Static Heuristic task scheduling in tree based Environment

Mrs.S.Selvarani[1]*, Dr.G.Sudha Sadhasivam*[2] *& R.Kingsy Grace*[3]*,*

[1]*Department of Information Technology, Tamilnadu College of Engineering, Coimbatore, India*
[2]*Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, India*
[3]*Department of Computer Science and Engineering, Sri Ramakrishna Engineering College, Coimbatore, India*

Abstract:

The emergence of grid computing has facilitated the linking of heterogeneous and widely distributed resources. A suitable and efficient scheduling algorithm is needed to schedule users jobs to distributed resources. So grid scheduling is an important research area. A tree based grid platform is employed because most of network models can be simplified into tree-based models to resolve and to use parallel processing in grid computing environment.

The objective of this paper is to schedule task groups on a tree-based grid computing platform, where resources have different speeds of computation and communication. Due to job grouping, communication of coarse-grained jobs and resources optimizes computation/communication ratio. For this purpose, the multi-level tree is reduced to a single level tree. To transform a multilevel tree to a single level tree push-pull algorithm is used.

The proposed approach employs a static distributed heuristic task scheduling algorithm for establishing efficient mapping between tasks and available resources. This scheduling strategy groups the user jobs according to a particular Grid resource's processing capability, and sends the grouped jobs to the resource. Job grouping in tree based grid environment enhances the computation/communication ratio.

.

Key words: *Task scheduling; Grid computing, multi-level tree;*

1. Introduction

The popularity of the Internet and the availability of powerful computers and high speed networks, facilitate the usage of geographically distributed, heterogeneous resources to solve large-scale problems in science, engineering, and commerce at low-cost.

A Grid scheduler acts like a medium that receives applications from multiple users and selects feasible resources for them according to a certain rules and predicted system performance so as to generate application-to-resource mappings [1].

An FCFS scheduler [9] executes processes from the queue in their submission order. It is simple and easy to implement but it is not preemptive. The SJF scheduler [9] is exactly like FCFS except that instead of choosing the job at the front of the queue, it will always choose the shortest job available. A sorted list is employed to order the processes from longest to shortest. It offers better performance in comparison to FCFS because the shorter jobs are executed immediately but starvation may occur.

Min-Min Scheduling [9] evaluates the average execution time and data transfer time for the job to the resource. It then allocates the job to the resource that provides minimum-average value.

It is more dynamic and improves CPU utilization by reducing the idle time of the resources. The main disadvantage is the computation and communication overhead.

In Optimization-based Priority-Bandwidth Heuristic Algorithm for Task Allocation (OPBHATA) [9] the unscheduled tasks are allocated by the parent node in accordance with the network communication speed. In this approach optimal task assignment and distribution of programs becomes possible but the overhead time required for querying the Grid Information Service and gaining information is very high.

In Optimization-based Priority-Computation Heuristic Algorithm for Task Allocation (OPCHATA) [9] the unscheduled tasks are allocated based on the computing speed of the computing nodes. In this approach efficient optimization based on computation cost and distribution of programs becomes possible but the communication cost is not taken into account and the overhead time for routing the required information is high.

Tree based approaches

Sang Cheol Kim and Sunggu Lee proposed a new deterministic guided search algorithm termed "Push-Pull" that is found to be particularly effective for DAG scheduling on heterogeneous cluster systems [14]. The algorithm uses "pull" operations that eliminate

Manuscript received June 5, 2010
Manuscript revised June 20, 2010

communication requirements between nodes interconnected by slow network links.

Job grouping

To reduce communication overhead the scheduling strategy should group a number of user jobs together according to a particular Grid resource's processing capabilities, and send the grouped jobs to that resource.

Gerasoulis and Yang [11], in the context of Directed Acyclic Graph (DAG) scheduling in parallel computing environments, named grouping of jobs to reduce communication dependencies among them as clustering.

Nithiapidary Muthuvelu et al.[12] presented a scheduling strategy that performs dynamic job grouping activity at runtime. This dynamic grouping was done based on the processing requirements of each application, availability of grid resources and their processing capability.

