Statistical Approach to Optimize Master-Worker Allocation in Grid Computing

S. R. Kodituwakku[1] and H. R. O. E. Dhayarathne[2]

Department of Statistics and Computer Science
Faculty of Science
University of Peradeniya
Sri Lanka

[1]salukak@pdn.ac.lk [2]iru049@yahoo.co.in

Abstract

We investigate the problem arising in allocating master and workers in the Master-Worker Paradigm. Although various methods have been proposed for Master-Worker allocation, optimal allocation is yet to be achieved. This paper proposes an extension to the generic Master-Worker architecture for achieving optimal allocation of masters and workers. The architecture is extended by introducing an additional utilizing layer in between the Grid application layer and the Grid service layer, which is called Master-worker optimization layer. The layer consists of a knowledge base and uses statistical knowledge for utilization of the resource allocation. The salient feature of this layer is that it uses the prior knowledge and numerical data relating to groups of individual computers for the decision making process. The paper describes the architecture of the extended Master-Worker architecture; its functionality, possible alternatives for the Master-worker optimization layer and assessment of the proposed method.

Keywords: *Grid computing, Grip Applications, Master-Worker Paradigm, Master-worker optimization layer, Statistical knowledge*

1. Introduction

Advances in hardware and software technologies lead to the widespread use of loosely coupled, powerful and low-cost networked components known as clusters. Grids are examples of such clusters and are widely used in achieving complex computations. Nodes involved in Grid computing collaborate themselves in an organized manner to achieve complex tasks. There are several paradigms which increase the effectiveness of this team work. One of them is the Master-Worker (MW) paradigm [1][2]. Master-Worker paradigm is one of the main paradigms frequently used in parallel applications. Further it illustrates ensembles of most of the problems that are being solved by parallel applications. Therefore it can be nominated as the most common natural programming method for many Grid applications today.

In the MW model, one node acts as the controlling master for the parallel application and sends pieces of work to worker nodes. The worker node performs some computation, and sends the result back to the master node. The master has a pool of work that needs to be processed. It simply assigns the next piece of work to the next worker that becomes available. In general, it allocates as many workers as possible to reduce the total execution time and workers are not released until the computation is completed. This strategy does not work well in all situations. Cluster middleware usually have some restrictions that limit the number of resources that a user may allocate for an application. Such scheduling strategies result in poor utilization of resources because not all the allocated workers kept busy. Several approaches have been proposed, therefore, to solve the Master-Worker allocation problem [3][4][5][6][7][8][9].

The *CONDO* [4] is a software framework that allows users to quickly and easily parallelize scientific computations using the master-worker paradigm on the computational grid. It provides both a top level interface to application software and a bottom level interface to existing grid computing toolkits. The necessary Grid services are provided by the CONDOR high-throughput computing system, and the MW-enabled application code is used to solve a combinatorial optimization problem of unprecedented complexity. It contains several focal points named MWDriver, MWTask, MWWorke and so forth. All these allow programmers to assign them offline.

Development and Tuning Framework of Master/Worker Applications [5] emphasizes features such as data distribution and the number of workers. They must be tuned properly to attain optimal adequate performance. In most cases such features cannot be tuned statically offline since they totally depend on the exact conditions of each separate execution. Even the same program is ran over same data it would differ according to the availability of Grid nodes. It proposes a dynamic tuning environment based on a theoretical model of Master/Worker behavior and allows for the adaptation of such applications to dynamic execution conditions.

By hiding the low level details, performance tuning is successfully been carried out on the fly. In this environment, programmers can design the application in quite a simple way without worrying about performance analysis or tuning, because dynamic performance tuning automatically takes care of these tasks. The performance model for evaluating the optimal number of workers has been integrated in that environment.

Adapting Master-worker paradigm for high throughput applications in grid environment [3] describes the problems exist in the previous cases in adapting the Master-Worker paradigm in high performance environment. A concept and evaluation of the system are presented. It provides a grid based implementation of master-worker paradigm where execution time of data streams is reduced by using a Real-Time Dispatcher (RTD) supported with high throughput data streams. It is based on separation of grid jobs initiation from execution, where they are treated as two concerns.

