Load Balancing in Partner-Based Scheduling Algorithm for Grid Workflow

Muhammad Roman
Institute of Information Technology
Kohat University of Science and Technology
Kohat, Pakistan

Asad Habib
Institute of Information Technology
Kohat University of Science and Technology
Kohat, Pakistan

Jawad Ashraf
Institute of Information Technology
Kohat University of Science and Technology
Kohat, Pakistan

Gohar Ali
Information Systems and Technology
Sur University College
Sur, Sultanate of Oman

***Abstract*—Automated advance reservation has the potential to ensure a good scheduling solution in computational Grids. To improve global throughput of Grid system and enhance resource utilization, workload has to be distributed among the resources of the Grid evenly. This paper discusses the problem of load distribution and resource utilization in heterogeneous Grids in advance reservation environment. We have proposed an extension of Partner Based Dynamic Critical Path for Grids algorithm named Balanced Partner Based Dynamic Critical Path for Grids (B-PDCPG) that incorporates a hybrid and threshold based mechanism to achieve load balancing to an allowed value of variation in workload among the resources in Partner Based Dynamic Critical Path for Grids algorithm. The proposed load balancing technique uses Utilization Profiles to store the reservation details and check the loads from these profiles on each of the resources and links. The load is distributed among resources based on the processing element capacity and number of processing units on resources. The simulation results, using Gridsim simulation engine, show that the proposed technique has balanced the workload very effectively and has provided better utilization of resources while decreasing the workflow makespan.**

***Keywords*—Load Balancing; Advance Reservation; Resource Utilization; Workflow Scheduling; Job Distribution**

I. INTRODUCTION

Large-scale distributed and parallel computing has been changed by the growth of the Internet, powerful computing and network speed. With the coordination of distributed computing power, resources and applications, Grid computing has emerged as distributed computing platform. It utilizes the power of wide range of heterogeneous distributed resources for execution of compute- and data-intensive applications. It provides consistent, inexpensive, and pervasive access to geographically widely distributed resources to solve large scale scientific, engineering, and commerce problems. The resources joining the Grid are independent of each other in terms of performance, memory speed, and bandwidths. They are owned and administered by different providers. The motivation of Grid computing is to provide users and applications a seamless access to a number of high performance resources by creating illusion of a single system image [1]. User submitted applications are distributed among various Grid resources and executed in parallel. This makes the execution of applications efficient. In order to decrease the overall execution time of the application, effective and efficient load balancing algorithms are needed to be designed for Grid computing. Advance reservation (AR) provides facility to the user to reserve CPU and bandwidth for an application before the actual execution of the application [2]. AR jobs cause other non-AR jobs to wait as the resources are reserved in advance. Also consecutive reservations may leave fragments between them which cannot be used by other jobs, leaving un-used empty time slots. This leads to the under utilization of Grid resources. We are trying to fit the jobs in such a way that it is balancing the load over resources by considering all resource's load, which results in reducing empty fragments between the consecutive reservations. Although a lot of work has been done in non-AR environment for load balancing, yet there is a little contribution by the researchers in providing a solution for load balancing in advance reservation environment [3]. The objective is to prevent the condition where some of the resources are heavily loaded by the tasks submitted by users and others are lightly loaded or not being utilized at all [4]. The advantages of implementing good load balancing policies are better utilization of resources, low rejection rate, minimized wait time, high performance, maximized throughput, and reduced cost of job execution. Some of them are good for the resource providers, in terms of, resource utilization and throughput while others are good from Grid user's perspective, in terms of, reduced cost and minimized completion time of application.

Large scale scientific applications, modeled as workflow and represented as Directed Acyclic Graph (DAG), are submitted to Grid by using Workflow Management System [5]. For efficient execution of workflow good scheduling heuristics are trivial. Partner Based Dynamic Critical Path for Grids (PDCPG) proposed in [6], is one of the proposed algorithms for advance reservation environment in Grid and is based on Dynamic Critical Path for Grids (DCPG) [7]. PDCPG tries to schedule partner jobs on same resource in

Sur University College, Sur, Sultanate of Oman (*sponsor*)

order to minimize the communication involved to transfer required files of the child jobs. This may lead the overall Grid to an un-balanced state in which some of the resources are highly utilized while others remain under-utilized. We have proposed a hybrid policy restricted load balancing technique named as B-PDCPG to solve the problem of load balancing in PDCPG.

The rest of the paper is organized as follows: Section 2 presents load balancing problem. Section 3 explains related work to solve load balancing problem. Section 4 defines keywords and explains system model. Section 5 explains proposed load balancing technique and section 6 discusses observations and results. The paper is concluded in Section 7 and future work is discussed.

II. LOAD BALANCING

Grid architecture involves a large number of geographically distributed worker nodes connected together to achieve a level of performance. However, increasing the number of worker nodes does not always guarantee increased level of computing power. The resources involved in the system must be used such that all of the resources are utilized appropriately. The unequal demands and heterogeneity of Grid resources leads to the problem of job distribution. Algorithms try to map jobs on resources such that all the resources are utilized equally and makespan of the applications is reduced. Workload is the amount of work to be done by resource which can be heavy, light or moderate. Load balancing is sometime confused with load sharing and load leveling. Load sharing confirms that there is no idle node when there is highly loaded node in the Grid. This is the most basic level of load distribution which only checks if there is a load available on a resource or not, i.e., it is viewed as a binary set. Load balancing is the finest form of load distribution which tries to achieve strictly balanced distribution of load among all of the resources of the system. Load leveling is in the middle of the two extremes of load distribution which seeks to avoid congestion on any of the resources in the system.

For defining load balancing algorithm we need to implement certain policies. Following are some of the policies of load balancing [8]:

A. Information Policy

Defines what information is required for load balancing algorithm at what time and from where. It decides the type of information that will be collected, based on which algorithm will take decision. The information may include both static, like number of CPUs, size of memory, and dynamic information, like current load on the resources, memory being utilized. It also decides when to collect and update this information, to understand the current status of the system. Updating this information very frequently will lead to communication overhead. Normally a special information agent does this job either after a specific period of time or when an event is triggered [9]. Agents may directly or indirectly collect this information from the working nodes of the system. The effectiveness of load balancing algorithm is very much associated with how the load information is gathered. Simple load index calculates load based on one metric like CPU load, bandwidth utilization, or disk storage, while complex load index combines more than one metrics to aggregate them to single load information. The information gathered is then exchanged periodically, on demand, or when the state of a node has been changed [9].