A job scheduling strategy that encompasses the job grouping concept coupled together with bandwidth-aware scheduling is proposed and evaluated by T.F.Ang et al. [13]. The proposed scheduling strategy focuses on grouping independent jobs with small processing requirements into suitable jobs with larger processing requirements and schedules them in accordance with indeterminist network conditions.

Muthuvelu *et al.* [12] proposed the Dynamic Job Grouping strategy which concentrates on maximizing the utilization of Grid resource processing capabilities and reducing the overhead time and cost taken to execute the jobs using a batch mode dynamic scheduling.

In this approach to determine the optimal number of tasks assigned to each computing node a static distributed heuristic task scheduling algorithm is employed. This approach pushes a multi-level tree into a similar equivalent tree with only one node and then pulls this equivalent tree to calculate the optimal number of tasks assignment to each new two-level sub tree (called single-level sub tree). Another merit of the algorithm is to reduce the total time taken in transmitting the user jobs to/from the resources, and also it reduces the overhead processing time of each job at the resources.

In the existing approaches no job grouping was done. So the proposed approach groups a number of user tasks together according to a particular Grid resource's processing capabilities, and transfer the coarse-grained jobs to the resources for processing.

This paper discusses how to schedule independent tasks on the tree-based grid computing platforms, where resources differ in their computation and communication capability. The optimal scheduling scheme that determines the optimal number of tasks assigned to each computing node is obtained. This is done by pushing a multi-level tree up into a similar equivalent tree and then pulling this equivalent tree to obtain an optimal task assignment. A static distributed heuristic task scheduling algorithm is used. This approach reduces the total time taken in transmitting the user jobs to/from the resources and the overhead processing time of the jobs at the resources.

2. Proposed Scheduling Model

Tree-based grid computing platform has a master-slave architecture where each computing node has zero or more slave nodes, but has only one master node [10]. The proposed approach uses tree-based heterogeneous grid computing platform and considers migration costs and computation costs at each node.

Figure 1 shows a general single level tree-based grid computing platform, consisting of k computing nodes n_0, n_1,….. n_{k-1}. Let $w_0, w_1, \ldots, w_{k-1}$ be the computing capability of the slaves. Let $c_0, c_1, \ldots, c_{k-1}$ be the communication time needed to transfer the unit tasks from master to slave nodes

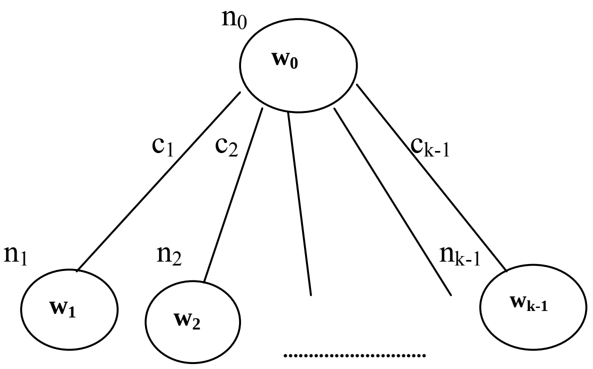

Figure 1. Single-level Grid Computing platform

A single task attempter in tree-model Grid computing environment [2] neglects its communication time from master to slave. So, the task assignment solution obtained using linear planning, is only the upper bound of the optimal solution.

An optimal task assignment for 'M' independent tasks in a single level grid computing platform should consider the number of tasks to be assigned to the master as well as slave nodes in order to minimize the execution time of the tasks.

Some of the constraints in a tree-based environment are:

- Consider the number of tasks running on node n_j is x_j, M equals total number of tasks running on all nodes then $\sum_{i=0}^{k-1} x_i = M$

- Number of tasks run on any one node is smaller than the total number of tasks
  $0 \le x_i \le M \;\; for\; 0 \le i \le k-1$
- Execution time of a node is less than the total execution time. Then $\forall i$ $w_i . x_i \le T$

3. Proposed push-pull methodology with job grouping

The proposed tree based scheduling algorithm has two basic operations namely push and pull.

