Journal of Computer Science 8 (7): 1108-1113, 2012
ISSN 1549-3636

Explicit Allocation Strategy with Deadline and Budget Constraint Algorithm in Bag of Tasks Grid

[1]M. Suresh Kumar and [2]T. Purusothaman
[1]Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College Coimbatore, India
[2]Department of Computer Science and Engineering,
Government College of Technology, Coimbatore, India

Abstract: Problem statement: This study is for effective scheduling of grid jobs based on economy for space shared resources in Bag of tasks grid. Grid Computing aims in combining the power of heterogeneous, geographically distributed, multi-domain computational resources to provide high performance or high throughput. **Approach:** Space shared resources are parallel supercomputers and clusters of workstations that provides a great amount of computational power. These resources require jobs to be specified formally in terms of the amount of time (tr) and number of processors (p) needed for execution. Bag-of-Tasks (BoT) is an application consists of several uniprocessor and independent tasks that have no inter-task communications or task-dependencies. BoT is highly suitable for execution in grids. It is capable of tolerating network delays or faults and does not require formal job submission. The Explicit allocation strategy assigns the formal job parameters (p, tr) to the job requests, minimizing the overhead on the grid users to provide a formal job specification. This strategy uses adaptive heuristics to determine the parameters based on certain heuristics, in order to improve throughput. In the proposed system, explicit allocation strategy combined with Deadline and Budget Constraint (DBC) Cost Time optimization algorithm performs effective scheduling of the jobs based on the user's quality of service (QoS) requirements such as deadline, budget and optimization strategy. **Results:** The cost-time optimization scheduling allocates the cheapest resources to ensure that the deadline can be met and computation is minimized. In case if there are two resources with the same cost, scheduling is done in any affordable resource so that the job gets executed as early as possible. **Conclusion:** The performance of this scheme against the existing system is evaluated using cost factor (C_{factor}) and speed up ratio ($T_{speedup}$) and this scheme is more effective than the existing system.

Key words: Grid computing, bag of tasks, explicit allocation, space-shared resource

INTRODUCTION

Grid computing is a method of computing in which very large problems are divided into small tasks that are distributed across a network for simultaneous processing. Due to the widespread use of grid in almost all fields, there is a need for effective utilization of the available computational power of all types of resources. Space-shared resources are parallel supercomputers and clusters of workstations with great amount of computational power. They are among the most powerful resources in a grid. Space-shared resources are used through a formal job submission to the resource scheduler specifying the number of processors needed and the amount of time these processors should be allocated to the incoming job.

Bag-of-Tasks (BoT) (Lee and Zomaya, 2007) application is a cluster of uniprocessor tasks that are independent of each other and do not communicate among themselves. In a grid, when a local user demand for a resource already in use by a grid user, the grid user job gets preempted and local user gains access to the resource. BoT applications are most suitable for execution in grids as they can be preempted and recovered easily from failures by executing the tasks. These applications do not need specifications such as maximum number of resources, or period of time needed for job execution to be provided by the user during job submission. Different load balancing techniques (Boukerram and Azzou, 2006; Buyya *et al*., 2002; Foster and Kesselman, 1997) were proposed. This paper proposes an effective algorithm for Bag of Tasks Grid.

Problem definition: The Explicit Allocation strategy aims to execute BoT applications in space shared resources in order to utilize its high computational power effectively.

Corresponding Author: Suresh Kumar, M., Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College Coimbatore, India

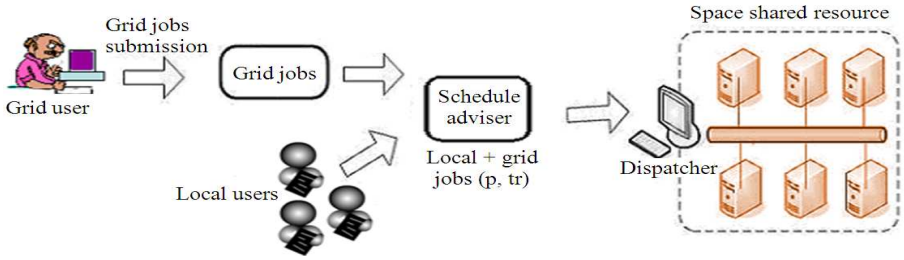

Fig. 1: Scheduling of BoT applications in Space shared resources

The problem lies in the job submission mechanism as space shared resources require formal job submission parameters (p, tr) whereas BoT does not include these specifications. So the Explicit Allocation strategy uses heuristics to generate the formal job parameters (p, tr) implicitly. This scheduling strategy combined with DBC Cost Time optimization algorithm makes effective utilization of resources based on the budget and deadline specifications of users.