B. Transfer type Policy

Determines when to start load balancing and transfer of load from a sender to receiver of the load. It performs the classification of resources as source or receiver based on the information gathered from information policy and the current status of the working nodes based on a predefined threshold value [10]. The resource which is heavily loaded and is going to transfer load to another resource is called sender and the resource which is going to receive load from other resource is called receiver.

C. Selection Policy

Decides which of the processes should be transferred from overloaded nodes (source) to the idle nodes (receiver). The criteria of selection may be defined at sender or receiver nodes based on the type of initiation policy. In sender-initiated approach, the sender decides which tasks should be selected for transfer. In receiver-initiated approach, the receiver defines the selection policy in terms of what type of, how big, and how many. Normally the criteria is based on; shortest remaining time, longest remaining time, First-in-First-Out (FIFO), Last-in-First-Out (LIFO), or a process is randomly selected [8].

D. Location Policy

The responsibility of location policy is to find a suitable partner node for the heavily loaded node for transferring some of its load to it. This policy works on the basis of information collected in information policy. In sender-initiated approach, heavily loaded node tries to find a lightly loaded node to share the load with it. In receiver-initiated approach lightly loaded node searches for a node which is heavily loaded. As the load information can be centralized or distributed, location selected decision can also be central or distributed. In case of distributed, each node tries to find a light resource in its neighbors and from the partial information saved at it. A node which has light load on it may not always be selected by location decision, other properties such as communication link and processing power are also consider for selecting it. Ultimately, the goal is to reschedule a job, which is already scheduled on a highly loaded resource, on a resource after which it will be executed earlier and will not delay other jobs or increase the makespan of the application.

Load balancing problem is described by different terminologies by many literatures which gives different meanings to the same problem in different situations. In [8] a detailed hierarchical model of load balancing is defined, as shown in Fig. 1.

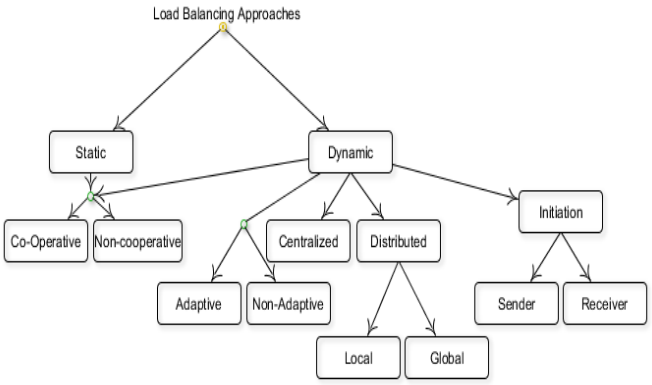

Fig. 1. Hierarchical Taxonomy for Load Balancing Approaches

A. Static vs. Dynamic Load Balancing

Static load balancing is based on the static information such as CPU capacity, memory size, link speed, and etc., whereas dynamic load balancing is based on the current status of the system and working nodes [11]. Static load balancing is good as it does not require information to be collected at all the time but this may lead to low utilization rate [12]. Dynamic load balancing algorithm may change the distribution of tasks among resources at runtime. This is similar to job re-scheduling, where the scheduled jobs are rescheduled after some changes are encountered in the system, like resource failure. In dynamic load balancing, if a resource is assigned large number of tasks and there are some lightly loaded resources available in the system, it will transfer some jobs to the resources which are lightly loaded or idle, selected on the bases of selection policy. A question arises, to which resource it has to be transferred and who will decide when to transfer them. There are two approaches in common: sender-initiated and receiver-initiated. In sender-initiated, a heavily loaded node requests to transfer some of its load to an idle or lightly loaded resource. In receiver-initiated, an idle resource initiates the process of transfer [13]. In some cases a combination of these two is used in which both sender and receiver can initiate the process. The main problem of the dynamic load balancing is how to distribute work among the resources as the resources are continuously changing their status, especially in case of heterogeneous system like Grid where the number of resources and bandwidth changes more frequently.

B. Centralized vs. Distributed

The load balancing decision may be taken at a centralized location or this may be taken at various levels of the system as in [14] and [3] where Grid is represented as a tree having four different levels. Centralized algorithm is easy to implement but has the problem of scalability, fault tolerance, and also becomes a bottleneck as all the decisions are to be made at one central location. In distributed strategy the tasks are distributed among different locations, where each location takes decision of its own region and sometime communicates with each other to take a global decision. This approach is difficult to implement but is highly scalable, fault tolerant and the load is divided which prevents them from becoming a bottleneck.

C. Adaptive vs. Non-Adaptive

In adaptive techniques the decision is based on the past and current system behavior and is based on the previously taken decisions and the changes in the environment. Confusion may arise as to differentiate between dynamic and adaptive techniques. The dynamic load balancing technique keeps into account the current status of the system at that time. On the other hand, adaptive technique considers the environmental stimuli in taking load balancing decision.

D. Local vs. Global

In local load balancing, each processing element gets information from its neighbor nodes and makes the decision based on this locally gathered information. In global load balancing, information from all or part of the system is collected to take the balancing decision. This may require a lot of information exchange.

E. Initialization

The task migration can by initialized by the sender, receiver or both (symmetric). In sender-initiated model, the resource which is highly overloaded transfers some of its selected processes to an idle or under loaded resource. In receiver-initiated model, the under loaded resource requests the system that it has the capacity to engage some more jobs. The probability of finding a lightly loaded resource is high in case of a lightly loaded system, therefore, sender-initiated algorithms works better in lightly loaded systems. In contrast, receiver-initiated algorithms are good in case of heavily loaded systems, as the probability of finding a highly loaded node is higher (heavily loaded nodes are more than lightly loaded nodes) [10]. A combination of these two policies is used in symmetrically initiated approach. The nodes behave intelligently by working as sender-initiated in low system load and receiver-initiated in highly loaded system.

