily reclaimed. Resource selection involves several factors such as resource access policies, status of execution, and computation capability of resource provider. The two main approaches in resource selection: centralized matchmaking-based approach and local knowledge-based approach are discussed.

***Index Terms*—MaxRobust Heuristic, Multi-state Resource Availability Prediction, Probability Mass Function, Stochastic Completion Time.**

I. INTRODUCTION

Effective task scheduling is a key concern for the execution of performance driven Grid applications. Resource availability is changing dynamically in time mainly due to load of the system, available network bandwidth and also due to other events such as resource or network failure or recovery. Dynamic resource availability is critical to achieve better system performance. There must be a mechanism to evaluate and manage the availability levels of grid resources while scheduling the tasks. Appropriate resource discovery and selection mechanism is important. Resource availability can be predicted by multistate availability predictors [1]. In heterogeneous computing environment, there is uncertainty in system parameters. Execution time of each application is modeled by a random variable. Each application is associated with a set of probability mass function, one pmf for each machine in the HC suite, describing the probability of all possible execution times for that application. Stochastic task completion times requires the summation of random variables. The MaxRobust heuristic [2] considers the stochastic robustness metric that gives the probability that a given resource allocation will complete all assigned applications by their deadline. A volatile processor can be in one of three states: UP (available), DOWN (crashed due to software or hardware fault) or RECLAIMED temporarily pre-empted by owner. A 3 state Markovian model[3] is for scheduling iterative master worker applications on processors that can fail or be temporarily reclaimed. Random heuristic selects a random processor picked up using a uniform probability distribution among the ones in the UP state to execute the next task. Resource selection involves several factors including the resource access policies, status of execution, and computational capability of resource provider. There are two main approaches in resource selection: centralized matchmaking-based approach and local knowledge-based approach[4]. Candidates are ranked based on their preparation times, and the jobs are divided and allocated to one or more selected workers.

II. LITERATURE REVIEW

A. *Scheduling on the Grid via Multi-State Resource Availability Prediction [1]*

Availability predictors, consider different periods of resource history, and use various strategies to make predictions about resource behavior. Schedulers must use these predictions to make scheduling decisions. Multi-state availability model specifies various availability states which is important because applications with varying characteristics will react differently to different types of unavailability. The five availability states are

1. Available: A machine in this state is currently running with network connectivity, has no user present, and a local CPU load below the CPU threshold, allowing the Grid to use the resource.

2. CPU Threshold Exceeded: A resource may transition to this state if the local CPU load becomes too high.

3. User Present: A resource may transition to the User Present state if the keyboard or mouse is touched.

4. Job Eviction: If the resource remains in either of the previous two suspension states too long, or if the machine is shut down, it transitions to the Job Eviction state (and can potentially checkpoint itself).

5. Unavailable: If a machine fails or becomes unreachable, it directly transitions to the Unavailable state (check- pointing is impossible).

If a job is suspended, and enters the Job Eviction state, it is called a Graceful transition because it may take a checkpoint and migrate to another resource. A transition directly to Unavailable state is Ungraceful transition. Grid resources will exhibit different characteristics in terms of how long they are in each availability state, how often they transition between the states, and the states to which they transition. Different applications will behave differently on different resources. There are different approaches to predict resources' transition into various unavailability states. Each of them analyzes some period of a resource's recent history to forecast future availability. The predictors take a resource, prediction time,

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

and estimated application duration i.e. prediction interval as input, and generate a vector of percentages corresponding to the predictor's forecast for the likelihood that the resource will next enter each of the availability states, within the prediction interval. The predictor also forecasts the likelihood that the resource will complete the interval without becoming unavailable. In considering recent history, different predictors consider either Transitional behavior by counting transitions between the states, or Durational behavior, by summing the total time spent in each state. Predictors also consider behavior within the most recent N hours, for many values of N, and at similar times of day over the past N days for various numbers of days. Three different ways of weighing unequally the past behavior within the analyzed history are considered. Different predictors weigh more heavily, behavior from the same days of the week, from the same times of day, and leading up to the current prediction. Each of these correlates to future availability behavior. Transitional Day-of-week Equal weight (TDE) predictor considers transitional behavior for the same prediction interval in the past 16 equally weighted days. Transitional Recent-hours Freshness (TRF) predictor considers transitional behavior from a resource's past 216 hours (9 days), weighing the most recent (i.e. freshest) behavior most heavily. Resource scoring schedulers give each resource a score based on factors that define the scheduler's placement policy. The resource with the highest score executes the job. Ties are broken arbitrarily. Resource scoring approaches utilize some or all of the factors such as CPU Speed (MIPS), Current Load (L) the machine utilization information at scheduling time, Completion Probability (P[COMPLETE]) which is the predicted probability of completing the projected job execution time without becoming unavailable and is one minus the sum of the probabilities of the three unavailability states, Ungraceful Eviction Probability (Pi[U_GRACE]) which is the predicted probability of exiting job execution directly to unavailable (with no chance for a checkpoint) and checkpoint ability i.e. whether the job could take an on demand checkpoint before being evicted from a machine (CKPT) or not (NON_CKPT). To choose from among resources for application execution, schedulers score each available resource according to the following expression:

$$RSi = (1 - W) \times Pi[COMPLETE] + W \times (MIPSi / MIPSmax) \times (1 - Li)$$

Pi[COMPLETE] is resource i's predicted probability of completing the job interval without failure, according to the TRF predictor, $MIPSi$ is the resource's processor speed, $MIPSmax$ is the highest processor speed of all resources (for normalization) and Li is the resource's current processor load.

Reliability influences completion probability Pi[COMPLETE], performance influences $(MIPSi / MIPSmax) \times (1 - Li)$, and the Tradeoff Weight (W) determines which more heavily influences the resource's overall score. The multi-state model and predictor combination captures user presence on a machine, distinguishes between graceful and ungraceful process eviction, and counts transitions differently. The Pseudo Optimal scheduler selects the available resource that will execute the application in the smallest execution time, without failure, based on omnipotent future knowledge of resource availability. When all machines would fail before completing the application, the Pseudo Optimal Scheduler chooses the fastest available resource in terms of MIPS speed.

B. *Stochastic Based Robust Dynamic Resource Allocation in a Heterogeneous Computing System [2]*

In heterogeneous, distributed computing systems there is uncertainty in system parameters. Robustness can be defined as the degree to which a system can function correctly in the presence of parameter values different from those assumed. The stochastic robustness metric gives the probability that a given resource allocation will complete all assigned applications by their deadline. Application execution times are a known source of uncertainty. The robustness of a resource allocation can be quantified as the joint probability that all applications will complete by their deadline, as predicted at a given point in time. Exact execution time of any given application is dependent on the details of the data, including the size and actual content, that is to be processed by the application and the machine that is to execute the application. Thus, the execution times of these applications may be highly variable and, are treated as random variables. Because the list of applications that may be requested is limited, the execution time random variable for each application is assumed to be well characterized. That is, we assume that a probability mass function (pmf) is available for each application execution time random variable on each machine (determined by historical, experimental, or analytical techniques). Stochastic Application Execution Time is the execution time of each application i, when executed alone on machine j (one of the M machines in the HC suite), is modeled as a random variable, denoted by ηij . The probability distribution describing the random variable ηij has been created from measurements of the response times of actual application executions. A typical method for creating such distributions relies on a histogram estimator that produces a discrete probability distribution known as a probability mass function pmf. Each application is associated with a set of pmfs, one pmf for each machine in the HC suite, describing the probability of all possible execution times for that application. Each request i is assigned an absolute deadline for completion, denoted by δi . At time-step t(k), when machine j will complete all of the requests that it has been assigned up to that point is to be predicted. If request i is the last entry in the input queue of machine j this corresponds to the completion time for request i. This requires a means of combining the execution times for all requests assigned to that machine. The estimated execution times for all requests assigned to machine j would be summed with the machine ready time to produce a completion time. The stochastic completion times requires the summation of random variables which is the convolution of their corresponding pmfs. Let MQ (t(k)) be the set of all requests that are either queued for execution or are currently executing on any of the M machines in the HC suite at time-step t(k). To determine the completion time for request i on machine j at time-step t(k), identify the subset of requests in MQ(t(k)) that were assigned to machine j

