Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Adaptive Scheduling using ETF Technique in Grid Computing

Navneet Kaur*
Scholar M.TECH (CSE)
India

Prabhjit Singh
AP, GIMET, Amritsar
India

Abstract— Grid Computing provides resource sharing on large scale and provides platform for development of various computer science engineering applications. Effective scheduling is a key concern for the execution of performance-driven grid applications such as workflows. In this paper, various strategies have been proposed, including static and dynamic scheduling Strategies. In this study, we perform optimization on existing AWS algorithm, thereby reducing the makespan of jobs and propose a new technique called AWT (Adaptive workflow technique) and algorithm called ETF (Earliest Time First). This approach schedule workflow tasks to the dynamic grid resources based on fact that job having early time (arrival time and service time) minimum will be executed firstly. The proposed earliest time first workflow scheduling (ETF) approach involves initial static scheduling, calculating scheduling heuristics, and calculating scheduling attributes with the aim to achieve the minimum execution time for workflow application. The propose approach differs from other techniques in literature to best of our knowledge.

Keywords— Adaptive workflow Scheduling, Dynamic Scheduling Strategies, earliest time first, Grid Computing, Resource Monitoring, Rescheduling.

I. INTRODUCTION

Recently, the rapid and advance development of networking technology and web has led to the possibilities of using large number of geographically distributed heterogeneous computing resources. These developments have led to the foundation of new paradigm known as Grid Computing [8]. Grid Computing is a type of parallel and distributed system that involves the integrated and shared use of resources a grid has to provide strong incentive for participating sites to join and stay in it. The workflow scheduling in grid is one of the key challenges which mainly deals with assigning workflow tasks to the available grid resources. In general, scheduling tasks on distributed grid resources belongs to a class of NP-hard problems [14]. So heuristics or approximations are the preferred options to obtain near optimal solutions. Many heuristics and experiments have been devoted to this problem as discussed in literature [3, 7] considering that accurate calculation is available for computation cost and communication cost of resources. However, in real actual environment, it is difficult to correctly predict the values due to heterogeneous and dynamic characteristics of the grid environment. The fluctuations in the resource availability (computing speed and links bandwidth) due to resource's local loads cause the original schedule to become sub optimal. Hence, it is a key challenge to maintain an application performance during its execution. In order to ensure high performance in dynamic grid environment, we considered the adaptive scheduling [9] where scheduling policy change dynamically as per the previous and current behaviour of the system to cope with the variations in the resource availability.

In this paper, we proposed an Earliest first time (ETF) algorithm for grid applications consisting of workflow tasks (dependent tasks) to meet the performance requirements. The procedure of proposed ETF differs from other approaches in literature by considering the dynamic availability of resources, both due to presence of computing nodes and communication links due to existence of local load or load by other users. The ETF algorithm is efficient one as it achieves minimum execution time of the application. The advantage of algorithm is, it is high performance reducing makespan and faster than the other heuristic algorithms. Computation cost is low [1].

The rest of the paper is organized as follow- Section 2 begins with a brief introduction to the traditional scheduling algorithm, followed by the various scheduling algorithm applied on grid computing. Section 3 describes in detail the ETF algorithm. Section 4 provides algorithm and detailed example .Simulation results on workflow are discussed in Section 5. Section 6 gives Conclusions and Future scope.

II. SCHEDULING METHOD FOR GRID

The problem of scheduling for workflow (DAG-based) has been discussed here. Most of work attempts to achieve minimum execution time (makespan) on heterogeneous grid environment.

In **2002** H.topcuoglu [1] presents Heterogeneous Earliest Finish Time (HEFT) and Critical –Path-on-a-processor (CPOP) the two most popular lists based experiment. In this, rank is computed using average execution time and average

communication time. It orders the tasks based on priorities and then assign them suitable resources to achieve high performance

IN **2004** R. Sakellario[4] paper presents a low-cost rescheduling policy SLACK which considers rescheduling at a few, carefully selected points during the execution. This policy achieves performance results, which are comparable with those achieved by a policy that dynamically attempts to reschedule before the execution of every task. It uses the concept of spare time, which does not have impact on the schedule length of the workflow. If execution time of task goes beyond the spare time then only rescheduling event is triggered.