The allocation problem is identified as the Quality of Service (QoS) where the delay or the time factor plays a major role in two major apprehensions: Grid delay and Job delay. It clearly brings out the fact that Job delay can only be minimized by modifying the Grid application itself, but the Grid delay which occurs due to several environmental issues can be avoided or make minimum in essence of Job Submission or Scheduler Service. It is also related with current grid workload. The most important fact is that it cannot be solved offline; neither can be avoided for one certain time instant as it appears every time when a job is submitted.

The proposed solution uses the Master-Worker paradigm to allow entrustment of computation outside of a particular application to the grid without engaging job submission service and scheduler. Furthermore, it suggests the elaboration of Worker selection strategies based on data from the monitoring system and dynamic sizing of the worker pool based on predicted Master loads.

Other proposed approaches [6][7][8] also address the efficiency issue in allocating master and workers. However the efficiency and effectiveness of master-worker allocation is still an open research problem. Our objective is to improve the Master-Worker allocation for achieving high efficiency by keeping the allocated workers usefully busy. Therefore we propose an extension to the generic Master-Worker architecture. An optimization layer in between the Grid service layer and Grid application layer is introduced in order to allocate the Master and Workers in optimal manner.

2. Methods and Materials

In case of mutually independent tasks, the performance of the application only depends on the number of workers, because load balancing is dynamic and automatic. In case of dependent tasks, the performance of the application mainly depends on two factors: balanced computational load among workers and appropriate number of workers. The order of these factors is important, because changing the number of workers makes little sense if the computational load has not been previously balanced.

In both cases the proper number of workers depends on the granularity of tasks, resource cost and the relation between communication time, computation time and so forth. Therefore, the Master-Worker performance model consists of a strategy for achieving a balanced computational load, a strategy for adapting the number of workers and a linking expression that enables the combination of both.

The single master and multiple master approaches are used in MW framework. The multiple master approaches allow higher concurrency and increase efficiency. However, it is required to use optimal number of Masters and to achieve load balancing. There are two approaches to decide on number of masters to be allocated: Full Cover Problem and P-Master Problem.

Full Cover Problem is used to locate the minimum number of masters required to cover all the platform computing resources. The number of workers will be allocated first and then the masters will be appointed to utilize the corporation. The ultimate objective is to minimize the total cost rather than the number of masters used. It is found that this problem is NP-hard [10][11].

P-Master Problem is used to locate P masters to maximize the throughput of the platform. It is again found that this problem is NP-hard. When P is fixed, the P-master problem can be solved in polynomial time, because it can enumerate each possible set of candidate locations in polynomial time.

Since the large-scale distributed platforms are usually composed of shared resources contention will be caused by other applications running simultaneously on these resources. Consequently, the load and availability of the resources vary over time due to the unpredictable interactions of the users and the platform components. So the problem becomes more and more complex. Therefore, this paper focuses on adaptive scheduling strategies and mechanisms that enable a practical use of several masters.

Set Cover Problem

In a given universe *U* there exists a family of subsets S of *U*, a cover is a subfamily $C \leq S$ of sets whose union is ultimately *U*. In the considered minimum set covering decision problem, the input is a pair (*U, S*) and an integer *k*; k is the value which should be decided whether there is a set covering of size *k* or less. In the set covering optimization problem, the input is a pair *(U, S)*, and the task is to find a set covering which uses the fewest sets. Naturally the decision version of set covering is NP-complete, and the optimization version of set cover is NP-hard [11].

Greedy Approximation

Greedy Approximation is the proven best approximation for the set cover problem. When utilizing Greedy approximation for the set cover problem the basic procedure is as follows. At each and every step the set covering algorithm chooses sets based on one major rule: select the set which covers the largest number of uncovered elements.

Mathematically it is proved [11] that the algorithm achieves an approximation ratio of H(s) ; where s is the size of the largest set and H(n) is the n^{th} harmonic number defined as

$$H\,(n) = \sum_{k=1}^{n} \frac{1}{k} \leq \ln n \, + \, 1$$

When the Universe consists of $n \, = \, 2^{(k+1)} - \, 2$ elements the approximation ratio is $\log_2(n)/2$.

Load Balancing Through Data Distribution

The execution time of a Master/Worker application t, with N workers and a set of tasks that can be sequentially processed in time T, can be roughly bounded by the expressions,
[T/N + α] (lower bound) < t < [T + α] (upper bound), where α represents communication time.