The DBC cost-time optimization scheduling algorithm is based on the cost-optimization and Time optimization algorithm. The Cost-optimization algorithm performs scheduling by allocating the cheapest resources to ensure that the deadline can be met and the computational cost is minimized. If two resources are of same cost, then allocation is based on Time optimization (optimize the time without incurring additional processing expenses).

Assumptions in execution environment:

- A job can be scheduled anywhere in the grid environment for its execution
- Once the distribution of resources starts, the resources are locked for a local user's job execution and reclaimed after use
- The execution time of a job is calculated from the time the job starts its execution till the end of its execution.
- The network delay and communication time are not taken into consideration
- The time taken for inter-process communication of parallel applications is not included as BoT applications are used

Model of the proposed strategy:

- The grid environment consists of diverse machine types, disks/storage and networks. The resources in the grid environment are made of desktops, servers, clusters and multi-processor systems
- The CPU resource's computational capability is represented in the form of Million Instructions Per Second (MIPS). The grid resources are also accessible to outside users when they are idle
- As shown in Fig. 1, Grid and local jobs are submitted to the scheduler adviser. The schedule Adviser determines how jobs should be scheduled and how the resources should be utilized. It uses the Deadline Budget Constraint scheduling strategy with cost time optimization
- The jobs are sorted according to deadline and budget specifications and inserted into the scheduled-job queue
- The scheduled-job queue is then processed by the dispatcher which sends the jobs to the allocated resources for execution
- After the execution of all jobs, the results are recorded and analyzed

Explicit allocation strategy: The Explicit allocation strategy (Rose *et al*., 2008) enables grid users to achieve local user priority. This is obtained by allowing grid users to submit formal job request as any local user in the grid. The grid users unaware of the environment, cannot specify the job parameters (amount of time needed to execute the job, number of processors required) directly. To ease this job submission task, the strategy crafts the formal job request automatically based on the grid user job specification. For assigning the job parameters, the explicit strategy takes the following into consideration:

- The maximum allowed number of pending requests that grid broker can have on a space-shared resource scheduler (maxPendingRequests)
- The maximum allowed number of processors per request (maxProc)
- The maximum allowed amount of time requested per request (maxTr)
- The queue state

As the strategy uses adaptive heuristic, the execution time and number of processors needed is varied for each job. With an initial estimate of the parameter values, a set of job requests is considered and the throughput (i.e., number of jobs that can be executed from the given set) is calculated. If a new possible request could improve the throughput, a previous chosen request is discarded and the new request is inserted into the set of job requests. The chosen set is added to the unassigned-jobs list.

DBC with cost time optimization: This study combines the explicit allocation strategy with the Deadline and Budget Constraint (DBC) Scheduling with cost time optimization. The resources in the grid environment are ordered in terms of increasing the processing cost per million instructions. In case two resources have the same cost, then they are sorted according to the available processing time. The resources with same cost are considered as groups and these groups are sorted according to increasing order of cost.

For each resource group, the following procedure is carried out. Each job in the unassigned jobs list is considered and the job completion time in the resource group is compared with the deadline one after the other. If the job completion time is less than the deadline for a particular resource, then the budget to be spent for the execution of the job in that resource is compared to the specified budget. The job is allocated to that resource if the budget needed is within the specified budget.

The resources are allocated such that job's requirements are satisfied with a single site to improve performance.

Deadline and budget constraints: The Constraints needed to perform scheduling are:

- Absolute Deadline
- Absolute Budget

These constraints make the system more economical with faster computational capability through allocation of cheapest resource to meet the deadline.

Absolute deadline: The absolute deadline value is calculated based on the D_{factor} (Deadline factor). A D_{factor} close in 1 signifies the user's willingness to set a highly relaxed deadline, which is sufficient to process applications even when only the slowest resources are available.