3.1 Push operation

For a multi-level tree T, starting from leaf nodes with the same master node, a single-level tree consisting of a master node and its child nodes can be substituted by an equivalent node, that finally become a child node of the new tree node has computation capacity approximately equal to the original tree. Equivalent Tree Transformation algorithm (ETT) is used for push process is given below
(1) Starting from leaf nodes with the same father node, a single-level tree consisting of a father node and its leaf nodes is substituted by an equivalent node [4]. That is, first delete the leaf nodes of the single level tree in the original tree, then use an equivalent node to substitute the root node of the single level tree, which has the same computation and communication capacity, finally a new tree is created, and the equivalent node become a leaf node of the tree.
(2) Record the sequence S of Push process and the link between equivalent node and its corresponding single level tree.
(3) Repeatedly use step (1) and step (2) on the newly created tree in step (1), until a tree with only one node is left.

An example for push operation is shown below in Figure 2, Figure 3 and Figure 4

A multilevel tree with 6 nodes n_0 to n_6 is shown in Figure 2. The computation capacity of nodes n_0 to n_6 is w_0 to w_6 respectively and the communication time needed to transfer the unit tasks from n_0 to n_1 is c_0 and from n_0 to n_2 is c_1 and so on.

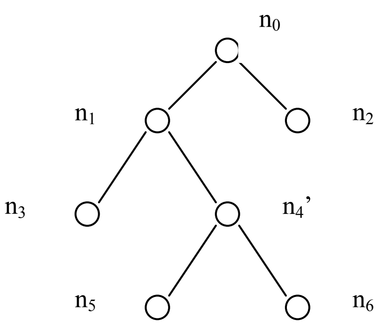

Figure 2 Multi-level tree

First push operation is shown in Figure 3 below. The single level tree consisting of n_4, n_5 and n_6 is converted into equivalent node n_4'. Computation capacity of n_4' is the computation capacity of the single level tree with nodes n_5 and n_6 namely $w_4+w_5+w_6$. The communication ability is approximately equivalent to the total communicating ability of the single level tree between n_4' and n_1 namely c_4.

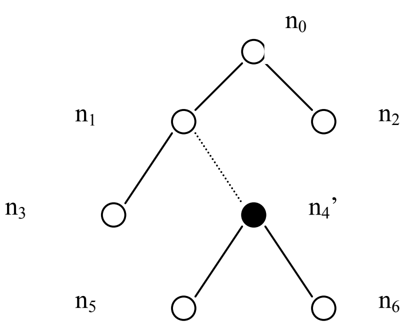

Figure 3 First push operation

Second push operation is shown in Figure 4 below. The single level tree consisting of n_4', n_3 and n_1 is converted into equivalent node n_1'. Computation ability of n_1' is equivalent to the computation capacity of n_4' and node n_3 namely $w_3+w_4+w_5+w_6$.

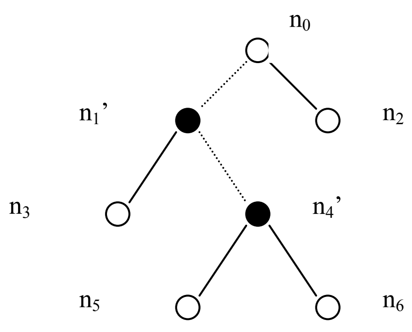

Figure 4 Second push operation

3.2 Pull Operation

Pull is the inverse process of Push operation.

- Take the equivalent node from recorded sequences.
- Perform optimal task assignment by comparing the characteristics of the incoming tasks and available resources.

The proposed tree based algorithm with job grouping, removes the defects of FCFS, SJF, Min-Min, OPCHAT and OPBHAT algorithms [2]. This algorithm deals with single level tree. Multi-level tree can be converted to single level tree to use this algorithm.

Initialization

The user must specify the number of Gridlets(jobs), average Gridlet length (MI), MI deviation percentage, granularity time, Gridlet overhead processing time, and the resource file to the system through the GUI. The user can choose to perform the simulation with or without using the Gridlet grouping method. This facilitates the user to compare the output between a simulation without gridlet grouping and simulation with Gridlet grouping method.

Grid Resource Creation

Grid resources are created from a resource file. The resource file contains a list of available resources on the Grid. Each resource has their characteristics specified in the file namely, resource name, architecture, operating system, number of machines, number of PEs, MIPS of each PE, time zone, processing cost, communication speed, random seed, and resource load during peak hour, off-peak hour, and holiday.