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

in advance of request i that have not yet completed execution, denoted MQij(t(k)). The execution time pmfs for the requests in MQij(t(k)) will be convolved with the execution time distribution for request i on machine j to produce the stochastic completion time pmf for request i on machine j. The execution time pmf for the currently executing request z on machine j requires some additional processing prior to its convolution with the pmfs of the queued requests to create a completion time pmf. For example, if z began execution at time-step t(h) (h < k), some of the impulse values of the pmf describing the completion time of z may be in the past. Therefore, to accurately describe the completion time of request z at time t(k) requires that these past impulses be removed from the pmf and the remaining distribution renormalized. After renormalization, the resulting distribution describes the completion time of z on machine j as predicted at time-step t(k).An operator GT(s, d) that accepts a scalar ***s*** and a pmf ***d*** as input is defined and returns a renormalized probability distribution where all impulse values of the returned distribution are greater than ***s***. The completion time pmf of the currently executing request on machine j is determined by applying the GT operator to its completion time pmf, using the current time-step t(k) = s. The resulting distribution is then convolved with the machine j execution time pmfs in MQij(t(k)) and the pmf of request i to produce the predicted completion time pmf for request i on machine j at the current time-step t(k). To calculate the robustness for each machine j the joint probability of completing all applications assigned to this machine is calculated by iteratively applying the product rule of probability to convert the joint probability of completing all applications by their deadline into a combination of simpler known probabilities. At any time-step t(k), each request in the resource allocation has been previously assigned to the input Queue of some machine j ($1 \leq j \leq M$). Because of the independence across machines in the heterogeneous suite, we can define the stochastic robustness $\rho^k$of a resource allocation at a given time-step t(k), as the product of the joint probability associated with each machine, i.e., the probability of all machines completing their application requests by their deadline is the product of each machine finishing its application requests by their deadline. A novel technique that attempts to greedily maximize the robustness of the system, referred to as the MaxRobust heuristic is proposed. The MaxRobust heuristic applies a simple greedy approach to maximizing the stochastic robustness of the resource allocation. Upon the arrival of a new request ri at time-step t(k), MaxRobust assigns request i to the machine that maximizes ρ^kat time-step t^k. That is, MaxRobust calculates the ρ^k value of request i as it were assigned to the end of the input queue of machine j, for each j = 1, . . . ,M, and selects the machine that maximizes ρ^k . In this way, MaxRobust greedily assigns applications to maximize the joint probability that all applications complete by their deadline at time-step t^k.

C. Scheduling Parallel Iterative Applications on Volatile Resources [3]

In the application model, the iterations of an iterative application involve the execution of a fixed number ***m*** of same size independent tasks. Each iteration is executed in a master-worker fashion, with a synchronization of all tasks at the end of the iteration. A processor is assigned one or more tasks during iteration. Each task needs some input data $Vdata$ from the master. Processor receives application program from master of size $Vprog$ which is same for all tasks and iterations. In the processor model, processor availability relies on a Markovian assumption for the temporal availability of processors. Processor volatility can be due to temporary interruptions. Processors are contributed by resource owners that can reclaim them at any time, without notice, and for arbitrary durations. A task running on a reclaimed processor is simply suspended. At a later date, when the processor is released by its owner, the task can be resumed without any wasted computation. At a given time, a volatile processor can be in one of three states: UP (available), DOWN (crashed due to software or hardware fault) or RECLAIMED (temporarily preempted by owner). Before going to the DOWN state, a processor may alternate between the UP and RECLAIMED states, the time needed by the processor to compute a given workload to completion is difficult to predict. For this it is assumed that state transitions obey a Markov process. A 3 state Markovian model is used for scheduling iterative master worker applications on processors that can fail or be temporarily reclaimed. When a UP or RECLAIMED processor becomes DOWN, it loses the application program, all the data for its assigned tasks, and all partially computed results. When it later becomes UP it has to acquire the program again before executing tasks. When a UP processor becomes RECLAIMED, its activities are suspended. When it becomes UP again, it can simply resume task computations and data transfers. It is assumed that execution occurs over a sequence of discrete time slots and that task computations and data transfers all require an integer number of time slots and that processor state changes occur at time-slot boundaries. The temporal availability of Pq is described by a vector Sq whose component Sq[t] ϵ {u,r,d} represents its state at time slot t. Here u corresponds to UP, r to RECLAIMED and d to DOWN. Processor Pq requires wq time slots of availability to compute a task. The problem is to maximize the number of successfully completed iterations of an application within some integral number of time slots N before a deadline. In the Scheduling Model, $config(t)$ denotes the set of processors enrolled for computing the ***m*** application tasks in an iteration, or configuration, at time ***t***. Enrolled processors work independently and execute their tasks sequentially. The configuration may be changed due to the possibility of a processor leaving the configuration. If a processor becomes down at time ***t***, the scheduler may simply use the remaining UP processors in $config(t)$ to complete the iteration, or enroll a new processor. Random heuristic selects a random processor picked up using a uniform probability distribution among the ones in the UP state to execute the next task. In addition, weights can be assigned to consider

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

reliability of processors. Processors are picked with a probability equal to their normalized weights.

