Gridlock Prevention by Job Split-up

Suresh S[†] and Poornaselvan[††],

Government College of Technology, Coimbatore, Tamilnadu, India

Summary
The absence of task completion and return timing guarantees from the remote clients in Internet based Computing papers is largely an annoyance when the tasks comprising the shared workload are mutually independent. When the workload's tasks have interdependencies, that constrain the order of execution, the lack of timing guarantees presents a nontrivial scheduling challenge. Such dependencies can potentially engender Gridlock when no other tasks can be allocated for an intermediate period, pending the execution of already allocated tasks. In a Grid Framework, due to inter dependencies of the tasks in a tree structured computation, one task keeps waiting for the results of another task. The results may be delayed due to no timing guarantees and no return of results guarantees which results in a problem called gridlock.

Key words:
Grid computing, Job scheduling, Gridlock, Grid framework, task interdependencies.

1. Introduction

In order to solve the problem of gridlock, two methods were introduced.

1.1 Allocation of Tasks to Multiple Clients

In order to ensure that a task's result is generated and passed to its dependent tasks, a particular task is allocated to multiple clients. However, it cannot be guaranteed that all the clients will successfully return the results. Moreover, this multiple allocation of tasks significantly thins out the remote workforce.

1.2 Deadline Triggered Re Allocation of Tasks

Another suggestion to avoid the problem of Gridlock is that, a dead-line is set to the remote client. The client's failure to return the results will result in re-allocation of the task to another client. But well-planned re-allocation of tasks requires a reliable model of clients computing behavior. Moreover, these methods do not eliminate the problem of Gridlock since the back-up remote clients assigned a given task may be as dilatory as the primary one. Thus, no such technique eliminates the danger of gridlock.

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

2. Proposed System

The proposed system for Grid lock avoidance monitors the Network for the availability using system information like CPU idle time and memory availability the application to be processed is split into small tasks for parallel processing in the Grid Framework. The split up tasks are scheduled level by level for execution using the optimized algorithm. This avoids the problem of Gridlock due to task interdependencies. It also schedules the eligible tasks so as to maximize the number of eligible tasks at each step of computation, thus making maximum utilization of the available resources in the Grid Framework. The schematic diagram in Fig 1 represents the working of IC scheduling algorithm in a Grid Framework. The system information named CPU idle time and the memory usage are monitored and given to the Grid Framework. The job submitted for execution by application server is split into small tasks and given to the grid framework which in turn schedules the eligible tasks to remote clients for parallel processing. The results of individual tasks are aggregated based on the task dependencies and the final result is returned to the application server.

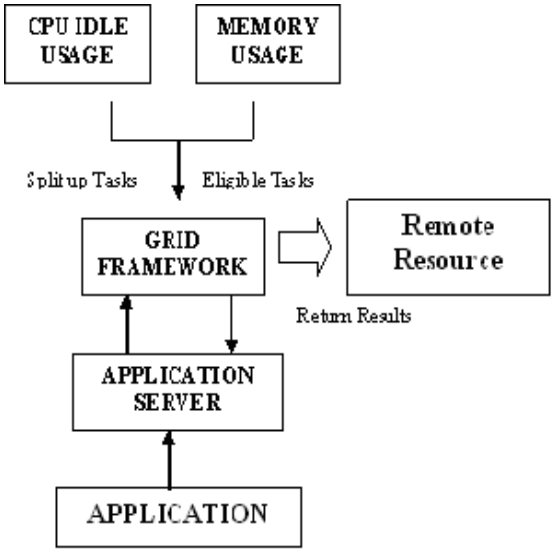

Fig1. Schematic Diagram of Proposed System

Grid computing simply means taking distributed computing resources and sharing them. It offers a way of

coordinating resources sharing and problem solving in or physically dispersed virtual organizations. Direct access to computers, data, software, storage and other resources is provided but with tightly controlled and clearly defined rules as to who can access, what is to be shared. Fig.2 below shows the approach used in Grid.