Getting an execution time closer to the lower bound mainly depends on good load balancing among workers, which in turn relies on a good data distribution policy. Generally instead of distributing the whole set of tasks among workers and waiting for the results without any control over load balancing, the master will make a partial distribution by dividing this set of tasks into different portions called batches. Different strategies can be used to determine the batch sizes with the objective of getting better load balancing with little computation and communication overheads.

3. Methodology

Generally Master-Worker designs are NP-Hard problems. Therefore, relevant approximations would be the major motivational method to escape from these complexities. A knowledge base which keeps track of problems occurred and holds templates for future problems would ease the Grid application developer’s tasks. Additionally, the complexity of development of such applications would be reduced.

In an ideal Master-Worker application the total execution time will be equal to the sequential execution time evenly divided by the number of workers. This is based on the assumption that there is no communication cost that the application is executing on a dedicated and homogeneous platform that has achieved a perfect load balancing, and that the computation also scales ideally. In this ideal scenario, any available resource that can be assigned to the application must be assigned, because it will be efficiently used to improve the application's performance.

The main anxiety towards the allocation problem is recognized as the coupling ability of nodes, the performance aspects and the availability. Therefore, the knowledge about the nodes which are involved in Grid Applications is stored in an organized manner based on those features of concern. In order to do so an additional layer in between Grid Service Layer and Grid Application Layer is incorporated.

3.1 Conceptual Architecture of the Master Worker (MW) Optimization Layer

Architecture of the Conceptual Optimization Layer is depicted in Figure 1. Components of this layer are described here.

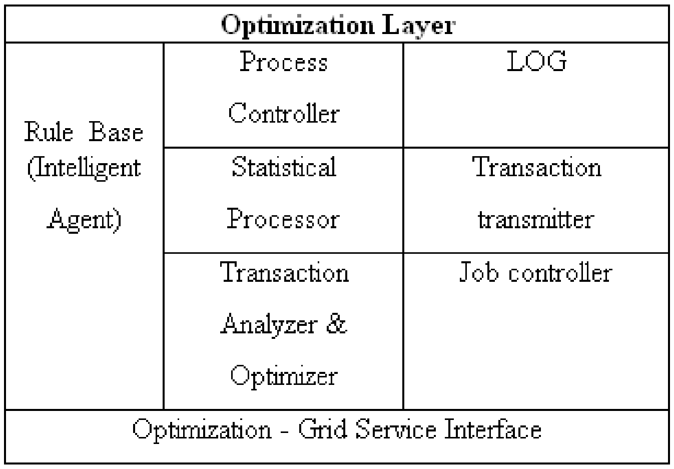
Optimization Layer
Rule Base (Intelligent Agent)
Process Controller
LOG
Statistical Processor
Transaction transmitter
Transaction Analyzer & Optimizer
Job controller
Optimization - Grid Service Interface

Figure 1 – Architecture of the Conceptual Optimization Layer

Rule Base (Intelligent Agent)

This is the most important part of the optimization process and it contains rules to be explored during the execution of Grid Applications. This is used in order to make the transaction analyzer decides which kind of task organizer to be employed. One basic part of the Rule base stores the information about Master-Worker relationships. It updates and keeps all the records of eligible Masters, Master-Worker relationship strengths and performance measurements periodically. Basically it is the knowledge about Masters and Workers that can be employed during a session.

Process Controller

The process controlling is a major task when the Grid environment is concerned. Therefore, the process controller component is introduced to the system. Its responsibility is to create and maintain processes corresponding to applications those are executed in the optimization layer. So it knows the status of the components that are being used and the knowledge about processes that will be available.

Furthermore, it must deal with the additional transactions which will be generated from processes and maintain them until they are killed. Its ultimate goal is to help the Optimization layer to achieve its efficiency and the effectiveness and to enhance the knowledge base.

Transaction Analyzer and Optimizer

This involves in deciding how the procedure or procedures in a transaction must be sent to the Grid for better efficiency. This is again directly corresponds to the rule base and some previous results.

LOG

Results of the jobs, their respective tasks and transactions submitted to the Grid are recorded in this component.

Job Controller

Job Controller submits the jobs which are obtained by analyzer and optimizer to the Grid service layer in order to be executed as parallel jobs. It keeps correspondence with the process transactions related to these jobs. When a job is completed its controller should record the information about the job in the log and should deal with the process controller.

Transaction Transmitter

This would verify at processes controller which transactions are recently arrived from processes. Depending on the rule, for example priority, at the reception of optimized results it should create a separate job controller which must be responsible for submitting jobs or tasks to the Grid.