D. Resource *Selection in Large-Scale Distributed System Using Dynamic Task Sharing [4]*

Resource selection involves several factors including the resource access policies, status of execution, and computational capability of resource provider. To select appropriate resource, it is important to consider the data accessibility, the status of execution and computational capability of the node. There are two main approaches in resource selection: centralized matchmaking-based approach and local knowledge-based approach. Resources can be integrated using Globus Toolkit or Condor. Under this framework, the respective capabilities and requirements of the providers and consumers are described in classified advertisements, which are pushed to a central matchmaker for matching. A scheme that estimates the execution capability, which observes the accessibility to local data to estimate the data download time and monitors the execution of current jobs to calculate the available time of resource is developed. Each node observes itself, and the previous observations of data download time are used to characterize the access behavior of the node. A dynamic policy is employed to select the resource. In the proposed approach the resource providers are ranked based on the defined preparation time, and then the jobs are allocated to one or more nodes. Here the objectives are maximizing the resource utilization and minimizing the job execution time. In centralized matchmaking, the provider has to register its information in the centralized matchmaker when it joins the system. The requester sends the request to the matchmaker first, and then the matchmaker retrieves the information of the providers to match the request. If there is a match with a node, the matchmaker returns the matched information to the requester, and then the requester will communicate with the node to get the service. In local knowledge-based matchmaking, a resource discovery algorithm is employed to determine the set of candidate nodes, while data locality is not considered in the decision. Once the requester selects a node, the job is transferred to the selected node which is called the worker. The worker then downloads the data object required for the job through the network and performs the computation. When the job execution is over, the worker returns the result to the requester. Three steps are involved in the process of job allocation in a provider: executing the accepted jobs, downloading the data object for the requested job, and execution of the job. The waiting time in the case where there are no accepted jobs is 0. The waiting time in the cab is being executed is the sum of the remaining execution time of the current work and the time that all the remaining jobs need and the waiting time in the case where worker is downloading data object for accepted job is the sum of the remaining download time and the execution time of the current work, and the time that all the remaining jobs need. $Tc = Tw + Td + Te$ where Tc is completion time, Tw is waiting time, Td is data download time, Te is execution time. The time containing Tw and Td is called the preparation time (Tp). When the preparation time is over, the requested job gets ready to be executed. In the proposed scheme two steps are used in the process of resource selection. The first step is ranking the candidates based on their preparation times, while the second step is dividing the job and allocating the divided jobs to one or more selected workers. The inputs to the algorithm are w which is the entire workload of the requested job, n which is the number of candidate workers, $T1$ and $c1$ are the preparation time and computational capability of the first worker, respectively. The output i is the number of workers used to undertake the job. After receiving the preparation times and execution times from the candidates, the requester first ranks these candidates by the preparation times, and then calculates the number of needed workers. Workload of the selected worker is calculated as

$$w_i = (t_n + T_n - T_i) \times c_i$$

For selected worker i, the allocated workload wi is calculated from its execution time and computational capability ci. Tn and Ti are the preparation times of the lastly selected worker and the i-th selected worker, respectively. tn is the period which lasts from the involvement of the last worker to the completion of the job, which is calculated as

$$t_n = \frac{w - \sum_{i=1}^{N-1}[(p_{i+1} - p_i) \times \sum_{j=0}^{i} c_j]}{\sum_{i=1}^{N} c_i}$$

The requester calculates which workers are needed and how much workload each selected worker should undertake, then sends the divided job to the workers. After completing the execution of the allocated job, each worker returns the result to the requester.