F. Co-operative vs. Non-cooperative

Both in static as well as dynamic load balancing, nodes may or may not work together in taking balancing decision. In co-operative mode the nodes work together in order to make a decision that is based on the collective objectives of the overall system. In non-cooperative mode, however, individual system works autonomously in taking decision keeping their own objectives in account, irrespective of the effects on the rest of the system.

III. Related Work

There are various ways to balance load over the resources connected to a Grid, mainly categorized as static and dynamic load balancing techniques. There are fewer studies on static approaches for Grid environment [15] due to the heterogeneity of the system. Dynamic load balancing is considered better for such systems, as it responds better to the changes in the system [16]. In dynamic load balancing, if a resource is assigned large number of tasks and there are some lightly loaded resources available in the system, it will transfer some jobs to the resources which are lightly loaded or idle, selected as per selection policy of the algorithm. The disadvantages of dynamic load balancing algorithms are clear; these policies are more complex and require a lot of communication to share

dynamic state information. A good dynamic load balancing algorithm tries to minimize this cost in order to decrease the overhead and yet achieve the best load balancing. Some of the algorithms use combination of these two in order to make scheduling decisions, called hybrid load balancing algorithms [17]. In [18], a hybrid technique is proposed based on table of effectiveness to keep information of each Grid resource. When a resource is requested for a job, dispatcher selects it based on the information in the table of effectiveness and maps the job to the selected resource. If the resource is overloaded, dispatcher updates the table of effectiveness accordingly. In [9], the table of effectiveness is updated after a fix interval; consequently the execution of the task is not delayed. This algorithm keeps only static information of nodes, like remaining capacity of CPU and remaining memory. Reference [19] also updates the table on timely bases, the difference is that it stores dynamic load information rather than static load information.

Some other algorithms are developed based on fuzzy algorithms which are easy to implement and provide fast response time with good load balancing results [20]. Fuzzy algorithms are rule based algorithms and use knowledge of experts in creation of rules for a specific domain. Genetic based algorithms are also used for load balancing which performs the mapping of jobs to nodes by genetic operators which include three operators of reproduction, exchange and mutation [21]. Reference [22] combines the features of genetic algorithm, simulated annealing, and clonal selection algorithm and introduces genetic clonal annealing algorithm to task scheduling problem. Hierarchical load balancing algorithms try to achieve load balancing at different levels of Grid while representing the Grid as a tree model [14]. Agent based algorithms try to achieve load balancing by using agents working together to find load balancing [23]. Policy based approaches try to get an optimized scheduling under the constraint of load on a resource policy [24].

Our objective is to propose a load balancing mechanism in advance reservation environment. Partner Based Dynamic Critical Path for Grid (PDCPG), proposed in [6], is based on Dynamic Critical Path for Grid (DCPG) for AR environment. For job selection, it uses same technique as of DCPG. For resource mapping, it gets all the scheduled partner jobs of the selected most critical job in reverse order of their criticality. It calculates the completion time of the selected job on the resource on which its partner is scheduled. If the completion time is less than combined execution time and data transfer time of a partner job minus data transfer time of the selected job on that resource, then the resource is selected. Otherwise the next partner job is considered. In case, there was no partner job of a job or none of them was already scheduled on a resource, then a resource that gives minimum completion time is selected. In this way, PDCPG tries to minimize the communication overhead involved in transferring the required data files from parent jobs to the child jobs in a workflow. It encourages majority of the jobs to get scheduled on a single most efficient resource. This may lead to the problem of un-equal load on resources; some of the resources get highly loaded while the rest of them are under-loaded. We tried to balance the load in PDCPG by introducing a policy based technique.

The main contribution of this paper is to design a policy to get an optimized job scheduling solution under the constraint of resource usage policy. We adopt some ideas from DCPG and PDCPG to design our algorithm in AR environment for Grid.

IV. Proposed Load Balancing Algorithm

Before going into the detailed discussion of the problem and proposed solution, we explain the workflow models, resource model and other related terminology used in the proposed work.

DAG Workflow: Large business and scientific applications are submitted to Grid in the form of workflows, usually represented by DAG,G=(V,E), where V is the set of *v* nodes representing jobs and E is the set of *e* edges/lines connecting two nodes; parent (predecessor) and child(successor). The order in which parent and child nodes are connected shows the interdependency of the jobs. Edge(i,j) $\in$E shows that job *j* is dependent on job *i*, therefore, job *i* must be completed before scheduling job *j*. A job is said to be dependent on another job if it uses a file generated after execution of that job. The entry job is the only job which takes as input an already existing file, so having no incoming edge, while the rest of the jobs get the files as inputs which are generated by other jobs. F= {f_1, f_2, f_3, ….., f_g} is the set of files which are submitted as required files to jobs and generated as result of execution of jobs in the workflow. DS_{f_i} denotes the size of a file f_i in bits, where 1 ≤ i ≤ g. $PTR_i = \{ ptr_1, ptr_2, \dots, ptr_p \}$ is the set of partner jobs of job *i*, where jobs are said to be partners if they have at least one common direct child job. Job computation length is described in Million Instructions (MI) and resource speed is described in Million Instructions per Second (MIPs). Each workflow has a single entry job and single exit job; in case there are more than one entry jobs, a 0 MI length job is added to the start and end of the workflow. The average of the ratio of the computation cost and communication cost of the jobs in a workflow is called granularity of a job, except for the exit job *snk* having no child jobs.

$$Granularity = \frac{1}{v-snk}\left(\sum_{i=0}^{v-snk} \frac{\text{expET}(j_i)}{\text{ADTT}_{j_i}}\right) \quad (1)$$

Grid Resources: All the resources in the proposed work support advance reservation and is represented as a set of $R = \{r_1, r_2, \dots, r_m\}$. Resource r has W_r CPUs of same speedofPS_r. The user who submits the workflow for execution is connected to r_0, where $r_0 \notin R$. Workflow is submitted to the system from this resource and the starting files are also stored on it. All the resources are connected by a set of links L={l_1,l_2,.....,l_n}, which support advance reservation. Capacity of the link l_i is measured in Mbps and denoted by C_{l_i}, whereas delay is measured in milliseconds and denoted by d_{l_i}. Our proposed work uses Optical Burst Switching (OBS) network architecture which is suitable for Grids supporting advance reservation [25]. In OBS, the files are transmitted in a single

burst from source to destination. Resources and links can be requested and reserved by a user in advance. The reservation details are stored in utilization profiles [26] stored at each resource and link, called UP_r and UP_l, respectively. The reservation request can be made for a CPU called r_{cpu} or for a bandwidth calledr_{bw}. The availability of resource and link is checked against the utilization profile which contains the information of number of reserved CPUs and reserved bandwidth at time *t*. Further details of how to maintain reservation details and processing reservation request can be found in [24].