3. Grid Approach

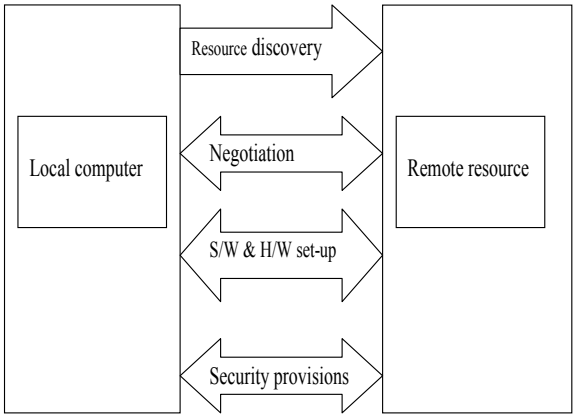

Fig. 2 GRID Approach

3.1 Grid Computing

Grid Computing is a collection of distributed, possibly heterogeneous resources, which can be used as an ensemble to execute large-scale applications.
Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations. Grid technologies promise to change the way organizations tackle complex computational problems. Grid computing is an evolving area of computing, where standards and technology are still being developed to enable this new paradigm. Today the Internet itself is changing to become a computing platform called the Grid. The grid is a dependable, universal computing infrastructure builds on the power of the net. Grid technology distributes computing jobs and database across numerous servers, has largely been an academic phenomenon. Grids are clusters of interconnected computers that can collectively tackle large computational problems or provide quicker access to very large bodies of data. Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across "multiple" administrative domains based on the resource availability, capability, performance, cost, and user's quality of service requirements. Grid computing allows coupling geographically distributed resources and offers consistent and inexpensive access to resources irrespective of their physical location or access point.

3.2 GRIDLOCK

Absence of task completion and return timing guarantees from remote clients in IC papers is largely an annoyance when workload's tasks have inter-dependencies that constraint the order of execution. Such dependencies can potentially engender Gridlock when no tasks can be allocated for an intermediate period pending the execution of already allocated tasks.

4. GRID Infrastructure

This technology allows us to access a much more powerful virtual computing infrastructure. A grid infrastructure, which is shown in the Fig 3, needs to provide more functionality than the Internet on which it rests, but it must also remain simple. And of course, the need remains for supporting the resources that power the grid, such as high-speed data movement, caching of large datasets and on-demand access to computing. Grid tools are concerned with resource discovery, data management, scheduling of computation, security, etc.

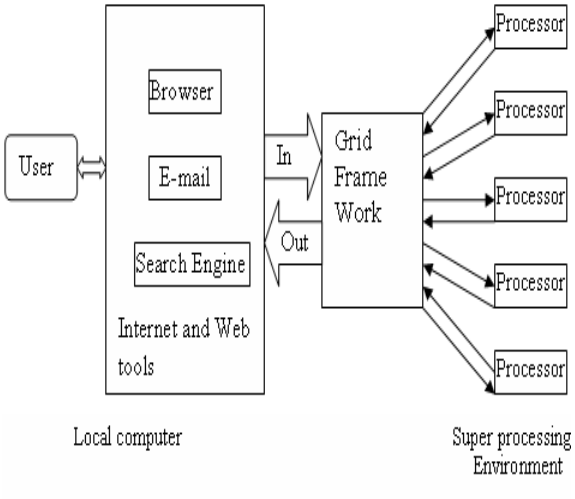

Fig. 3 GRID Infrastructure

Just as it is possible to tap into a power grid to access additional electricity on demand, it is also possible to tap into a computer grid to access additional computing power on demand using the cheapest possible resource: idle processing cycles on the existing machines. Basically, when a job is submitted to the grid through an interface on the computer, it serves as a portal to the grid. Special grid-

management software accepts the job and breaks it down into hundreds or even thousands of independent tasks then locates idle processors and distributes the tasks among them. Finally, it aggregates the work and split out the results.

5. Importance of GRID

Organizations that depend on access to computational power to advance their business objectives often sacrifice or scale back new papers, design ideas, or innovations due to sheer lack of computational bandwidth. Paper demands simply outstrip computational power, even if an organization has significant investments in dedicated computing resources.
Even given the potential financial rewards from additional computational access, many enterprises struggle to balance the need for additional computing resources with the need to control costs. Upgrading and purchasing new hardware is a costly proposition, and with the rate of technology obsolescence, it is eventually a losing one. By better utilizing and distributing existing compute resources, Grid computing will help alleviate this problem.