In **2005** A. Mandal, K . Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, L. Johnsson, [5] author described new strategiesi.e MAX-, MIN, MIN-MIN for scheduling and executing workflow applications on grid resources using the GrADS [Ken Kennedy et al., 2002] infrastructure. The results of experiments show that strategy of performance model based, in-advance heuristic workflow scheduling results in 1.5 to 2.2 times better makespan than other existing scheduling strategies.

F. Berman, H. Casanova, A. Chien, K. Cooper [6] proposed recent extensions to the GrADS software framework in which a new approach to scheduling workflow computations, applied to a 3-D image reconstruction application and a simple stop/migrate/restart approach is applied to rescheduling Grid applications also a process-swapping approach to rescheduling, applied to an N-body simulation

L.F. Bittencourt, E.R.M. Madeira, F.R.L. Cicerre, L.E. Buzato [7] presents a new scheduling algorithm- PCH algorithm that uses a hybrid clustering-list-scheduling strategy. In this tasks with communication cost are grouped together and assigned to the same resource in a cluster. It aims to reduce the schedule length by reducing the communication Cost.

In **2007** Z. Yu, W. Shi [9] in this paper, author proposed and model evaluation metrics for the Grid Service performance. In addition also they proposed a low-overhead rescheduling method referred to as Adaptive List Scheduling for Service (ALSS) to adapt to the dynamic nature of a grid environment. ALSS provides stable performance for workflow applications, even in abnormal circumstances

In **2010** S.H. Chin, T. Suh, H.C. Yu [10] In this author proposed an adaptive rescheduling algorithm AHEFT based on static strategy. This paper propose a adaptive rescheduling concept, which allow the workflow planner works collaboratively with the run time executor and reschedule in a proactive way had the grid environment changes significantly.

In **2011** H.A. Sanjay, S.S. Vadhiyar[11] proposed three strategies or algorithms for deciding when and where to reschedule parallel applications that execute on multi-cluster Grids. Using large number of simulations, it was shown that the rescheduling plans developed by the algorithms can lead to large decrease in application execution times when compared to executions without rescheduling on dynamic Grid resources.

In **2012** A. Olteanu, F. Pop, C. Dobre, V. Cristea[12] proposes a generic rescheduling algorithm rescheduling i.e used for wide and large scale distributed systems (RE-LSDS) to support fault tolerance and resilience.. The system was evaluated and implemented in a real-world implementation for a Grid system. The proposed method supports fault tolerance and offers an improved mechanism for resource management.

In **2013** M. Rahman, R. Hassan, R. Ranjan, R. Buyya[13] This Paper introduced the dynamic critical path based workflow scheduling algorithm for grid namely DCP-G that provides efficient schedule in static environment. Additionally, it adapts to the dynamic grid environment, where resource information is updated and changed after fixed interval and rescheduling (Re-DCP-G) is performed if needed. It also describe the exactness of hybrid heuristic algorithm that combine the features of the adaptive scheduling technique with meta-heuristics for optimizing execution time and cost in dynamic cloud environment.

In **2014** E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, K. Wenger, Pegasus[15] describes the design, development and evolution of the Pegasus Workflow Management System. This performs mapping of abstract workflow descriptions onto distributed computing infrastructures in order to achieve reliable and scalable workflow execution.

III. ADAPTIVE WORKFLOW TECHNIQUE (AWT)

An AWT is a methodology for improving performance of parallel computing jobs by reducing makespan of jobs. The basic idea of algorithm is to use their arrival time and service time of jobs as a parameter in achieving efficient improvement rate of jobs and also their Mean response time . The algorithm is below:

Input: Tasks/ Job

Output: Makespan.