We have proposed a technique that prevents PDCPG from taking the overall system into an un-balanced state. It is a policy based technique which tries to schedule jobs on resources while implementing a policy to stop a resource from being over-loaded. The job selection technique is the same as inherited by PDCPG from DCPG. The resource selection technique of PDCPG has been modified to balance the load on the computing resources of the Grid. For implementing information policy; all the reservations and current load information of the system is kept in utilization profiles of the computing resources and communication links. At the time of resource selection this information is used to check the percent load on each of the resource, calculated as:

$$Load_r = \frac{\sum_{t=0}^{Z_r} UP_r\,(t) * PS_r}{Z_r * PE_r * PS_r} * 100 \qquad (2)$$

where Z_r is the total size of UP_r. $Load_r$ is the percent load on the resource *r*. UP_r (*t*) is the number of CPUs already reserved at time *t* in the dimension Z_r. PE_r is the number of processing elements on resource r. PS_r in the numerator and denominator can be cancelled out, but they are shown in the calculation for better understanding. For selecting a resource R_i for a job *j*, it is checked if some of the partner jobs of job *j* PTR_j have already been scheduled. The resource, on which job *j*'s most critical job partner is scheduled, is selected. The selected job is mapped onto this resource if the following condition [18] is met:

$$expCT_j^{r_{ptr_i}} \leq CT_{ptr_i} + ADTT_{ptr_i} - ADTT_j \qquad (3)$$

where $expCT_j^{r_{ptr_i}}$ is the expected completion time of the job *j* on the resource on which i^{th} partner job of *j* is scheduled. CT_{ptr_i} is the completion time of i^{th} partner job on that resource, $ADTT_{ptr_i}$being the communication time required to transfer the output file generated by the i^{th} partner job of *j* to the common child job and $ADTT_j$ is the communication time required to transfer the output file from the job *j* to the common child job [27].

If the condition is not satisfied, the next most critical partner job is considered. By doing so, the communication time will decrease for the most critical child job. However, this may increase the probability of scheduling more jobs on single resource, the resource on which the very first partner job was scheduled. This shall un-balance the load in Grid resources and some of the resource will get overloaded while others lightly loaded or not being used at all. To prevent this state, the resources selected after satisfying condition in (3) is checked whether its load is going beyond a predefined allowed variance from the load of other resources in the Grid. To check this, percent load on the selected resource $Load_r$ is calculated as in (2). Then standard deviation of the loads on resources SD_{Load_R} is determined, excluding the resource on which load is begin checked. Based on this data the resource is selected if the following condition is satisfied:

$$\left(Load_r - \frac{\sum_{i=0}^{m-r} Load_i}{m-1}\right) - SD_{Load_{R-r}} \leq SD_{Load_{allowed}} \qquad (4)$$

Where $SD_{Load_{R-1}}$ is the variation of loads on resources from the mean load of all the resources, excluding the load on the resource itself, calculated as standard deviation. $SD_{Load_{allowed}}$ is the allowed threshold value of standard deviation, provided by the algorithm as input parameter. The allowed threshold controls the variation in the loads in Grid system. Keeping the threshold value too small ensures that all the resources are strictly balanced and before assigning any extra load to a resource the rest of the resources also gets similar load. But this may lead to selecting a resource which is not very fast and may delay the completion time of the job. Keeping the threshold value too large relaxes the policy and allows the resource selector to make decision without considering the load balancing. Setting a threshold value helps us in tuning the overall functionality of the algorithm as per the preferences and goals (better resource utilization and throughput) of the Grid owner.

If the condition is not satisfied for the given resource, the next partner will be checked to see if it satisfies both of the conditions. If no partner job was scheduled or none of the partner jobs satisfy the conditions, then the resource that gives earliest completion time is selected. In this way, resource utilization is improved and also in most of the cases makespan of the workflow is also reduced. Makespan is the difference between submission time of the workflow and completion time of the last job of the workflow. The complete process is described in Algorithm 1.

Algorithm 1: Pseudo-Code for Balanced-PDCPG

```
1  function SELECTRESOURCE(j)
2  PTR_j ← list of partner jobs of j
3  foreach partner ptr_i in PTR_j do
4
        r_ptr_i ←
resource r on which partner ptr_i is scheduled
5       if IsNotOverLoaded(r_ptr_i) then
6
        expCT_j^(r_ptr_i) ←
expected completion time of j on r_ptr_i
7               CT_ptr_i ← completion time of ptr_i on r_ptr_i
8               ADTT_ptr_i ← data transfer time of ptr_i
9
ADTT_j ← data transfer time of j when scheduled on r_ptr_i
10              if expCT_j^(r_ptr_i) ≤ CT_ptr_i + ADTT_ptr_i −
 ADTT_j then
11                      return r_ptr_i // selected this resource
12              end if
13      end if
14 end foreach
15
return resource r from R which gives minimum expCT
16 end function

17 function ISNOTOVERLOADED(r)
18 Load_r ← CHECKLOAD(r)
19 Load_mean ← (Σ_{i=0}^{m−r} Load_i) / (m−1)
20 Load_allowed ← allowed standard deviation
21 if (Load_r − Load_mean ) − SD_Load_{R−r} ≤
SD_Load_allowed then
22      return false // resources in not overloaded
23 else
24      return true // resource is overloaded
25 end if
26 end function
```

V. Evaluation

The proposed algorithm is compared with PDCPG and DCPG. The algorithms are compared in different working environments of workflow and loads. The comparison is based on utilization of resources and makespan of the workflows. This section explains the experimental environment followed by the results.