The time needed to process all the jobs, in parallel, using the fastest resource the highest priority is calculated and taken as T_{min}. The time required to process all the jobs, serially, using the slowest resource is calculated and taken as T_{max}.

The absolute Deadline $D_{absolute}$ is calculated as:

$$D_{absolute} = T_{min} + D_{factor} * (T_{max} - T_{min})$$

An application with D_{factor} less than zero would never be completed. An application with D_{factor} greater than or equal to one would always be completed as long as some resources are available with minimal user-share is available throughout the deadline.

Absolute budget: The absolute budget value is calculated based on the B_{factor} (Budget factor). A B_{factor} close to 1 signifies the user's willingness to spend as much money as required even when only the most expensive resource is used.

The cost of processing all the jobs, in parallel within deadline, giving the cheapest resource the highest priority is calculated and taken as C_{min}. The cost of processing all the jobs, in parallel within deadline, giving the costliest resource the highest priority is calculated and taken as C_{max}.

The absolute Deadline $B_{absolute}$ is calculated as:

$$B_{absolute} = C_{min} + B_{factor}*(C_{max}-C_{min})$$

An application with B_{factor} less than zero would never be completed. An application with B_{factor} greater than or equal to one would always be completed as long as some resources are available with minimal user-share is available throughout the deadline.

Dispatcher policy: This strategy performs allocation of jobs to space shared resources. The jobs are dispatched based on certain parameters. The number of jobs in ready state is taken as a list of jobs to be dispatched. The number of jobs that can be dispatched for each resource is calculated by subtracting the number of jobs in the queue from the product of the number of processing elements in the resource and maximum jobs per processing element. The optimal dispatch size of the resource is the minimum of the number of jobs to be dispatched and number of jobs that can be dispatched.

Performance metrics: The metrics like Cost factor and Speedup ratio are used to compare the explicit allocation strategy with DBC optimization against the explicit allocation strategy without optimization.

Cost factor: Cost Factor (C_{factor}) is the ratio of processing cost of all jobs using explicit allocation strategy without optimization ($C_{without\_optimization}$) to the processing cost of all jobs using an explicit allocation strategy with DBC cost time optimization ($C_{with\_DBCoptiimization}$).

This metric is a measure of economic efficiency of this system when compared to the explicit strategy used without optimization.

Speedup ratio: Speedup Ratio ($T_{speedup}$) is the ratio of execution time of all jobs using explicit allocation strategy without optimization ($T_{without\_optimization}$) to the execution time of all jobs using an explicit allocation strategy with DBC cost time optimization ($T_{with\_DBCoptiimization}$).

This metric is a measure of improvement in Computational performance of this system when compared to the explicit strategy used without optimization.

MATERIALS AND METHODS

Algorithms:

Explicit allocation strategy with DBC cost time Optimization scheduling algorithm: Get the user

requirements through a GUI on submission, do the following:

- Create the grid environment with the given specification for the following. (Resource name, Resource architecture, Operating System, Number of Machines, Number of Processing Elements (PE), Million Instructions Per Second (MIPS) for Processing Elements, Processing cost)
- Create local user and grid user jobs with given specifications. (Number of user jobs, Average Million Instructions per second, deviation percentage, granularity time, overhead time and resources required according to budget)
- Calculate
  - Absolute Deadline
  - Absolute Budget
- Submit jobs to Schedule Advising which uses cost time optimization scheduling strategy
- Once scheduling is done, dispatch the jobs to allocate resources. Dispatching of jobs is usually done as long as the number of user jobs deployed (active or in a queue) is less than the number of PEs in the resource
- Receive the processed job details from the resources
- Calculate the execution time and processing cost

Calculation of absolute deadline: Calculate T_{max}, the maximum value of the ratio of total job length to the minimum MIPS rating with load among all the resources and the ratio of maximum length value among all jobs to the maximum MIPS rating among all the resources.

Calculate T_{min}, the maximum value of the ratio of the total job length of the sum of the MIPS rating of all the resources and the ratio of maximum length value among all jobs to the maximum MIPS rating among all the resources.