Statistical Processor

Statistical Processor holds the responsibility of extracting relevant statistical information from the applications to utilize the resource allocations and information of the Grid to be used in future requirement or from the experts to revise the system.

Since the optimization layer should be able to deal with users and Grid additional implementations of interfaces are also taken into consideration. Extended architecture with Grid Service Layer and the Grid Application Layer is shown in Figure 2.

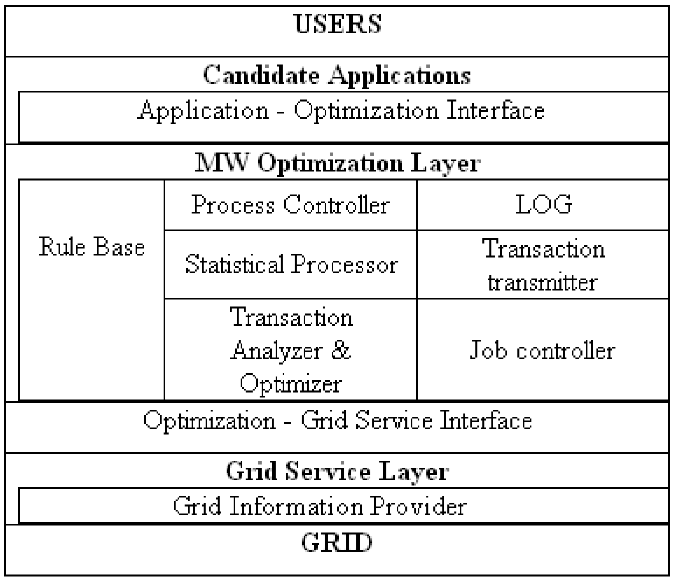

Figure 2 – Architecture of the Conceptual Optimization Layer with collaboration with Grid Service Layer and the Grid Application Layer

3.2 Statistical Approaches in the Master-Worker Optimization Layer

When considering the statistical constituent of the optimization layer the basic components that deal with it are the Rule Base, the Statistical Processor, and the Transaction Analyzer and Optimizer.
Several statistical methods can be adapted and utilized in the optimization layer as statistical approaches. All of them together can be collectively used to obtain a generic solution for the Optimization layer. Grid Applications are not only oriented in optimizing the allocation. Other Grid Application specific factors could also be considered

depending on the circumstances. These factors might be concentrated on Worker allocation specifically and then allocating the Masters optimally, or on the other hand allocating a specific number of Masters first and the appointing relevant Workers. Since it is hard to implement a single specific approach several possible approaches are considered.

3.3 Set Cover Problem with a Greedy Approximation to Solve Optimal Master Allocation

In computing and complexity theory, the Set Cover Problem can be seen as a major approach to select the optimal number of subsets those can cover a given set. The objective is to minimize the number of subsets being selected.

In this context, the minimum set cover problem is being considered and it can be formulated as the following integer linear program (ILP).

$$\text{Minimize } \sum_{s \in S} c(S) x_s \quad \text{(minimize the total cost)}$$

$$\text{subject to } \sum_{s : e \in S} x_s \geq 1 \quad \text{for all } e \in U \text{ (cover every element of the universe)}$$

$$x_s \in \{0,1\} \quad \text{for all } s \in S \text{ (every set is either in the set cover or not)}$$

This ILP belongs to the more general class of ILPs for covering problems. The integrality gap of this ILP is at most log *n*. So its relaxation gives a factor-log *n* approximation algorithm for the minimum set cover problem, where *n* is the size of the universe.

3.3.1 Approach to solve Full Cover Problem

To solve the Master-Worker allocation problem, the Worker allocation is first taken place. Then the covering problem is solved to achieve the Mastering of those Workers. The Rule Base will be the prior knowledge for solving the problem, where a binary matrix corresponding to the coupling ability and the mastering ability is being introduced in order to ease the solvability. A sample Master-Worker matrix is shown below

Worker / Master	1	2	3	4	5	6	7	8	9	10	.	.	.
A	1	0	0	0	1	1	0	0	1	1	.	.	.
B	0	0	1	0	1	0	0	1	1	0			
C	1	0	1	1	0	0	0	1	0	0			
D	1	0	0	1	0	1	1	0	1	1			
.	.												
.	.												