Step 1: Input tasks: A directed acyclic graph (DAG) is the standard way to represent a workflow. In this representation, parallel application/job consisting of bag of tasks with precedence constraints (dependency) and is represented as DAG [1]$G_{t=}$ (T, E) where:

1. T is the set of vertices representing n different tasks
2. E is the set of directed edges e_{ij}= (t_i, t_j) representing dependencies among the tasks ti and tj, indicating a task tj cannot start its execution before ti finishes and send all the required output data to task tj.
3. The weight w(ti) is assigned to task ti represents the size/ computing demand of ith task and expressed as number of instructions (MI) to be executed by the task .
4. Weight w (e_{ij}) assigned to edge e_{ij} represents the amount of data required to be transfer from task ti to tj if they are not executed on the same resource.

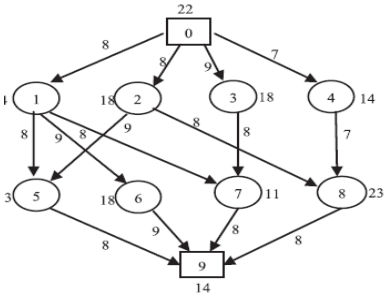

Fig 1. Shows an example of DAG [16]

Step2: Calculate scheduling heuristics: (Directed Acyclic Graph) is submitted by user. All incoming jobs get directed to Workflow Schedular Manager using **DAG** i.e representing tasks or workflow is given as input.
Then calculation of parameters like Communication time of task, Total task completion time, Task Individual time is calculated.
Step3: Calculate Scheduling Attributes: We have 3 processors P1 P2 P3.
It defines the task order sequence.
The procedure for static workflow task scheduling includes two steps: (i) assigning priorities to the tasks in order to ensure task dependency and
(ii) Mapping the tasks to the available resources in order to minimize execution time.
For Assigning priority or task ordering: To achieve this goal, we need to generate the ordered task sequence. Here, the ordered task sequence is generated based on b-level priority. The b-level [1] of a task is the length of longest path from the task to exit task. Thus the priority of the task is calculated.
Static Task Scheduling time (ms), B-level scheduling time (ms) and ALAP (at last as possible time is calculated) is calculated.
Step 4: Execution: After the tasks are input, and all scheduling attributes are calculated, now execute the ordered tasks using algorithm.
Here we have compared our proposed ETF algorithm with HEFT, AWS AND MAX-MIN algorithm and the corresponding graph is generated.
The results show that the proposed ETF achieved the minimum makespan in comparison to existing algorithms.
Step5: Communication to computation ratio (CCR): It indicates the characteristics of input workflow application. Higher value of CCR means data intensive application while lower value indicated computation intensive function. Size of communication data or message size can be determined from CCR values. It includes:
Calculation of Computational cost parameters (i) Estimated Start time (ii) Estimated Finish Time (iii) CCR (Cost to computation Ratio) and (iv) Wait time of jobs.
Step 6: Improvement rate (IR):
It specifies the performance improvement rate of ETF algorithm with respect to other algorithms and is calculated as the difference of the makespans over the makespan of ETF algorithm. Reduction in the makespan of ETF algorithm over other considered algorithms can be calculated by IR(%).
System IR(Improvement Rate) and MRT(Mean Response Time) is calculated as :

$$\mathbf{IR\%=}\frac{makespan(other)-makespan(ETF)}{makespan(ETF)}\times 100$$

$$\mathbf{MRT}=\sum\frac{Response\ time\ \ of\ jobs}{Total\ no\ of\ jobs}$$

IV. PROPOSED ALGORITHM WITH DETAILED EXAMPLE

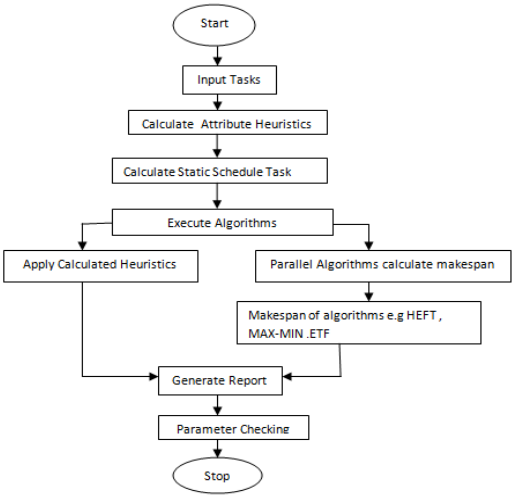

Fig 4.1.Flow Chart of ETF

Step 1: Tasks are given as input in form of directed acyclic graph (DAG),Then task are assigned operation and we also entering task using random matrix also we are considering test suit (TC1) consisting of 10 jobs.