A. Environment

The simulation is carried out by using Gridsim [28] simulator in advance reservation environment under OBS network architecture. The underlying machine configurations are; 2.7 GHz (8 CPUs) Core i7 with 8 GB primary memory running Windows 7 Professional64 bit Operating System. The resources are connected with 100 Mbps link. For every reservation request the requested number of CPUs is one and requested bandwidth is 10 Mbps. The users connected to resource 0 submit a workflow with the objective of earliest completion time and the Grid trying to complete the job while utilizing all of the resources equally. Multiple users are connected to the Grid submitting workflows at the same time in order to check the load balancing in the Grid on the resources. At start, the utilization profiles are empty and there are no reservations in them, to check how each of the algorithms makes reservations in the utilization profile. As we are mainly interested to balance the resource load and increase throughput, we have provided enough bandwidth at each of the links.

B. Workflow

To compare the algorithms in different situations there are various workflows selected to be submitted to the system as application. E-protien has minimum number of 15 jobs, and is selected from real world applications [29]. Job60PSPLIB, Job90PSPLIB, and Job120PSPLIB having 60, 90, and 120 jobs, respectively, are taken from Project Scheduling Problem Library (PSPLIB) [30]. LIGO, Montage, SIPHT, and Cybershake [31] have 83, 25, 30, and 21 jobs respectively. For each of these workflows, jobs are created with different granularity settings. For every setting simulation is run 100 times and mean values are taken to remove the errors in individual simulation execution. During every execution, job lengths are randomly assigned ranging from 20,000 to 50,000 MI, based on the number of the jobs in a workflow. The output file size is given randomly according to the granularity. For low granularity the output file size ranges from 420 to 450 MB and for higher granularity it ranges from 80 to 120 MB. The number of users submitting jobs in each 100 rounds is kept the same and ranges from 5 to 45 users based on the number of jobs in the workflow. This is to keep the load in the reservation profile of the resources always from 60% to 95%. For small workflows like e-protein we have submitted jobs by 45 users to populate the utilization profiles with reservations in them and check how the load is balanced among all the resources of the Grid.

VI. Results

Our main objective is to observe the effectiveness of our load balancing policy and see the effects of this balancing technique on makespan of different workflows. To understand this, we have experimented our proposed technique on low granularity workflow followed by higher granularity workflows. We have studied the improvement in resource utilization and have also studied the effects of this on makespan of individual workflows, as listed above. As we are trying to balance the load, the balancing policy will disparate the idea of selecting a resource on which a partner job has already been scheduled. Apparently, this may increase the communication cost and increase the makespan of a workflow. In contrast, the results show that makespan has also been decreased in many cases by using our proposed balancing technique, because of better resource utilization and future optimal approach. The results contain the average of 100 simulations for each of the settings. For low granularity we have taken mean of 0.25, 0.5, and 0.75 granularities, calculated as in (1). For each of the granularities we have simulated 100 times and then taken mean of them. This means

that we have simulated 300 times for low granularity for each of the algorithms, and similarly for high granularity as well. For understanding the effects of our proposed balancing technique, we have embedded our load balancing policy in DCPG and PDCPG named Balanced-DCPG (B-DCPG) and Balanced-PDCPG (B-PDCPG), respectively.

