International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 6 November 2015
International Manuscript ID : 2249054XV5I6112015-02

RELIABLE AND EFFICIENT TASK SCHEDULING ALGORITHM BASED ON GENETIC ALGORITHM IN CLOUD COMPUTING ENVIRONMENT

Marish Kumar
Research Scholar
Department of Computer Science and Engineering
National Institute of Technical Teachers Training & Research
Chandigarh, India

Amit Doegar
Assistant Professor
Department of Computer Science and Engineering
National Institute of Technical Teachers Training & Research
Chandigarh, India

ABSTRACT

This research task is focused on the developed and implementation of a unique and effective approach in the cloud infrastructure for the higher throughput and less cost in the job scheduling. The proposed algorithm is implemented and integrated using genetic algorithm for optimization of the results including the cost and performance factor. The system is generating efficient results in terms of the optimal solution when executed using genetic algorithm. In this approach, the technique is efficient also in terms of the execution and turnaround time despite of the number of iterations. The limitations may be included regarding the proposed work in terms of its further enhancement using assorted metaheuristics. The proposed system gives better results in executed using genetic algorithm that is one of the prominent metaheuristic techniques.

Keywords – Multiprocessor Job Scheduling, Scheduling in Cloud, Cloud Computing

INTRODUCTION

Real-time multiprocessor systems are now commonplace. Designs range from single-chip architectures, with a modest number of processors, to large-scale signal-processing systems, such as synthetic-aperture radar systems. For uniprocessor systems, the problem of ensuring that deadline constraints are met has been widely studied: effective scheduling algorithms that take into account the many complexities that arise in real systems (e.g., synchronization costs, system overheads, etc.) are well understood. In contrast, researchers are just beginning to understand the trade-offs that exist in multiprocessor systems.

Traditionally, there have been two approaches for scheduling periodic task systems on multiprocessors: partitioning and global scheduling. In global scheduling, all eligible tasks are stored in a single priority-ordered queue; the global scheduler selects for execution the highest priority tasks from this queue. Unfortunately, using this approach with optimal uniprocessor scheduling algorithms, such as the rate-monotonic (RM) and earliest-deadline-first (EDF) algorithms may result in arbitrarily low processor utilization in multiprocessor systems . However, recent research on proportionate fair (Pfair) scheduling has shown considerable promise in that it has produced the only known optimal method for scheduling periodic tasks on multiprocessors.

In partitioning, each task is assigned to a single processor, on which each of its jobs will execute, and processors are scheduled independently. The main advantage of partitioning approaches is that they reduce a multiprocessor scheduling problem to a set of uniprocessor ones. Unfortunately, partitioning has two negative consequences. First, finding an optimal assignment of tasks to processors is a bin-packing problem, which is NP-hard in the strong sense. Thus, tasks are usually partitioned using non-optimal heuristics. Second, as shown later, task systems exist that are schedulable if and only if tasks are not partitioned. Still, partitioning approaches are widely used by system designers.

In addition to the above approaches, we consider a new "middle" approach in which each job is assigned to a single processor, while a task is allowed to migrate. In other words, inter-processor task migration is permitted only at job boundaries. We believe that migration is eschewed in the design of multiprocessor real-time systems because its true cost in terms of the final system produced is not well understood. As a step towards understanding this cost, we present a new taxonomy that ranks scheduling schemes along the following two dimensions:

P fair scheduling - In recent years, much research has been done on global multiprocessor scheduling algorithms that ensure fairness. Proportionate-fair (Pfair) scheduling, proposed by Baruah et al. , is presently the only known optimal method for scheduling recurrent real-time tasks on a multiprocessor system. Under Pfair scheduling, each task is assigned a weight that specifies the rate at which that task should execute: a task with weight w would ideally receive w · L units of processor time over any interval of length L. Under Pfair scheduling, tasks are scheduled according to a fixed-size allocation quantum so that deviation from an ideal allocation is strictly bounded.Currently, three optimal Pfair scheduling algorithms are known: PF , PF, and PD2 . Of these algorithms, PD2 is the most recently developed and the most efficient.

The primary advantage of Pfair scheduling over partitioning is the ability to schedule any feasible periodic, sporadic, or rate-based task system[3]. Hence, Pfair scheduling algorithms can seamlessly handle dynamic events, such as tasks leaving and joining a system. Furthermore, fair multiprocessor scheduling algorithms are becoming more popular due to the proliferation of web and multimedia applications. For instance, Ensim Corp., an Internet service provider, has deployed fair multiprocessor scheduling algorithms in its product line.

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 6 November 2015
International Manuscript ID : 2249054XV5I6112015-02

The main disadvantage of Pfair scheduling is degraded processor affinity. Processor affinity refers to the tendency of tasks to execute faster when repeatedly scheduled on the same processor. This tendency is usually the result of per-processor first-level caching. Preemptions and migrations, both of which tend to occur frequently under Pfair scheduling, limit the effectiveness of these first-level caches and can lead to increased execution times due to cache misses. On the other hand, under partitioning with EDF, there is no migration and the number of preemptions on a processor is bounded by the number of jobs on that processor (assuming independent tasks).