5.1 Benefits of GRID

Many companies want to take advantage of the cost and efficiency benefits that come from a grid infrastructure today, without being locked in to a system that will not grow with their needs.

- **Lower Computing Costs**

On a price-to-performance basis, the Grid platform gets more work done with less administration and budget than dedicated hardware solutions. Depending on the size of your network, the price-for-performance ratio for computing power can literally improve by an order of magnitude.

- **Faster Paper Results**

The extra power generated by the Grid platform can directly impact an organization's ability to win in the marketplace by shortening product development cycles and accelerating research and development processes.

- **Better Product Results**

Increased, affordable computing power means not having to ignore promising avenues or solutions because of a limited budget or schedule. The power created by the Grid platform can help to ensure a higher quality product by allowing higher-resolution testing and results, and can permit an organization to test more extensively prior to product release.

6. GRIDLOCK Avoidance Scheduling

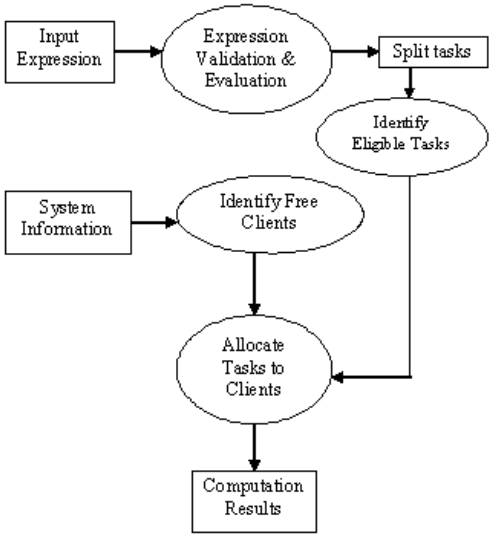

Fig. 4 System Diagram

The input expression is validated and evaluated using the w3eval algorithm to split the expression into individual tasks. The tasks that are independent of results of other tasks are identified as eligible tasks. The system information like CPU usage and memory availability of the resources is used to identify the free clients. The eligible tasks are scheduled to free clients and the result of the expression evaluated is displayed finally. The Fig. 3. 1 represents the system diagram.

7. Module Description

This section describes the modules in the paper.

1. Network Monitoring
2. Job Splitting
3. Tasks Scheduling

7.1 Network Monitoring

Network Monitoring is the module that monitors the remote clients for their availability. Availability of these clients is decided based on the amount of their CPU cycles and their Memory capacity being used for any on-going executions. Such system information is retrieved using native languages such as C, C++ or C#.
Load Balancing [4] is the technique wherein a scheduling that minimizes the maximum completion time for a job is aimed at. This scheduling technique collects the system information of all clients in a Grid Framework. The

system information like CPU idle time and memory availability are used to decide the availability of a client.

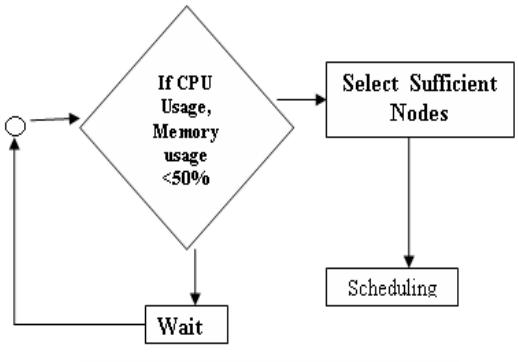

Fig. 5 Network Monitoring

The System waits for clients with low utilization of CPU and memory before scheduling the tasks as shown in the Fig. 3. 2.