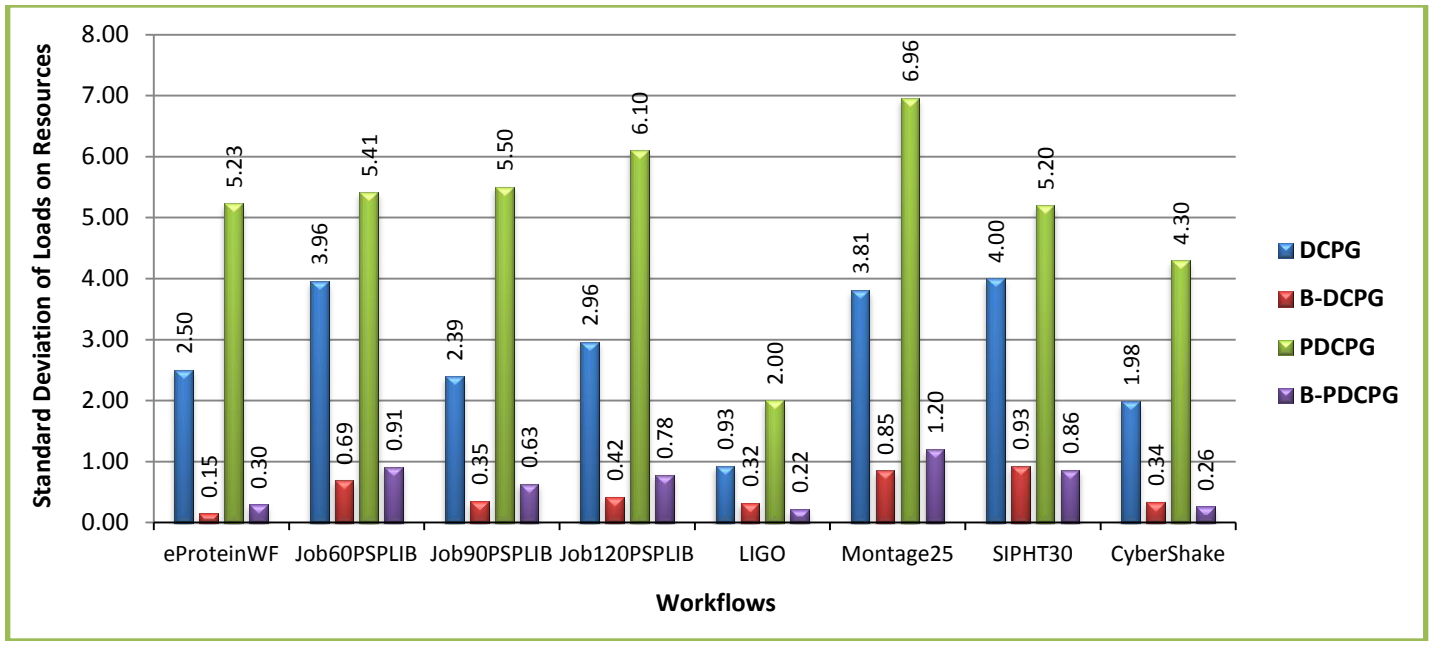

Fig. 2. Standard Deviation of Loads on Resources

Fig 2, shows a comparison of how different algorithms have distributed load on resources of the Grid. The values shown are the standard deviations of the loads. We have compared the variation of loads on resources of the Grid in DCPG to that of B-DCPG and similarly PDCPG to that of B-PDCPG. It can be observed that the load on resources has been distributed equally with negligible difference in balanced version of DCPG and PDCPG. The results are very similar in low and high granularities; therefore, we have combined them in a single chart. It can be seen that PDCPG has a very large variation in load distribution, which shows that the load is not equally distributed among the resources of the Grid system. In other words, PDCPG does not guarantee a good utilization of resources.

The balancing technique has achieved significant improvement in utilization of resources. As discussed above, the allowed variance can be controlled by setting the allowed threshold in balancing mechanism. Here we set a very small value of allowed threshold value, to check how much we can balance the loads on Grid resources.

To check the effects of load balancing on execution time required for a single application, we have also measured the effects of this load balancing technique on individual makespans of the user applications.

TABLE I. AVERAGE MAKESPAN OF THE WORKFLOWS IN LOW GRANULARITY

WF	MAKESPANS				PERCENT IMPROVEMENT			
	DCPG	*B-DCPG*	*PDCPG*	*B-PCPG*	*B-DCPG over DCPG*	*B-PDCPG over PDCPG*	*PDCPG over DCPG*	*B-PCDPG over DCPG*
eProteinWF	631.67	630.67	622.67	623.00	0.16	-0.05	1.45	1.39
Job60PSPLIB	976.67	962.00	957.33	955.33	1.52	0.21	2.02	2.23
Job90PSPLIB	1364.00	1363.33	1343.00	1345.33	0.05	-0.17	1.56	1.39
Job120PSPLIB	1114.00	1114.00	1094.67	1093.67	0	0.09	1.77	1.86
LIGO	1175.33	1177.33	1176.00	1172.67	-0.17	0.28	-0.06	0.23
Montage25	480.57	480.30	470.87	470.33	0.06	0.11	2.06	2.18
SIPHT30	424.33	422.33	420.67	419.00	0.47	0.4	0.87	1.27
CyberShake	373.67	374.67	372.67	370.67	-0.27	0.54	0.27	0.81

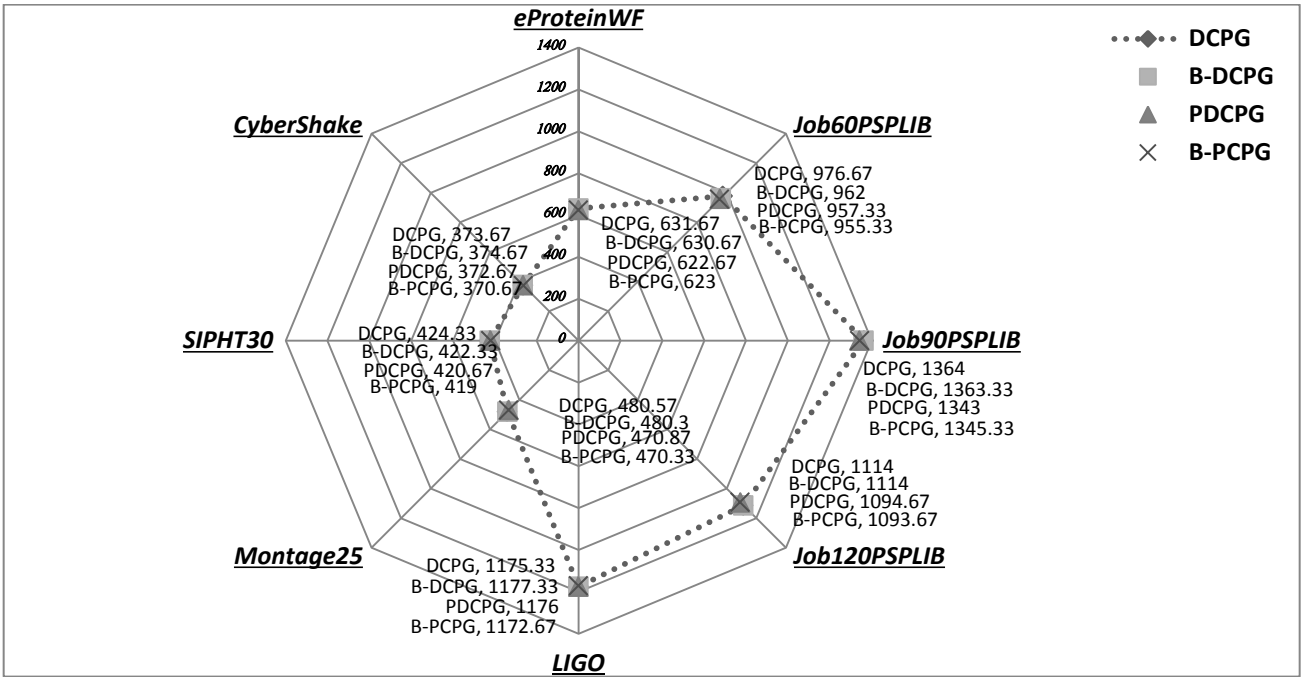

Fig. 3. Average Makespan of the Workflows in Low Granularity

Table 1 and Fig. 3, explains the effects of load balancing on makespans of workflows in low granularity jobs. It can be seen that the overall execution time required to complete an application is unchanged in most of the cases and has even been reduced in some cases, especially in the workflows which have more breadth rather than depth. Although the completion time of individual jobs may get delayed due to increased communication cost in our proposed balancing technique, it has no effect on the overall execution time of a workflow. Table 1 and Fig. 3, also shows the percent improvements in the makespans of the workflows. We can see that there are some cases in which our balanced approached has not achieved any improvement but that is not our goal at all. Our objective is not to achieve improvements in makespan reduction, but to show that our balancing technique has no effect on individual makespan of the workflows.