Gridlet Creation

User specifies three parameters that affect the Gridlet creation task, namely, the number of Gridlets, the average MI of each Gridlet, and the MI deviation percentage. The MI deviation percentage refers to the deviation percentage of average MI of each Gridlet. Each resulting Gridlet will be assigned to a Grid user, and then added into a Gridlet list.

They will be assigned with attributes like Gridlet ID,MI, File size and Output size

Push Operation

Push operation for a multi-level tree as explained above

Pull Operation with job grouping

The proposed job grouping algorithm is done in pull operation is described as follows (Figure 5)

The Gridlet_List (n jobs) is submitted to the scheduler. The scheduler receives the Resources_List which is the inverse sequence of the resource list recorded in step 2 of push operation. It multiplies the selected resource's MIPS with the granularity size which indicates the total MI the resource can process within a specified granularity size. The scheduler groups the Gridlets by accumulating the MI of each gridlet while comparing the resulting Gridlet total MI with the resource total MI. If the total MI of the Gridlets is more than the resource MI, the very last MI added to the Gridlet total MI will be removed from the Gridlet total MI. Eventually, a new Gridlet (grouped Gridlet) of accumulated total MI will be created with a unique ID and scheduled to be executed in the selected resource. This process continues until all the user jobs are grouped into few groups and assigned to the Grid resources. After all the Grouped_Gridlets are processed by the Grid resources, and sent back to the I/O queue of the scheduler/system, collect the grouped gridlets from the I/O queue. Set the resource ID, and job execution cost of each Grouped gridlets. Get the total simulation time. Display the details of the processed Gouped_Gridlets to the user through GUI.

The terms used in the proposed algorithm are

n : Total number of gridlets

m : Total number of Resources available

G_i : List of Gridlets submitted by the user

R_j : List of Resources available

MI : Million instructions or processing requirements of a user job

MIPS : Million instructions per second or processing capabilities of a resource

Tot-Jleng : Total processing requirements (MI) of a Gridlet group (in MI)

$Tot\text{-}MI_j$: Total processing capabilities (MI) of jth resource
$R_j\text{-}MIP$: MIPS of jth Grid resource
$G_i\text{-}MI$: MI of ith Gridlet
Granularity_Size : Granularity size (time in seconds) for job grouping
GJ_k : List of Grouped Gridlets
$TargetR_k$: List of target resources of each grouped job

Figure 5 Proposed algorithm with job grouping

```
Get Gridlet-size n
  Get Resource-list size m
    for i=0 to n-1 do
      get Gridlet $G_i$
    endfor
    for j=0 to m-1 do
  get Resource $R_j$ from the resource list recorded in push operation
    endfor
    k=0
    for i=0 to n do
    for j=0 to m do
        Tot-Jleng = 0
        $Tot\text{-}MI_j = R_j\text{-}MIP * granularity\text{-}size$
        while((Tot-Jleng <= $Tot\text{-}MI_j$) and (i <=n-1)) do
            Tot-Jleng = Tot-Jleng + $G_i$-MI
             i++
        endwhile
        i--
        if Tot-Jleng > $Tot\_MI_j$ then
            Tot-Jleng = Tot-Jleng – $G_i$-MI
             i--
        endif
       create a new Gridlet with Tot-MI equals to Tot-Jleng
        assign a unique ID to the newly created Gridlet
        insert the Gridlet into Grouped Gridlet $GJ_k$
        insert $R_j$ in $TargetR_k$
        k++
    endfor
    endfor
    for i = 0 to Grouped Gridlet-1 do
        send $GJ_i$ to $TargetR_i$
    endfor
    for i = 0 to Grouped Gridlet-1 do
        receive computed $GJ_i$ from $TargetR_i$
    endfor
```

4. Experiment results

4.1 Performance evaluation

The job execution time, $T_{i,exec}$ involves both computational time, $T_{i,comp}$ and communication time, $T_{i,comm}$.

$$T_{i,exec} = T_{i,comp} + T_{i,comm} \qquad (1)$$

Where:
$T_{i,exec}$ = Execution time of a grouped job
$T_{i,comp}$ = Computation time of a grouped job
$T_{i,comm}$. = Communication time of a grouped job