7.1.1 CPU Usage

CPU Usage is the data that represents the amount of CPU cycles being spent at the remote client for local processes. This data is to be taken into account because busy processes should not be overloaded with more tasks which may lead to system failure or hang up.It is calculated based on the Operating System versions. On Windows NT, CPU usage counter is '% Total processor time' whose index is 240 under 'System' object whose index is 2. And, in Win2K/XP, Microsoft moved that counter to '% processor time' whose index is 6 under '_Total' instance of 'Processor' object whose index is 238. The index value is specified as a parameter to the method that finds the CPU Usage information using performance counters.

7.1.2 Memory Usage

Memory Usage is one another information to be measured to find the availability of remote clients. Memory Usage is the amount of main memory capacity occupied by the program being executed and to store its intermediate results. A remote client in the Grid Framework is available for use only if a minimum of 50% of its main memory is free.

7.2 Job Splitting

The application to be processed is split into small tasks for parallel processing in the Grid Framework. This module has two sub-modules: Expression Validation and Expression Evaluation.

7.2.1 Expression Validation

This sub module checks whether the input job is valid for processing. The input expression is checked for invalid characters, invalid sequence of operators and operands and for unpaired braces. Error message is displayed in case any error of such kinds is found.

Table 7. 1 VALID TOKENS

Characters	a-h A-H
Operators	Addition : + Subtraction : - Multiplication : * Division : /
Parentheses	Open Parenthesis : (Close Parenthesis :)

Table 7. 2 LEGAL TOKEN SEQUENCES

	Characters	Operators	(	)
Characters		X		X
Operators	X		X	
(	X		X	
)		X		X

Table 7.1 shows the list of valid characters that can be given in the input expression. The token on the X-axis can follow the token on the Y-axis if their meeting place is marked with an X in Table 7.2.

7.2.2 Expression Evaluation

The job checked for validity is then split up into several levels of inter dependent tasks. A close brace in the expression is identified first. Its corresponding open brace pair is identified then by traversing back through the expression. The characters between this brace pair are concatenated to a string variable that is a part of string array that represent a task. The level and the task number are kept in track.

Example Job: **(((a+b)*(c+d))-((e+f)*(g+e)))** L0

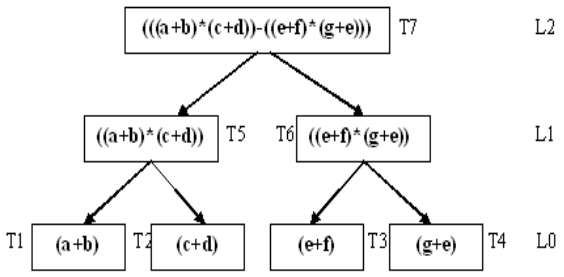

Fig. 6 Take Split Up

T1, T2, T3, T4, T5, T6, T7 – Task Number
L0, L1, L2 – Levels in Tree Structure

Fig.6. depicts the expression evaluated into individual tasks and the levels and task number assumed with each and every task.

7.3 Task Scheduling

The split up tasks are scheduled level by level for execution, in this module. Tasks are scheduled based on two different algorithms to show the difference between the existing system and proposed system. Scheduling is the process of allocating the tasks to available clients in the Grid Framework.

8. IC Scheduling Algorithm

Arnold L. Rosenberg in his study [1] used IC Scheduling algorithm to avoid the problem of Gridlock that occurs due to inter dependencies of the tasks. IC Scheduling algorithm aimed at avoidance of Gridlock, schedules all the tasks level by level. Since the tasks are scheduled only with the guarantee that all parent task results are available, no clients will be waiting with a task at hand. With the results of parent tasks readily available, tasks are executed immediately after they are scheduled. Since no waiting is involved with remote clients, the problem of Gridlock is avoided. To ensure this, the tasks whose parent tasks results are available, are marked as Eligible ones. Eligible tasks are then scheduled and the results are updated simultaneously. The following analysis shows how to schedule tree-structured computations IC Optimally. Considering l-level tree T that has S leaves (source nodes) with Sl source nodes at each level l, for l Є {0, 1, 2… l-1}. At step t of the computation, each level l of T has El ELIGIBLE nodes and Xl EXECUTED nodes. Let c be the smallest level number. The EXECUTED nodes at level l of T are sibling-paired if there is at most one EXECUTED node α at level l whose sibling node β is not EXECUTED. The aggregate number of ELIGIBLE nodes at step t is maximized if the EXECUTED nodes along each level of T are sibling-paired. A Schedule for Tree Structured computation is IC optimal if it is parent-oriented, i.e. , it always executes a node of a reduction-tree and its sibling in consecutive steps. Further, the algorithm is optimized to ensure the efficient use of the available client resources. For this, the eligible tasks stored in an array are retrieved in a First in First out fashion, representing queue model instead of stack model. A result of Rosenberg [5] characterizes optimization of IC scheduling algorithm. He proved that a schedule for reduction trees is IC optimal if, and only if, it is parent oriented, that is it always executes a node of the reduction tree and its sibling in consecutives steps. The graph for IC Optimal Scheduling is depicted in the following Fig 3. 4