TABLE II. AVERAGE MAKESPAN OF THE WORKFLOWS IN HIGH GRANULARITY

WF	MAKESPAN				PERCENT IMPROVEMENT			
	DCPG	*B-DCPG*	*PDCPG*	*B-PCPG*	*B-DCPG over DCPG*	*B-PDCPG over PDCPG*	*PDCPG over DCPG*	*B-PDCPG over DCPG*
eProteinWF	637.67	637.67	636.67	636.67	0	0	0.16	0.16
Job60PSPLIB	1324.33	1324.33	1322.00	1322.00	0	-0.05	0.18	0.13
Job90PSPLIB	1340.67	1337.67	1333.33	1333.33	0.22	-0.4	0.55	0.15
Job120PSPLIB	1575.00	1575.00	1566.67	1566.67	0	0.11	0.53	0.64
LIGO	1731.33	1731.00	1735.67	1735.67	0.02	0.08	-0.25	-0.17
Montage25	626.10	626.47	622.00	622.00	-0.06	0.02	0.66	0.68
SIPHT30	556.67	553.67	556.33	556.33	0.54	0.48	0.06	0.54
CyberShake	485.00	484.67	484.33	484.33	0.07	0.07	0.14	0.21

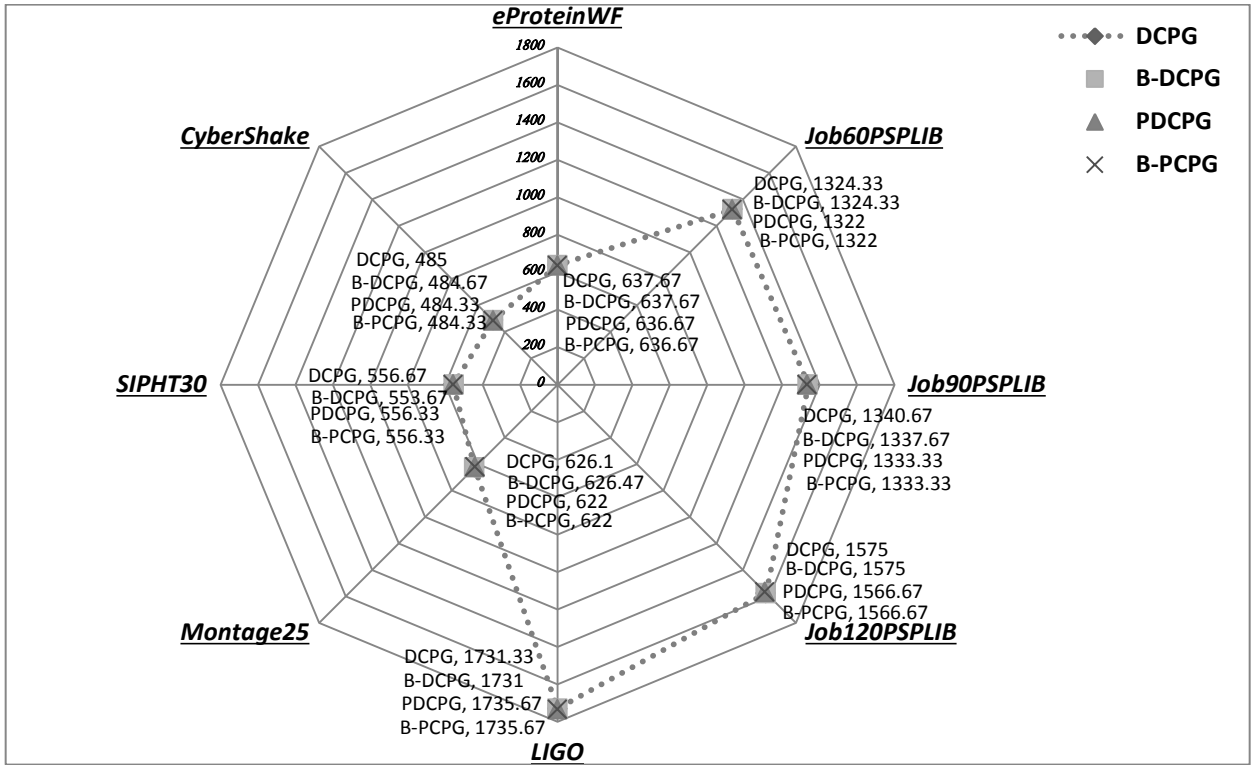

Fig. 4. Average Makespan of the Workflows in High Granularity

Table 2 and Fig. 4 below, explains the effects of load balancing on makespans of workflows in high granularity jobs. For high granularity, we have taken mean of simulations with 1.0, 1.25, 1.50, and 1.75 granularities and combined them together. The results show that, like PDCPG, B-PDCPG is also providing very small percentage improvement over DCPG. The reason is that there is not much communication involved in high granularity jobs.

It has been observed that the standard deviation of the makespans also reduces by balancing the loads on resources. What this means, is that PDCPG completes some of the workflows in a very short time by scheduling most of its jobs (especially partner jobs) on the fastest resource, while others are scheduled on relatively slower resources resulting in large completion time. By balancing the workload on resources the makespan is not that much different from each other, as the balancing policy stops a workflow from reserving all of the slots on a high speed resource.

VII. Conclusion and Future Work

In this study, we have addressed the problem of load balancing in advance reservation environment. We have proposed a policy based dynamic and centralized load balancing technique for balancing the loads on different resources of the Grid. We have observed that by trying to reduce communication, PDCPG leads the overall system to an un-balanced state where some of the resources are highly reserved in advance while the rest are rarely used or idle. This can result in resource under-utilization and increased rejection rate at some time. To solve this problem we have presented a technique based on a policy restriction while assigning job of a workflow to a resource. Simulation results shows that the proposed technique enhances the performance of PDCPG and increase utilization of resources and jobs throughput.