The total processing time for a single job is calculated as follows :

$$T_{proc} = T_{submit}+T_{i.exec}/n+T_{receive} \qquad (2)$$

Where:
T_{proc} = Total processing time
T_{submit} = Time taken to submit a job
$T_{receive}$ = Time taken to receive a processed job
n = Number of jobs in a group

However, for r_i grouped jobs $J_{i,1}$, $J_{i,2}$, …, $J_{i,ri}$, the job processing time is the maximum time taken to execute all the grouped jobs from the time of the first job sent by the user to the final job received by the user after all the jobs are executed at the resources.

To summarize:

$$T_{effect} = T_{i\ end} – T_{i\ start} \qquad (3)$$

Where:
T_{effect} = Effective execution time
$T_{i\ end}$ = Time when the last job received
T_{istart} = Time when the first job sent

The total processing time is calculated based on the time taken to group the jobs, to submit all the grouped jobs, to receive all the processed jobs and the effective execution time.

$$T_{proc} = T_{grouping}+T_{submit}+T_{oh}+T_{effect}+T_{receive}, \qquad (4)$$

Where:
T_{proc} = Total processing time
$T_{grouping}$ = Time taken to group jobs
T_{submit} = Time taken to submit all the grouped jobs

T_{oh} = Overhead time
T_{effect} = Job processing time
$T_{receive}$ = Time taken to receive all the processed jobs

The grouping of jobs depends on a specific granularity size. Granularity size is the time within which a job is processed at the resources. As stated by Muthuvelu *et al*[12], this value is used to measure the total amount of jobs that can be completed within a specified time in a certain resource. It is one of the main factor in job grouping strategy that influences the way job grouping is performed to achieve the minimum job execution time and maximum utilization of the Grid resources.

4.2 Results and Discussions

GridSim has been used to create the simulation environment. The inputs to the simulations are total number of gridlets, average MI of Gridlets, MI deviation percentage, granularity size and Gridlet overhead time. The MIPS of each resource is also specified (Table 1)

Resource	MIPS
R1	20
R2	44
R3	69
R4	296
R5	126
R6	210
R7	204

Table 1 : MIPS of Grid Resources

The performance of tree based algorithm has been compared with FCFS, with and without job grouping [5], [6].

This paper adopts GridSim4.2 [8] to simulate the algorithm of task scheduling given above. GridSim provides a series of core function for the establishment and simulation of heterogeneous distributed computing environment, particularly suitable for simulation and research of task scheduling on grid. We simulated a tree-based grid computing platform with seven nodes, five seconds of granularity time, five seconds of overhead time and 10% deviation. Then the platform used FCFS and tree-based scheduling algorithm with and without job grouping to schedule independent tasks of number of 50, 100, 150, 200 and 250 of the same scale. Table 2 below shows the results of the simulation.

Number of Gridlets	**With Grouping**		**Without Grouping**
	Number of Groups	**Process Time**	**Process Time**
50	2	70	655
100	4	115	1308
150	4	123	1958
200	4	135	2612
250	5	146	3257

Table 2 Simulation with and without grouping

When scheduling 50 Gridlets, simulation with job grouping method groups the Gridlets into two groups (R1 and R2). The total overhead time is 10 (5 x 2) seconds and the total process time is 70 seconds. Therefore 60 seconds was spent for grouping, scheduling and sending back the scheduled Gridlets. This result can be compared with the simulation without job grouping, where all Gridlets are sent to individual resources and the total overhead time is 250 seconds (50 x 5) and total process time is 655 seconds. Here the total Gridlet computation time is 250 seconds and the communication time is 405 seconds. The communication time is very high when compared with the communication time of the simulation with grouping.