III. CONCLUSION

In the current meta-computing systems such as grids and clouds, processor availability cannot be taken for granted. Most of the computational resources attached to grids and clouds have local as well as global jobs assigned for execution. Often priority is assigned to local jobs over global jobs. The local operating system is expected to switch seamlessly between these jobs thus making the CPU bandwidth available for global jobs unpredictable. The objective of this work would be to factor the randomness of CPU bandwidth available for global jobs into scheduling decisions. So resource availability prediction, discovery, selection, allocation for tasks is very important for scheduling tasks in heterogeneous computing environment such as grids and clouds. Multistate resource availability prediction is used to schedule tasks on grids. Three different approaches such as Transitional Day-of-week Equal weight (TDE) predictor, Transitional Day-of-week Equal weight (TDE) predictor, Resource scoring schedulers are discussed. MaxRobust heuristic is proposed to dynamically allocate resources in a Heterogeneous Computing System such that stochastic robustness is maximized. Parallel iterative applications can be scheduled on grids using volatile resources which can be in one of the three states UP, DOWN or RECLAIMED. The centralized matchmaking-based approach and local knowledge-based approaches for Resource Selection in

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

Large-Scale Distributed System by dynamic task sharing are discussed.

REFERENCES

[1] Brent Rood and Michael J. Lewis," Scheduling on the Grid via Multi-State Resource Availability Prediction",9th Grid Computing Conference, 2008 IEEE.

[2] B. Rood and M. Lewis, "Multi-State Grid Resource Availability Characterization," International Conference on Grid Computing, pp. 42-49, 2007.

[3] B. Rood and M. Lewis, "Resource Availability Prediction for Improved Grid Scheduling," Binghamton University Computer Science Technical Report, April 2008.

[4] J. Mickens and B. Noble, "Exploiting Availability Prediction in Distributed Systems,", Symposium on Networked Systems Design & Implementation, pp. 73-86, 2006.

[5] X. Ren, S. Lee, R. Eigenmann and S. Bagchi, "Prediction of Resource Availability in Fine-Grained Cycle Sharing Systems and Empirical Evaluation," Journal of Grid Computing, Vol. 5,No. 2, pp. 173-195, 2007.

[6] Jay Smith, Edwin K. P. Chong, Anthony A. Maciejewski, and H. J. Siegel," Stochastic-Based Robust Dynamic Resource Allocation in a Heterogeneous Computing System",2009 International Conference on Parallel Processing.

[7] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 7, pp. 630–641, Jul. 2004.

[8] J. Smith, L. D. Brice˜no, A. A. Maciejewski, and H. J. Siegel, "Measuring the robustness of resource allocations in a stochastic dynamic environment," in Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007), Mar. 2007.

[9] M. A. Iverson, F. ¨Ozguner, and L. Potter, "Statistical prediction of task execution times through analytical benchmarking for scheduling in a heterogeneous environment," IEEE Transactions on Computers, vol. 48, no. 12, pp. 1374–1379, Dec. 1999.

[10] Henri Casanova, Fanny Dufosse', Yves Robert, Fre'de'ric Vivien," Scheduling Parallel Iterative Applications on Volatile Resources",2011 IEEE International Parallel and Distributed Processing Symposium.

[11] E. Byun, S. Choi, M. Baik, J. Gil, C. Park, and C. Hwang, "MJSA: Markov job scheduler based on availability in desktop grid computing environment," FGCS, vol. 23, no. 4, pp. 616–622, 2007.

[12] B. Hong and V. K. Prasanna, "Adaptive allocation of independent tasks to maximize throughput," IEEE TPDS, vol. 18, no. 10, pp. 1420–1435, 2007.

[13] Dae Gun Kim, Zhong Yuan Li, Sung Soo Moon, Hee Yong Youn," Resource Selection in Large-Scale Distributed System Using Dynamic Task Sharing",2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing.

[14] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," Proc. Int'l Symp. High Performance Distributed Computing , p. 140, 1998

[15] J. Kim, A. Chandra and J. B. Weissman, "Using Data Accessibility for Resource Selection in Large-Scale Distributed Systems", IEEE Transactions on parallel and distributed system, VOL. 20, NO. 6, JUNE 2009.

AUTHOR'S PROFILE

Mrs. Chitra S is currently Associate Professor, Department of Computer Science, and Maharani Lakshmi Ammanni College for Women, Malleswaram, and Bangalore, India. She obtained her BE in Electronics and Communication from Karnatak University, Dharwad, MCA from Madras University, Chennai and M.Phil in Computer Science from Madurai Kamaraj University, Madurai.

Dr. Prashanth C.S.R is currently the Professor and Head of the Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore-India. He obtained his B.E in Computer Science from Bangalore University, M.S in Computer Science from the University of Texas at Dallas, and Ph.D in Computer Science from Auburn University, USA. Dr. Prashanth has published extensively in the areas of High Performance Computing and Cloud Computing. He is also on the advisory board of several reputed international conferences and journals