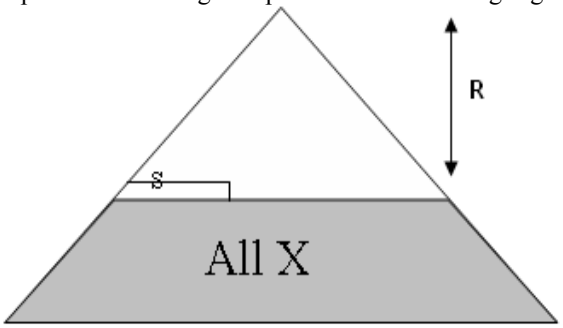

Fig. 7 Optimized IC Scheduling

S- Next eligible source node
R- No of levels executed
X- Executed Part

The steps in the ICOS algorithm are

1. Identify Eligible tasks. All unexecuted nodes having no parents, upon whose prior execution they depend, are eligible for execution
2. No task is allocated to clients until it becomes "Eligible"
3. When a client's System Configuration is high and CPU Usage, Memory Usage values are low, tasks are allocated for execution
4. If no task is eligible, no allocation is done
5. When a client returns a task result, identify the tasks that become eligible with the obtained results
6. Ensure that all tasks are executed eventually

The Optimized IC Scheduling Algorithm used for Avoidance of the problem of Gridlock is shown below.

8.1 Algorithm ICOS

Repeat until Halt
If the root of Tree is Eligible for execution
Then Execute the root; Halt

Else Determine the lowest level-number l of the Tree
that contains an Eligible Sibling-pair.
Execute the left-most Sibling-pair at level l

9. Conclusion

An Optimized IC Scheduling algorithm is developed to avoid the problem of Gridlock that occurs due to inter dependencies of tasks at various levels. The algorithm schedules the eligible tasks so as to maximize the number of eligible tasks at each step of computation, thus making maximum utilization of the available resources in the Grid Framework. Further the system can be enhanced to develop optimized algorithms to schedule tasks of butterfly computations that are present in Digital Signal Processing problems.

References

[1] Rajkumar Buyya and Kris Bubendorfer (eds.), Market Oriented Grid and Utility Computing, ISBN: 9780470287682, Wiley Press, New York, USA, 2009.

[2] Rajiv Ranjan, Aaron Harwood, Rajkumar Buyya, Peer-to-Peer Based Resource Discovery in Global Grids: A Tutorial, IEEE Communications Surveys and Tutorials, Volume 10, Number 2, Pages: 6-33, ISSN: 1553-877X, IEEE Communications Society Press, USA, 2008.

[3] Arnold L. Rosenberg (2004), "On Scheduling Mesh-Structured Computations for Internet-Based Computing", IEEE Transactions on Computers, vol 53, pp. 1176-1186.

[4] Joshy Joseph, Craig Fellenstein (2004), "Grid Computing", Pearson Education Publications

[5] D. Kondo, H. Casanova, E. Wing and F. Berman (2002), "Models and Scheduling guidelines for Global Computing Applications", Proceedings of International Parallel and Distributed Processing Symposium, pp 437 – 443

[6] Rosenberg, A. L (2003), "On Scheduling Mesh-Structured Computations for Internet", Proceedings of International Parallel and Distributed Processing Symposium, pp 115-139