We can compare the results for various number of Gridlets.

a) Figure 6 compares Tree based scheduling algorithm with and without grouping on the basis of time taken for completion of the tasks

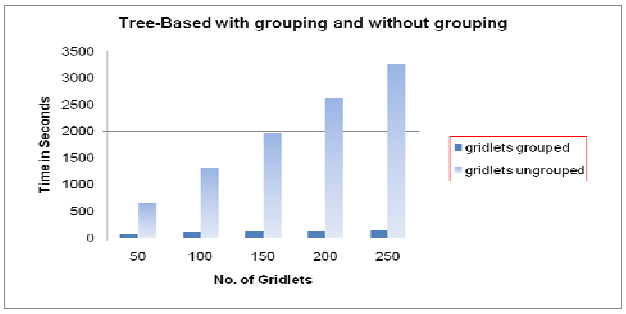

Figure 6 Comparison of Tree-Based Scheduling with and without job grouping

From the above Figure it can be seen that for Tree-Based Scheduling the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks.

b) Figure 7 compares FCFS scheduling algorithm with and without job grouping on the basis of time taken for completion of the tasks

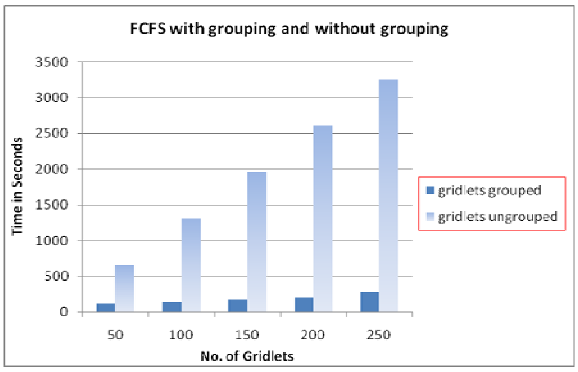

Figure 7 Comparison of FCFS Scheduling with job grouping and without job grouping

From the above Figure it can be seen that for FCFS scheduling the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks.

c) Figure 8 compares FCFS scheduling algorithm with and without job grouping on the basis of time taken for completion of the tasks

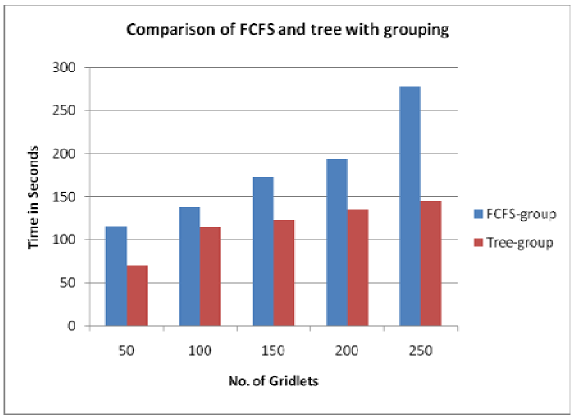

Figure 8 Comparison of FCFS and Tree-based Scheduling with job grouping

From the above graph which shows the comparison of completion time taken for FCFS scheduling and Tree-based scheduling with task-grouping, we can conclude that the Tree-based scheduling algorithm is better than FCFS scheduling algorithm.

6. Conclusion and Future works

This paper discusses job scheduling in heterogeneous tree-based grid computing environment. By doing research and analysis of this problem, that aims at task scheduling with minimum total tasks completion time on a multi-level tree grid computing platform, a new measure, called Push-Pull is used to build a single level tree, and develop a linear planning model for it. Through Push-Pull, the problems of optimal number of tasks assignment and task scheduling on a multi-level tree is iteratively converted to those of a groups of single level tree, which can be implemented in parallel on a tree grid platform. GridSim is employed to carry out and simulate the tasks assignment algorithm, and distributed task scheduling .The results are compared with FCFS. The conclusion is that the scheduling algorithm employed is better than FCFS.