This Master-Worker combination pattern would be useful in solving the allocation so that the Masters and the grouped Workers are able to achieve its task with a higher coupling ability and with better communication efficiency. Flow Diagram for the Master-Worker allocation is shown in Figure 3.

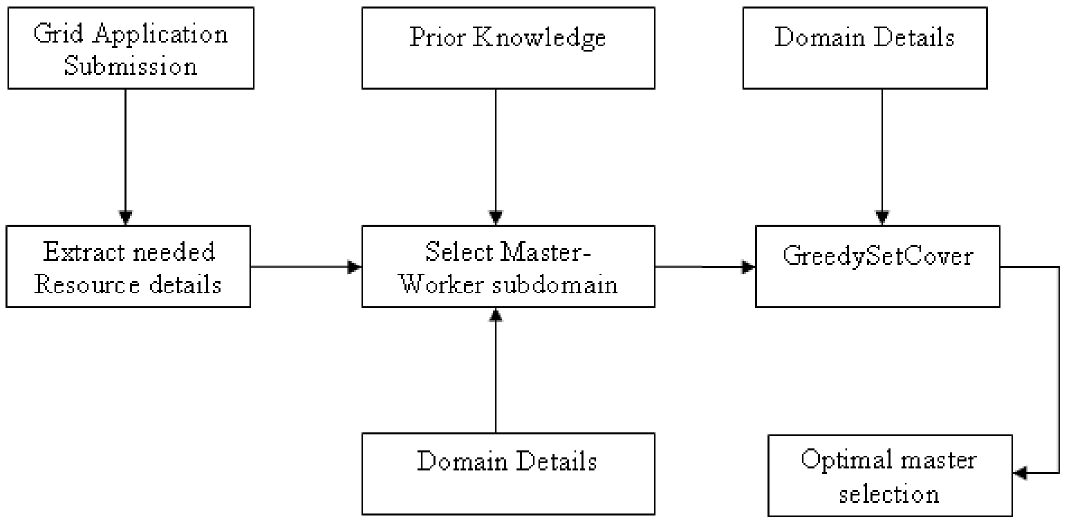
Grid Application Submission
Prior Knowledge
Domain Details
Extract needed Resource details
Select Master-Worker subdomain
GreedySetCover
Domain Details
Optimal master selection

Figure 3 - Basic flow of solving the Optimal Master Allocation problem

Furthermore, the Master-Worker allocation discussed can be enhanced by using the matrix elements to represent the strength of the relationship among the Workers rather than the binary value which represents the coupling ability. The knowledge base will store the matrix as in the previous approach and the weights or the similarity (S) matrix, which represents the coupling factor with a weighted value. In this method, the Workers are clustered into Groups and one of the workers can be granted as the Master. This would not give the optimal Master allocation, but it would give the best grouping so that all the Workers in a group is much efficient in assigning tasks as a group.

3.4 Worker Concerned Greedy Optimal Master Allocation

In this approach, the solving the problem gets somewhat complicated, because it concerns the Masters that can be used to cover the Worker, in contradictory with the above mentioned approach. However, it is not difficult to extract the details since the same matrix is being selected. Only difference is that the matrix is Worker based rather than Master based. The Worker with the least number of possible Masters will be selected first and the Master covering the highest number of Workers is being picked from those Masters. This increases the probability of the selected Master could be included in optimal solution. The load balancing can be considered in the first stage of appointing.

This would result in better allocation for higher loads. Basic flow of solving the Worker Concerned Optimal Master Allocation is shown in Figure 4.