Calculate the Absolute deadline using the formula:

$$D_{absolute} = T_{min} + D_{factor} * (T_{max} - T_{min})$$

Calculation of absolute budget:

- Calculate C_{max}, the product of total job length and the maximum value of cost per million instruction among all resources
- Calculate C_{min}, the product of total job length and the minimum value of cost per million instruction among all resources
- Calculate the Absolute budget using the formula:

$$B_{absolute} = C_{min} + B_{factor} *(C_{max}-C_{min})$$

Schedule Advisor (SA): For all resources in the environment Choose an initial set of possible job requests and estimate the number of jobs expected to execute from that set as:

$$\text{Expected To Finish} = \frac{\text{avail\_MIPS/ actual\_MIPS}}{\text{avail\_MIPSprev/ actual\_MIPS}} * \text{no of jobs}$$

where, avail_MIPS is the available processing power (MIPS) of the resource and avail_MIPSprev is the available processing power of the resource from the previous set of jobs. ActualMIPS is the original processing power of the resource:

- If a new possible request could improve this number (i.e., throughput), a previous chosen job is discarded and a new one is inserted into the set
- The Unassigned-Jobs list is then scheduled using cost time optimization as
- Sort the resources in the grid environment according to processing cost per million instructions. If two resources have equal cost, then sort according to the available processing time of the resource
- Repeat the following steps for each job in the Unassigned-Jobs-List
- Select a job from the Unassigned-Jobs-List
- For each resource, calculate/predict the job completion time taking into account previously assigned jobs and the job completion rate and resource share availability
- Sort resources by the increasing order of completion time
- Assign the job to the first resource and remove it from the Unassigned-Jobs-List if the predicted job completion time is less than the deadline

Table1: Resource configuration

RN	ARCH	OS	PE	PS	RC (G$)
R1	MAC	Linux	1	12	200
R2	IBM	Red hat	2	13	300
R3	Solaris	Mandrake	3	15	210
R4	MAC	Windows	3	20	400
R5	48C	Linux	2	12	50
R6	Solaris	Linux	1	11	60
R7	IBM	Windows	2	10	70
R8	IBM	Windows	5	10	150
R9	Solaris	Mandrake	3	20	400
R10	MAC	Windows	3	12	50

Table 2: Input specifications

	Number of jobs			
Input	Local	Grid	Deadline (sec)	Total load (MI)
1	4	3	19	1074.37
2	5	3	30	1718.49
3	6	3	30	1951.69
4	7	4	27	2333.46
5	7	5	30	2044.11

Simulation: In the simulation, resources of different configurations are considered. The capacity for computation in a CPU resource is provided in the form of MIPS. The other configuration parameters include resource name, Operation system, Architecture, Processing Speed (MIPS), resource cost (Grid Dollars). Each resource has a specific number of processing elements. The processing power of the resource depends on the number of PEs and the processing power of each PE in the resource. Table 1 indicates the Resource configuration for the current work. The Input Specification is depicted in Table 2 where input has both local jobs and Grid jobs and the deadline specified.

RESULTS

For the input specification of Table 2, the results are obtained.