Layer-Based Scheduling Algorithms - There has been a lot of research regarding scheduling algorithms for independent M-Tasks. However, these scheduling algorithms cannot cope with precedence constraints between M-Tasks. This limitation can be avoided using layer-based scheduling algorithms3 for MTasks with precedence constraints. These algorithms utilize a shrinking phase and a layering phase to decompose an M-Task dag into sets of independent M-Tasks, called layers. The subsequent layer scheduling phase computes a schedule for each layer in isolation. Our extension strategy enables the combination of the shrinking phase, the layering phase and the assembling phase with a scheduling algorithm for independent M-Tasks in the layer scheduling phase. Therefore, any scheduling algorithm for independent M-Tasks can be extended to support M-Task dags.

Layer Scheduling Algorithms - In this phase an M-Task schedule is computed for each constructed layer V_{Li} , i = 1, . . . , l in isolation. In the following we omit the index i and use V_L for the layer to be scheduled.

Two L-Level determines the total execution time for each possible partitioning of the set of available processors into K, K = 1, . . . , min(P, |VL|) subgroups $\hat{g}_{K},1, \ldots \hat{g}_{k,k}$ of about equal size[3]. The schedule for each of these partitionings is computed by adopting a list scheduling heuristic. In each step of this heuristic the M-Task v € V_L is assigned to group $\hat{g}*$ €{$\hat{g}_{K,1}, \ldots \hat{g}_{K,K}$ }, where $\hat{g}_K$ is the first subgroup becoming available and v is the M-Task with the largest execution time. The final processor groups g1, . . . , g_{K*} are computed by a subsequent group adjustment step from the groups ˆg $_{K*}$,1, . . . , ˆgk*,k* , where $_{K*}$ denotes the partitioning resulting in a minimum runtime.

Two L-Tree starts by constructing a tree for each M-Task *v* € VL consisting of a single node8. A dynamic programming approach is used to find all unordered pairs of trees $\{t_1, t_2\}$ with an equal depth and disjoint sets of M-Tasks. For each pair $\{t_1, t_2\}$ a new tree t with a new root node and children t_1 and t_2 is created. Each tree represents a schedule of the contained M-Tasks. The inner nodes of the trees are annotated with a cost table containing the execution time of the whole subtree for all possible processor group sizes $g^s = 1, \ldots, P$. A second annotation defines whether the schedules represented by the children of the node should be executed one after other or in parallel on disjoint processor groups. Finally, a set of trees each containing all nodes of the current layer is constructed, where each such tree symbolizes a different schedule. The output schedule of the algorithm is constructed from the tree which admits the minimum execution time.

LLREF Scheduling Algorithm Model - We consider global scheduling, where task migration is not restricted, on an SMP system with M identical processors. We consider the application to consist of a set of tasks, denoted $T = \{T_1, T_2, ..., T_N\}$. Tasks are assumed to arrive periodically at their release times r_i. Each task T_i has an execution time c_i, and a relative deadline d_i which is the same as its period pi. The utilization ui of a task T_i is defined as c_i/d_i and is assumed to be less than 1. We assume that tasks may be preempted at any time, and are independent, i.e., they do not share resources or have any precedences.

GENETIC ALGORITHM

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics (Fraser, 1957; Bremermann, 1958; Holland, 1975). We start with a brief introduction to simple genetic algorithms and associated terminology.

GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves.

To evolve good solutions and to implement natural selection, we need a measure for distinguishing good solutions from bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness measure must determine a candidate solution's relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions.

Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time. Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, we can start to evolve solutions to the search problem using the following steps:

Initialization. The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated.

Evaluation. Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.

Selection. Selection allocates more copies of those solutions with higher fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea, including roulette-wheel selection, stochastic universal selection, ranking selection and tournament selection, some of which are described in the next section.

Recombination. Recombination combines parts of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this (some of which are discussed in the next section), and competent performance depends on a properly designed recombination mechanism. The offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner (Goldberg, 2002).

Mutation. While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution.

Replacement. The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement and steady-state replacement methods are used in GAs.

Goldberg (1983, 1999a, 2002) has likened GAs to mechanistic versions of certain modes of human innovation and has shown that these operators when analyzed individually are ineffective, but when combined together they can work well. This aspect has been explained with the concepts of the fundamental intuition and innovation intuition. The same study compares a combination of selection and mutation to continual improvement (a form of hill climbing), and the combination of selection and recombination to innovation (crossfertilizing). These analogies have been used to develop a design-decomposition methodology and so-called competent GAs—that solve hard problems quickly, reliably, and accurately—both of which are discussed in the subsequent sections.