Step2: Calculate Approximate scheduling heuristics: Here calculation of important parameters like Communication time of task, Total task completion time, Task Individual time is done.

Table 4.1 Showing value of task completion time, individual task time and communication time.

TASK NO	TASK COMPLETION TIME (MS)	INDIVIDUAL TASK TIME (MS)	COMMUNICATION TIME (MS)
TASK 1	519	519	0
TASK 2	995	476	2048
TASK 3	1478	482	2048
TASK 4	1953	474	2048
TASK 5	2429	475	2048
TASK 6	2905	474	2048
TASK 7	3384	477	2048
TASK 8	3831	505	2048
TASK 9	4371	478	2048
TASK 10	4843	472	2048

Step3: Calculate Scheduling Attributes: We have 3 processors P1 P2 P3.

Here firstly Static scheduling of jobs is done. Then scheduling using b-level is the length of longest path from the task to exit task. Thus the priority of the task is calculated.

Like this, makespan (in ms) is calculated using Static scheduling, and then using b-level and then also ALAP (At last as possible time) is calculated.

Table 4.2 Static Scheduling, [b]-level, ALAP in (ms) are calculated.

TASK NO	STATIC LEVEL (MS)	BOTTOM LEVEL (MS)	AT LAST AS POSSIBLE TIME (MS)
TASK 1	492	3072	0
TASK 2	438	476	2596
TASK 3	443	482	2590
TASK4	436	474	2598
TASK 5	437	475	2597
TASK 6	436	474	2598
TASK 7	439	477	2595
TASK8	405	505	2597
TASK 9	440	478	2594
TASK 10	434	492	2600

Step 4: Execution: Here we have compared our proposed ETF algorithm with HEFT AND MAX-MIN algorithm and the corresponding graph is generated. Calculative value of HEFT= 3195 (ms), MAX-MIN = 2936(ms) and ETF =2672(ms).

The results show that the proposed ETF achieved the minimum makespan time in comparison to existing algorithms.

Step 5: Generate results: Computation Cost is calculated in this part. It includes calculation of parameters like Job Estimated Start time, Estimated Finish time, CCR (ratio of Communication to computation cost), Wait time of job.

Table 4.3 Calculation of computational cost parameters.

TASK NO	ESTIMATED START TIME (MS)	ESTIMATED FINISH TIME (MS)	CCR(MS)	WAIT TIME(MS)
TASK 1	0	519	0	0.00
TASK 2	519	995	4.303	472.29
TASK 3	995	1477	4.249	905.45
TASK4	1477	1951	4.321	1344.07
TASK 5	1951	2426	4.312	1775.41
TASK 6	2426	2900	4.321	2207.66
TASK 7	2900	3377	4.294	2639.00
TASK8	3377	3882	4.055	3073.07
TASK 9	3882	4360	4.285	3562.62
TASK 10	4360	4832	4.339	3967.60

Step 6: This step includes calculating Improvement Rate and Mean Response time of jobs. Following results are obtained.

Experimental value of IR= 0.35%

Mean Response time = 1896(ms)

V. EXPERIMENT RESULTS:

The algorithms are executed in framework tool. In Result analysis, performance evaluation of all approaches is evaluated based on calculation of minimum makespan.

***1. Effect of varying of size of input graph*:**

Graph show makespan calculated by each algorithm for different algorithm implemented in this paper. Early First algorithm showing much less makespan than other and it is much better than other algorithm. It is faster than HEFT and traditional algorithm. Max-min and AWS are showing better performance than HEFT. ETF provides superior performance to HEFT by almost 6% IR and 7% IR to MAX-MIN From the result, it is clear that ETF Algorithm is makespan reducing high performance algorithm than other heuristic algorithm.