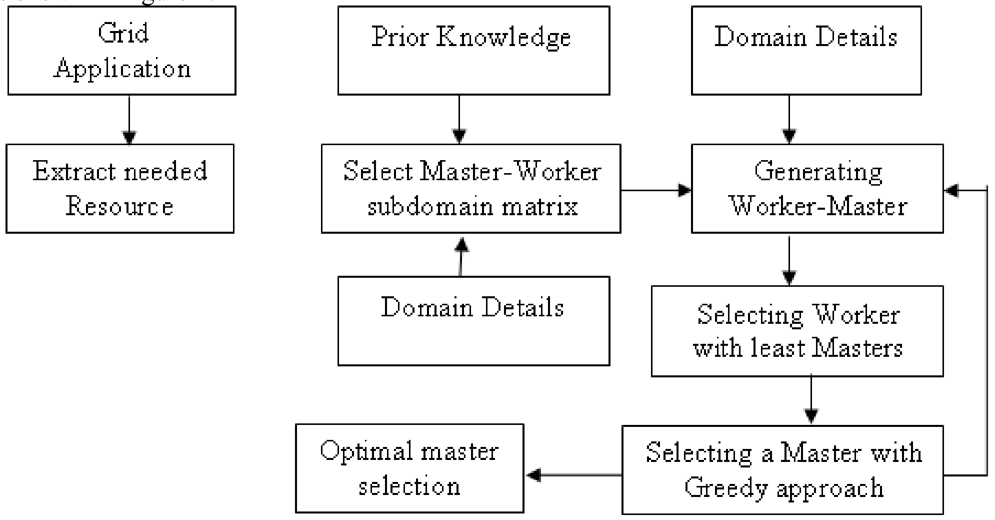

Figure 4 – Architecture of the Conceptual Optimization Layer with collaboration with Grid Service Layer and the Grid Application Layer

3.5 Achieving Load Balancing

Master and Workers collaborate by means of communication. Therefore, higher number of communications take place in some masters while the other masters are not involved in that much of communication. The proposed approach selects the Masters with a greedy approach where the selection is based upon the number of Workers a particular Master can cover. In each iteration, the Master that covers the maximum number of Workers that are still uncovered by previous selections is selected.

3.5.1 Minimum Cover Approach

The average coverage factor will be calculated so that in earlier stages the over allocation will not take place. The Workers are ranked according to the coverage capability of Masters where the Worker which can be covered by least number of Masters will assign the first priority.

First the Master covering least Workers in the selected Master group will be chosen. Then the Workers that can be covered using that Master will be removed from the list in order of the rank given above and the maximum coverage is taken as the average factor.

The Workers already being selected will be removed from the Master-Worker matrix which will be used in the next iteration. The same approach will be used in order to appoint the Workers to each Master. At the end of the iterations if there exists any excessive Workers they will be appointed to Masters which are capable of handling them according to number of Workers they already hold. Such workers are assigned to the master that holds minimum number of workers.

4. Results and Discussion

In the proposed approach, the heterogeneities in work stations are being analyzed and stored as a knowledge base to create this resource sets concept. Therefore, the allocating master is aware of the workers allocated beneath it and they are the sets which have shown most efficiency with respect to possible sets.

Not only that the knowledge base will be updated according to relevant domain. So this works as an intelligent agent in allocating Masters and assigning Workers relevantly. In case of online appointing, the same knowledge base can be used to couple the Workers into running Masters in order to ensure least Grid delay.

If the optimal number of masters is k to cover the given resource set with n points, then it can easily be shown that the time complexity to solve this problem is O(n). Additionally the space complexity is O(1) when the space to store the sets is neglected; else it can be stated as O(n). Generally the maximum complexities of time and space would be O(n) when the set cover problem is concerned. If the total proposed masters are r and the selected masters are k, the probability that the selected permutation be the optimal solution is $1 / P_k^r$. Greedy algorithm does not always yield an optimal solution but guarantees an approximate solution.

Let $C^* = |I^*|$ be the size of any optimal solution, and $C = |IG|$ denote the size of the solution reported by this greedy strategy. Then the following bounds can be obtained:

$$C^* \leq CG \leq C^*(1 + \log n)$$

Mathematically it is proved [11] that the greedy approximation algorithm achieves an approximation ratio of H(s), where s is the size of the largest set and H(n) is the nth harmonic number defined as $H(n) = \sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1$.

When the Universe consists of $n = 2^{(i+1)} - 2$ elements the approximation ratio is $\log_2(n)/2$. So it stands as a good approximation for the full cover problem.

Supposing there are m sets covering everything. After t choices, Greedy has at most $(1 - 1/m)^t$ fraction uncovered.

Proof:

At the beginning, there are m sets covering all, so there is a set S_i covering at least 1/m fraction of the elements.

After Greedy’s first choice, the fraction of uncovered elements is less than $1 - 1/m$.

In the next step, there are still m sets covering all uncovered elements, so there is a set which covers at least 1/m fraction of them. That is, after Greedy’s second step, the fraction of uncovered elements is less than $(1 - 1/\mathrm{m})^2$.

Similarly, after t time-steps, Greedy has at most $(1 - 1/\mathrm{m})^t$ fraction uncovered.