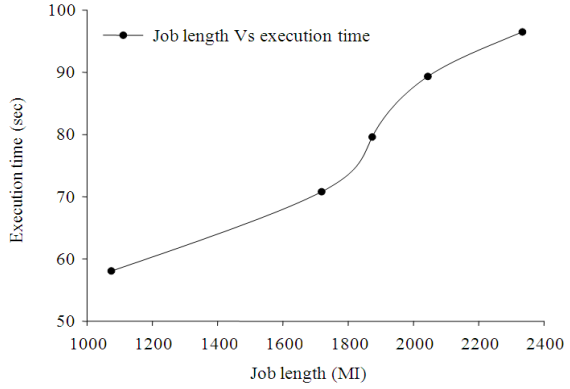

Fig. 2: Job length Vs execution time

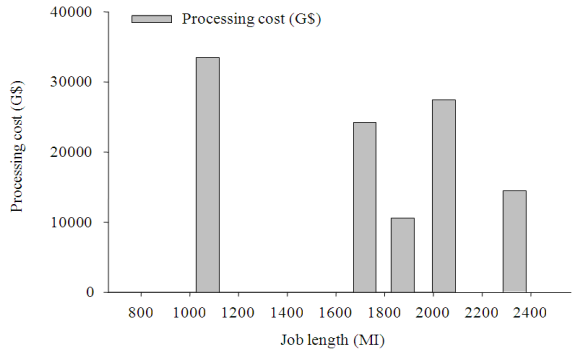

Fig. 3: Job length Vs processing cost

Table 3: Job length Vs execution time

Job length (MI)	No. of Jobs	Execution time (sec)
1074.37	7	58.06
1718.49	8	70.79
1873.78	9	79.60
2044.11	12	89.32
2333.46	11	96.48

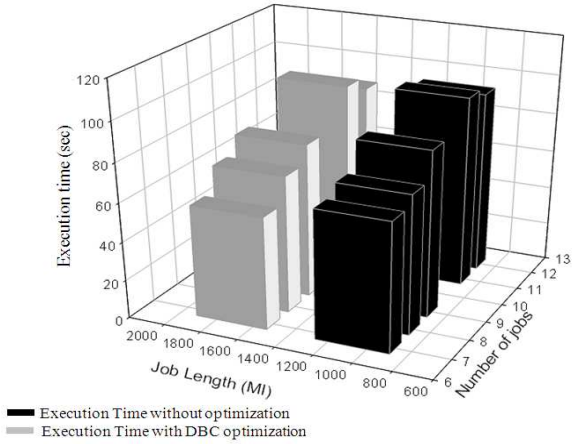

Fig. 4: Comparison of total execution time (without optimization Vs with DBC cost time optimization)

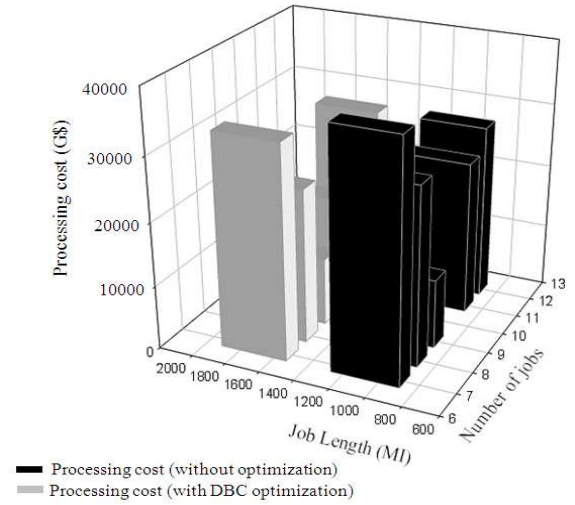

Fig. 5: Comparison of processing cost (Without optimization Vs with DBC cost time optimization)

Table 4: Job length Vs processing cost

No. of Jobs	Job length (MI)	Processing cost (grid Dollars)
7.0000	1074.3700	33476.4900
8.0000	1718.4900	24224.3400
9.0000	1873.7800	10562.9300
11.0000	2333.4600	14518.6500
12.0000	2044.1100	27384.7200

Table 5: Comparison of total execution time (Without optimization Vs with DBC cost time optimization)

No. of Jobs	Job length (MI)	Execution time without optimization	Execution time with DBC optimization	Speedup ratio
7	1074.37	66.71	58.06	1.148
8	1718.49	71.79	70.79	1.014
9	1873.78	85.92	79.60	1.079
11	2333.46	98.48	96.48	1.020
12	2044.11	94.15	89.32	1.054

Table 3 shows the execution time for various job length. Table 4 indicates the processing cost for these jobs. From Fig. 2 and Fig. 4 , Execution Time and Processing Cost varies proportional to Job Length. Fig. 3 depicts the processing cost in Grid dollar (G$) over the job length. Fig. 5 compares the processing cost of the DSC algorithm (optimization) versus the non optimized algorithm.

DISCUSSION

The two metrics considered in performance analysis are Cost Factor (C_{factor}) and Speedup Ratio ($T_{speedup}$). It is seen that the explicit allocation strategy with optimization out performs than the explicit allocation strategy without optimization in terms of processing cost and the total execution time needed to process all the jobs. From Fig. 6 and Fig. 7 the cost factor and speedup ratio for the given input specifications are greater than one, which shows that the proposed strategy is more economically and computationally efficient.