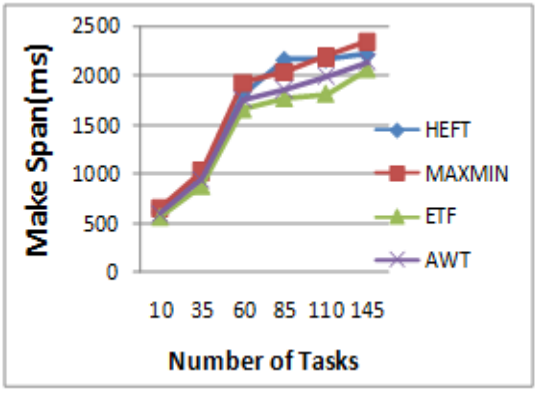

Fig 5.1: Makespan under different number of tasks

***2. Effect of varying of CCR Value*:** In this we are considering yhe effect of varying the CCR value of workflow application graph.The number of tasks are considered 35 by default .When CCR is less than 1,there is minimum improvement rate of 5% over HEFT and 3% over MAX-MIN .When the CCR value is greater than 1, there is minimum improvement rate ,IR of 10% over HEFT and 7% over MAX-MIN. Results indicate that the proposed algorithm performs efficiently under all CCR values.

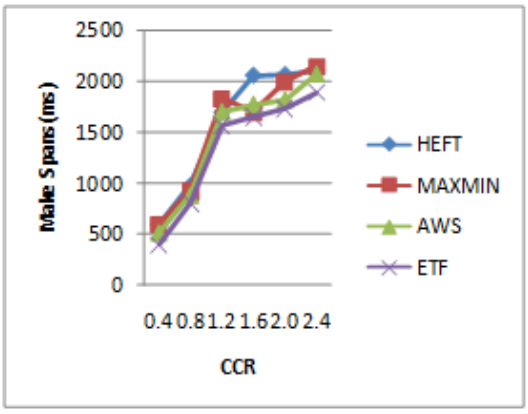

Fig 5.2: Makespan under Different CCR

3. Effect of improvement rate (IR):

Graph clearly shows that there is better improvement rate (IR) in AWT by varying the job size in comparison to existing AWS Algorithm.

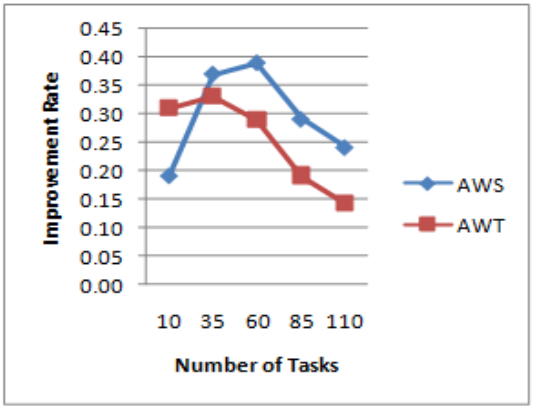

Fig 5.3: IR under different number of tasks.

4. Effect of Mean Response Time(MRT): Graph clearly shows that (MRT) of jobs submitted in AWT by varying the job size is less in comparison to MRT of job set submitted in existing AWS Algorithm.

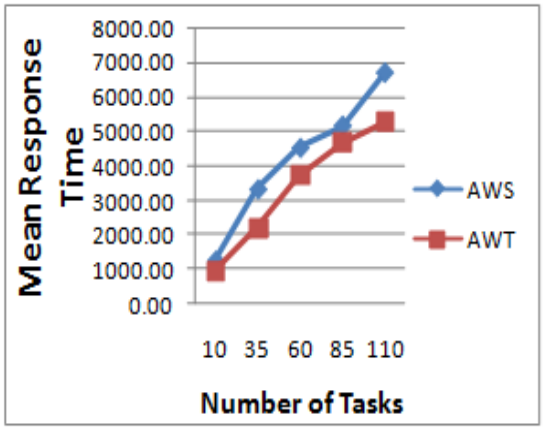

Fig 5.4 MRT under different number of jobs.