This Static Heuristic Scheduling algorithm only takes the initial research on task scheduling in tree based platform. However many issues remain open. Further improvement should be done to handle more complicated scenario involving dynamic factors such as dynamically changing grid environment and other QoS attributes. The improvement of this algorithm should concentrate on discussing simultaneous instead of independent task scheduling in heterogeneous tree-based grid computing environment

Acknowledgment
The authors convey their heartfelt thanks to Dr.C.Kalaiarasan, Principal, Tamilnadu College of Engineering, Coimbatore and Mr.K.V.Chidambaran, Director Cloud Computing Group, Yahoo Software Development (India) Pvt Ltd, Bangalore for providing them the required facilities to complete the project successfully. This project is carried out as a consequence of the PSG-Yahoo Research programme on Grid and Cloud computing

References

[1] Abraham A, Buyya R,Nath B. Nature's heuristics for scheduling jobs on Computational grids. In: Proc. of the 8th Int'l Conf on Advanced Computing and Communications (ADCOM 2000). New Delhi:Tata McGraw-Hill Publishing, 2000. 45-52.
[2] Linweiwei,qidele,liyongjun,wangzhenyu,zhangzhili. Single task attempter in tree-model Grid Computing environment. Software transaction.2006, 16(06), pp.1000-9825
[3] Independent Tasks Scheduling on Tree-Based Grid Computing Platforms LIN Wei- Wei$^+$, QI De-Yu, LI Yong-Jun, WANG Zhen-Yu, ZHANG Zhi-Li

(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)

[4] An implementation of Karmarkar's algorithm for linear Programming Ilan Adler, Narendra Karmarkar, Mauricio g.c. Resende, and Geraldo Veiga

[5] Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Technical Report,2006-504, School of Computing, Queen' s University, Canada, January.2006

[6] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour, "On the Design and Evaluation of Job Scheduling Systems", D. G. Feitelson and L. Rudolph, editors, IPPS/SPDP'99 Workshop: Job Scheduling Strategies for Parallel Processing. Springer, Berlin, Lecture Notes in Computer Science, LNCS 1659, 1999

[7] I. Foster, C. Kesselman. The Grid, "Blueprint for a Future Computing Infrastructure [M]", USA, Morgan Kaufmann Publishers, 1999

[8] R. Buyya, Manzur Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing" *Concurrency Computation: Practice and Experience, Vol. 14, No. 13-15*, 2002, pp. 1507-1542

[9] Abraham. A., Buyya. R., and Nath. B.: "Nature's heuristics for scheduling jobs on computational grids", Proc. 8th IEEE Int. Conf. on Advanced computing and communications, Cochin, India, 2000.

[10] Practical scheduling algorithms of independent tasks on tree-based grid computing platform zhen-yu wang, can-cheng yang, Proceedings of the Sixth nternational Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007

[11] Gerasoulis, A. and Yang, T. A comparison of clustering heuristics for scheduling directed graphs on multiprocessors. *Journal of Parallel and Distributed Computing*, **16**(4):276-291.

[12] Nithiapidary Muthuvelu, Junyang Liu, Nay Lin Soe, Srikumar Venugopal, Anthony Sulistio and Rajkumar Buyya, A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids A*ustralasian Workshop on Grid Computing and e-Research (AusGrid2005)*, Newcastle, Australia. Conferences in Research and Practice in InformationTechnology, Vol. 44.

[13] T.F.Ang,W.K.Ng,T.C.Ling,L.Y.Por and C.S. Liew : "A Bandwidth-Aware Job Grouping-Based Scheduling on Grid Environment" in the proceedings of Information Technology Journal, 2009,Vol:8, Issue 3,Page No. : 372-377

[14] Sang Cheol Kim, Sunggu Lee, "Push-Pull: Deterministic Search-Based DAG Scheduling for Heterogeneous Cluster Systems" Proceedings of Parallel and distributed systems, vol. 18, no. 11, November 2007

Authors Profile :

S.Selvarani received the B.E. degree in Computer Science and Engineering from Bharathiar University in 1991, and M.E. degree in Computer Science and Engineering from Manonmaniam Sundaranar University in 2004. She is currently working toward the PhD degree in Computer Science and Engineering under Anna University, Coimbatore, India. Her research work is in Grid and Cloud Computing with techniques for resource management and task scheduling

Dr G Sudha Sadasivam is working as a Professor in Department of Computer Science and Engineering in PSG College of Technology, India. her areas of interest include, Distributed Systems, Distributed Object Technology, Grid and Cloud Computing. She has published 20 papers in referred journals and 32 papers in National and International Conferences. She has coordinated two AICTE-RPS projects in Distributed and Grid Computing areas. She is also the coordinator for PSG-Yahoo Research on Grid and Cloud computing.