According to Corollary [12] Greedy is a factor [ln *n*] approximation algorithm. This is the counting version for calculating the value, not the fraction.

Proof:

Optmaly sets cover everything. Once less than 1/n fraction uncovered, it is done.

$$(1 - 1/\mathrm{Opt})^t < (e^{-1/Opt})^t$$
$$(e^{-1/Opt}) < 1/n$$
$$t > \ln n \cdot \mathrm{Opt}$$

The optimization solution taken for the set cover problem is the greedy approach. In most of the time it doesn’t yield an optimal solution but gives a guaranteed approximation. Furthermore, since the master can also be one of the workers itself the overhead of allocating that master is not a concern when the resource waste is concerned. Heterogeneities and required grouping can be embedded with the knowledge-base and that’s a major advantage as long as the concept is distressed. Since it guaranties better performance continuously the Grid delay is highly minimized.

In this method, the introduced overhead is completely flexible and developers just have to consider on task allocation. Grid delay will be the only factor that can be reduced when the efficiency measures are concerned from the resource side rather than the programmers’ side. So this whole effort is to reduce that avoidable delay. Without just assigning Masters and Workers the concern is to have knowledge about them in order to increase the efficiency and effectiveness.

The main issue of the proposed method is the increased complexity of the grid load, but it would not be a significant overhead since grid applications naturally runs for a long time period. Therefore this method would reduce running delays even a relatively small initial delay exists.

5. Conclusions

In the existing context, the Master- Worker allocation is not backed up by any prior knowledge concerns and it results in Grid delays. The proposed statistical based approach can be used to avoid that situation.

With a knowledge base or a rule base the existing mapping suitability, coupling abilities, distances and communication barriers are being considered as knowledge. They are being introduced to ease the causes which create Grid delays. Therefore this can be seen as a solution for existing complex Mater-Worker allocation problems in spite of the degree of the Master-Worker relationship. It can be easily adapted for any of the existing Master-Worker paradigms raging from Single-Master, Optimal-Masters, K-Masters or even hierarchical-Masters approaches. Additionally, it can be applied as a collective generic solution for the optimization of Grid resource allocation and to provide better utilization and efficiency.

References

[1] C. Banino, *Scalability Limitations of the Master-Worker,Paradigm for Grid Computing*, Norwegian University of Science and Technology (NTNU)

[2] E. Cesar , A. Moreno, J. Sorribes and E. Luq, *Modeling Master/Worker applications for automatic performance tuning*, 2006.

[3] J. Pieczykolan, L. Dutka, K. Korcyl, R. Slota, T. Kryza and J. Kitowski, *Adapting master-worker paradigm for high throughput applications in grid environment*, http://para08.idi.ntnu.no/docs/submission_55.pdf, 2007.

[4] CONDOR-PVM (Parallel Virtual Machine) home page http://www.cs.wisc.edu/condor/pvm/, 2010.

[5] A Morajko, E. César, P. Caymes-Scutari, J. G. Mesa, G. Costa, T. Margalef, J. Sorribes and E. Luque, *Development and Tuning Framework of Master/Worker Applications,*2006

[6] A. Morajko, E. Cesar, P. Caymes-Scutari, T. Margalef, Joan Sorribes, Emilio Luque, *Automatic Tuning of Master/Worker Applications*. LNCS, vol. 3648, Springer-Verlag, 2005, pp. 95–103.

[7] A. Morajko, P. Caymes, Toma`s Margalef, Emilio Luque, *Automatic Tuning of Data Distribution Using Factoring in Master/Worker Applications*. LNCS, vol. 3515/2005, Springer-Verlag, 2005, pp. 132–139.

[8] E. Heymann, M.A. Senar, E. Luque, M. Livny, *Efficient resource management applied to Master–Worker applications*, Journal of Parallel and Distributed Computing 64, 2004, pp: 767–773.

[9] J. P. Goux, S. Kulkarni, J. Linderoth and M. Yoder, *An Enabling Framework for Master-Worker Applications on the Computational Grid*, Conf. Proc. of the Ninth IEEE Symposium on High Performance Distributed Computing, Pittsburg, PA, 2000, pp. 43-60.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Cambridge*, Mass.: MIT Press and McGraw-Hill, 2001.

[11] D. S. Hochba, Approximation Algorithms for NP-Hard Problems, University of California, Berkeley, ACM New York, NY, USA, 1997.