VI. CONCLUSION ANF FUTURE SCOPE

As our literature indicates that various scheduling techniques are available for calculating the makespan. Paper presents a high-performance and makespan reducing algorithm to find enhanced solutions for scheduling in grid computing systems. The algorithm uses arrival time and service time as a concept to fine-tune the solutions obtained by HEFT and MAX-MIN algorithm to further perk up the scheduling results in terms of makespan. As the simulation results show that early time first algorithm provides improved results than the traditional scheduling algorithms and also outperforms the other heuristic scheduling algorithms. The simulation results shows that offered algorithm faster than the other heuristic algorithms and computational cost is also low.

This is primarily different from other algorithm, which requiring a much longer computation time. Early time first can be applied to grid computing systems to enhance the performance of scheduling problems. The Researchers can implement or can develop the new approach of calculating makespan. In this paper various scheduling techniques have been described. So for the future work we can extend and explore the new scheduling techniques in order to provide high and stable performance to workflow application.

REFERENCES

[1] H. Topcuoglu, S. Hariri, M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", in: IEEE Transactions on Parallel and Distributed Systems, 13(3), 2002, pp. 260-274.

[2] E. Huedo, R.S. Montero, I.M. Llorente, "Experiences on adaptive grid scheduling of parameter sweep applications", in: Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2004, pp. 28-33.

[3] R. Bajaj, D.P. Agarwal, Improving scheduling of tasks in a heterogeneous environment, in: IEEE Transactions on Parallel and Distributed Systems, 15(2), 2004, pp. 107-118.

[4] R. Sakellariou, H. Zhao, A low-cost rescheduling policy for efficient mapping of workflows on grid systems, Scientific Programming 12 (4) (2004) 253-262.

[5] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, L. Johnsson, "Scheduling strategies for mapping application workflows onto the grid", in: In Proceedings. 14th IEEE International Symposium on High Performance Distributed Computing, 2005, pp. 125-134

[6] F. Berman, H. Casanova, A. Chien, K. Cooper, et al., New grid scheduling and rescheduling methods in the GrADS project, Int J Parallel Program 33 (2-3)(2005) 209-229

[7] L.F. Bittencourt, E.R.M. Madeira, F.R.L. Cicerre, L.E. Buzato, A path clustering heuristic for scheduling task graphs onto a grid (short paper), in: Proceedings of the 3rd ACM International Workshop on Middleware for Grid Computing, 2005. Grenoble, France.

[8] R. Buyya, S. Venugopal, "A Gentle Introduction to Grid Computing and Technologies", CSI Communications, July 2005.

[9] Z. Yu, W. Shi, "An adaptive rescheduling strategy for grid workflow applications", in: In IEEE International Parallel and Distributed Processing Symposium, 2007, IPDPS, 2007, pp. 1- 8.

[10] S.H. Chin, T. Suh, H.C. Yu, Adaptive service scheduling for workflow applications in service-oriented grid, J. Supercomputing 52 (3) (2010) 253-283.

[11] H.A. Sanjay, S.S. Vadhiyar, Strategies for rescheduling tightly-coupled parallel applications in multi-cluster grids, J. Grid Comp. 9 (3) (2011) 379-403

[12] A. Olteanu, F. Pop, C. Dobre, V. Cristea, "A dynamic rescheduling algorithm for resource management in large scale dependable distributed systems", Comp. Math. Appl. 63 (9) (2012) 1409-1423.

[13] M. Rahman, R. Hassan, R. Ranjan, R. Buyya, "Adaptive workflow scheduling for dynamic grid and cloud computing environment" , Conc. Comp. Prac. Exp. 25 (13) (2013) 1816-1842.

[14] M. Gareym, D. Johnson, Computers and Intractability: "A Guide to the Theory of NP-completeness", WH Freeman & Co., San Francisco, 1979.

[15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, K. Wenger Pegasus, a workflow management system for science automation, Future Generation Computer Systems (2014).

[16] Ritu Garg*, Awadhesh Kumar Singh"Adaptive workflow scheduling in grid computing based on dynamic resource availability" Engineering Science and Technology, an International Journal xxx (2015) 1- 14