Future work will consider a distributed and adaptive technique that will implement the load balancing policy at different level of the Grids in a hierarchy while adapting itself to the changes. The implementation of proposed technique in non-advance reservation environment will also be considered.

Acknowledgement

The authors would like to thank Sur University College (Sur, Sultanate of Oman) for support and sponsorship of this research.

References

[1] B. Yagoubi and Y. Slimani, "Task load balancing strategy for grid computing," Journal of Computer Science, vol. 3, no. 3, pp. 186-194, 2007.

[2] M. Siddiqui, A. Villazon and T. Fahringer, "Grid capacity planning with negotiation-based advance reservation for optimized QoS," in Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006.

[3] S. F. El-Zoghdy, "A load balancing policy for heterogeneous computational grids," International Journal of Advanced Computer Science and Applications, vol. 2, no. 5, 2011.

[4] D. S. Gawande, R. C. Dharmik and C. Panse, "A load balancing in grid environment," International Journal of Engineering Research and Applications (IJERA), vol. 2, no. 2, pp. 445-450, 2012.

[5] V. Curcin and M. Ghanem, "Scientific workflow systems - can one size fit all?," in Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International, 2008.

[6] J. Ashraf and T. Erlebach, "A new resource mapping technique for grid workflows in advance reservation environments," in High Performance Computing and Simulation (HPCS), 2010 International Conference on, 2010.

[7] M. Rahman, S. Venugopal and R. Buyya, "A dynamic critical path algorithm for scheduling scientific workflow applications on global grids," in e-Science and Grid Computing, IEEE International Conference on, 2007.

[8] B. Yagoubi and Y. Slimani, "Load balancing strategy in grid environment," Journal of Information Technology and Applications, vol. 1, no. 4, pp. 285-296, 2007.

[9] L. Mohammad Khanli, "A new hybrid load balancing algorithm in grid computing systems," International Journal of Computer Science & Emerging Technologies, vol. 2, no. 5, 2011.

[10] L. M. Khanli, S. Razzaghzadeh and S. V. Zargari, "A new step toward load balancing based on competency rank and transitional phases in grid networks," Future Generation Computer Systems, vol. 28, no. 4, pp. 682-688, April 2012.

[11] Deepika, D. Wadhwa and N. Kumar, "Performance analysis of load balancing algorithms in distributed system," Advance in Electronic and Electric Engineering, vol. 4, no. 1, pp. 59-66, November 2014.

[12] B. Yagoubi and M. Meddeber, "Towards hybrid dependent tasks assignment for grid computing," Java in Academia and Research, iConcept Press.

[13] P. Bindu, R. Venkatesan and K. Ramalakshmi, "Perspective study on resource level load balancing in grid computing environments," in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011.

[14] J. C. Patni, M. S. Aswal, O. P. Pal and A. Gupta, "Load balancing strategies for grid computing," in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011.

[15] S. Penmatsa and A. T. Chronopoulos, "Cooperative load balancing for a network of heterogeneous computers," in Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, 2006.

[16] J. Balasangameshwara and N. Raju, "A hybrid policy for fault tolerant load balancing in grid computing environments," Journal of Network and Computer Applications, vol. 35, no. 1, pp. 412-422, 2012.

[17] Y. Li, Y. Yang, M. Ma and L. Zhou, "A hybrid load balancing strategy of sequential tasks for grid computing environments," Future Generation Computer Systems, vol. 25, no. 8, pp. 819-828, 2009.

[18] K.-Q. Yan, S.-C. Wang, C.-P. Chang and J. Lin, "A hybrid load balancing policy underlying grid computing environment," Computer Standards & Interfaces, vol. 29, no. 2, pp. 161-173, 2007.

[19] S. Zikos and H. D. Karatza, "Communication cost effective scheduling policies of nonclairvoyant jobs with load balancing in a grid," Journal of Systems and Software, vol. 82, no. 12, pp. 2103-2116, 2009.

[20] Y.-K. Kwok and L.-S. Cheung, "A new fuzzy-decision based load balancing system for distributed object computing," Journal of Parallel and Distributed Computing, vol. 64, no. 2, pp. 238-253, 2004.

[21] V. Di Martino and M. Mililotti, "Sub optimal scheduling in a grid using genetic algorithms," Parallel computing, vol. 30, no. 5, pp. 553-565, 2004.

[22] Z. Wenpeng and L. Hongzhao, "A load balancing method based on genetic clonal annealing strategy in grid environments," in Educational and Network Technology (ICENT), 2010 International Conference on, 2010.

[23] J. Cao, D. P. Spooner, S. A. Jarvis and G. R. Nudd, "Grid load balancing using intelligent agents," Future generation computer systems, vol. 21, no. 1, pp. 135-149, 2005.

[24] J. U. In, S. Lee, S. Rho and J. H. Park, "Policy-based scheduling and resource allocation for multimedia communication on grid computing environment," Systems Journal, IEEE, vol. 5, no. 4, pp. 451-459, 2011.

[25] E. M. Varvarigos, V. Sourlas and K. Christodoulopoulos, "Routing and scheduling connections in networks that support advance reservations," Computer Networks, vol. 52, no. 15, pp. 2988-3006, 2008.

[26] K. Christodoulopoulos, N. Doulamis and E. Varvarigos, "Joint communication and computation task scheduling in grids," in Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on, 2008.

[27] J. Ashraf, "Partner-based scheduling and routing for grid workflows," University of Leicester, 2012.

[28] A. Sulistio, C. S. Yeo and R. Buyya, "Visual modeler for grid modeling and simulation (gridsim) toolkit," in ICCS 03 Proceedings of the 2003 International Conference on Computational, Berlin, 2003.

[29] A. O'Brien, S. Newhouse and J. Darlington, "Mapping of scientific workflow within the e-protein project to distributed resources," in UK e-Science All Hands Meeting, 2004.

[30] F. Wittemann, "The Library PSBLIB," December 2015. [Online]. Available: http://www.om-db.wi.tum.de/psplib/library.html. [Accessed 5 January 2015].

[31] January 2015. [Online]. Available: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator. [Accessed 3 January 2015].