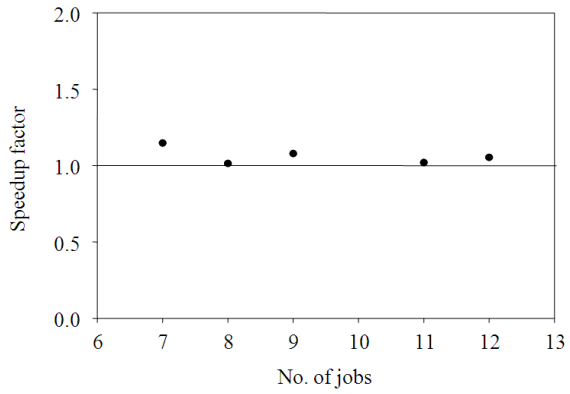

Fig. 6: Cost factor

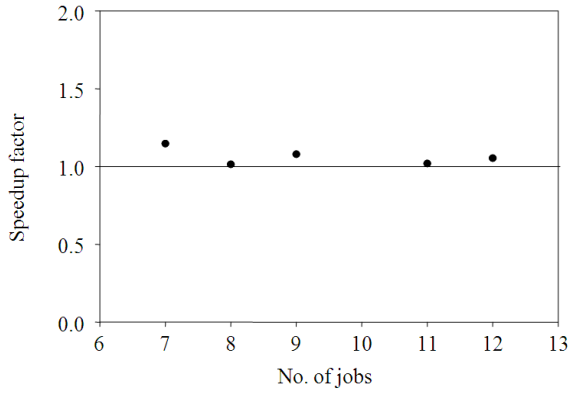

Fig. 7: Speedup ratio

CONCLUSION

This study presented is an Explicit allocation strategy with DBC optimization, which consists of deploying an adaptive heuristic to make a smart use of space shared resources based on the economic provisions of the user (budget), granting to grid users privileged access to an amount of resources as soon as possible, with deadline considerations. It enables execution of BoT applications in space shared resources thereby allowing efficient exploitation of high computation power.

This strategy is the extension of the existing explicit allocation strategy, which do not prioritize the jobs based on the user budget (i.e., user's willingness to pay more). In order to improve the performance, the proposed system includes the deadline and budget considerations provided by the user, schedules the job using cost time optimization, thereby making the system more economical.

The Cost Ratio and Speedup factor calculated from the analyzed results is greater than one. This shows that the proposed system (Explicit Allocation strategy with DBC optimization) is more economical and has high computational performance when compared to the existing system (Explicit Allocation strategy without optimization).

As future work, it is first intended to incorporate Load Balancing, to further improve the efficiency through effective scheduling. Secondly, "Task Replication" can be incorporated with the proposed system to speed up the processing of jobs.

REFERENCES

Boukerram, A. and S.A.K. Azzou, 2006. Implementation of load balancing algorithm in a grid computing. Am. J. Applied Sci., 3: 1810-1813. DOI: 10.3844/ajassp.2006.1810.1813

Buyya, R., D. Abramson, J. Giddy and H. Stokinger, 2002. Economic models for resource management and scheduling in grid computing. J. Concurr. Comput.: Practice Experi., 14: 1507-1542. DOI: 10.1002/cpe.690

Foster, I. and C. Kesselman, 1997. Globus: A metacomputing infrastructure toolkit. Int. J. Supercomputer Appli. High Perform. Comput., 11: 115-128. DOI: 10.1177/109434209701100205

Lee, Y.C. and A.Y. Zomaya, 2007. Practical scheduling of bag-of-tasks applications on grids with dynamic resilience. IEEE Trans. Comput. J., 56: 815-825. DOI: 10.1109/TC.2007.1042

Rose, C.A.F.D., T. Ferreto, R.N. Calheiros, W. Cirne and L.B. Costa *et al*., 2008. Allocation strategies for utilization of space-shared resources in Bag of Tasks grids. Future Generation Comput. Syst., 24: 331-341. DOI: 10.1016/j.future.2007.05.005