KEY POINTS OF THE SYSTEM

- The proposed system is generating efficient results in terms of the optimal solution when executed using genetic algorithm.
- The proposed technique is efficient also in terms of the execution and turnaround time despite of the number of iterations
- The limitations may be included regarding the proposed work in terms of its further enhancement using assorted metaheuristics.
- The proposed system may give better results in executed using simulated annealing that is one of the prominent metaheuristic technique.

PLATFROM SETUP FOR THE IMPLEMENTATION

- JDK
- Eclipse IDE
- CloudSim
- GridSim
- JUnit
- JCharts

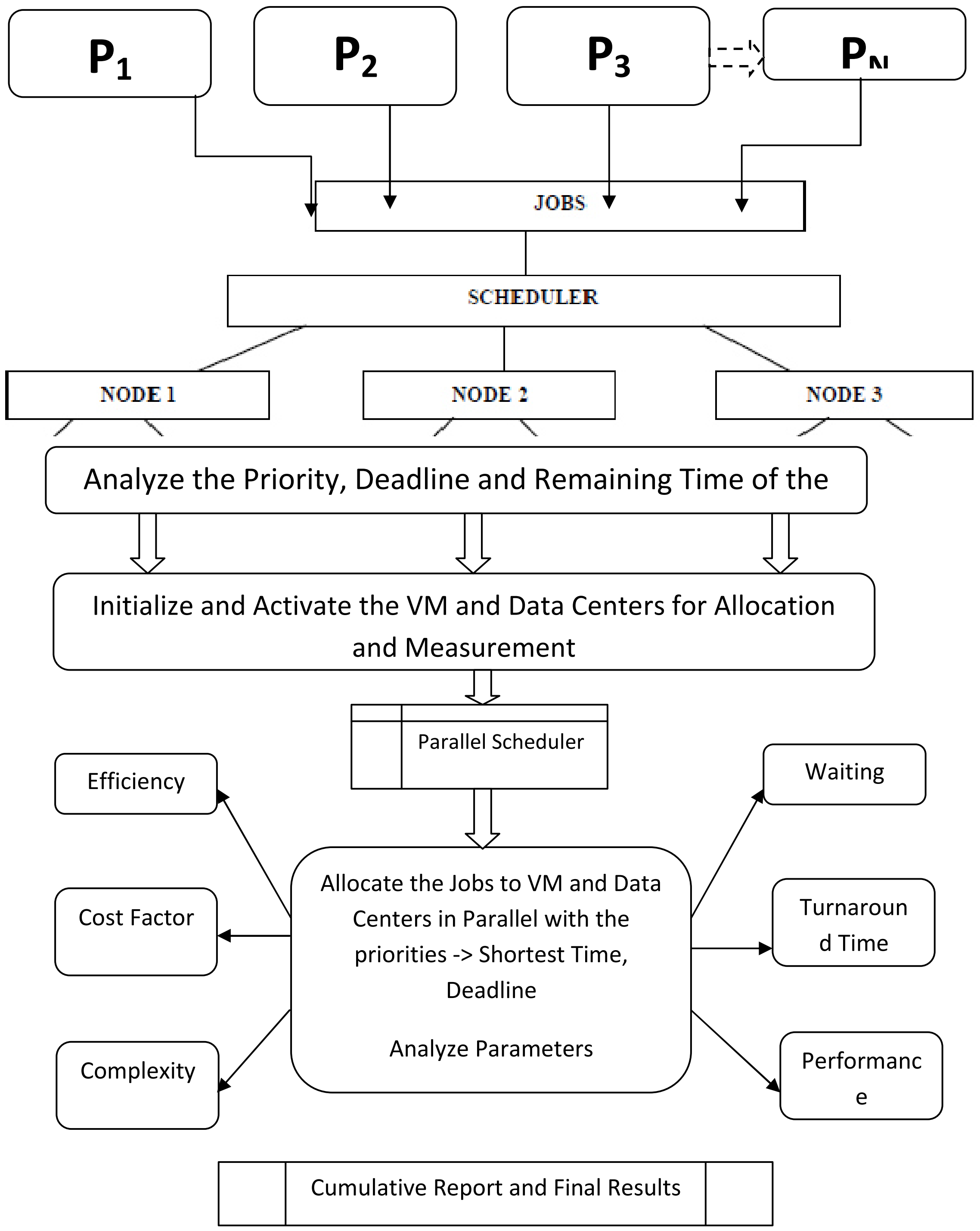

P1
P2
P3
PN
JOBS
SCHEDULER
NODE 1
NODE 2
NODE 3
Analyze the Priority, Deadline and Remaining Time of the
Initialize and Activate the VM and Data Centers for Allocation and Measurement
Parallel Scheduler
Efficiency
Cost Factor
Complexity
Allocate the Jobs to VM and Data Centers in Parallel with the priorities -> Shortest Time, Deadline
Analyze Parameters
Waiting
Turnaround Time
Performance
Cumulative Report and Final Results

Fig. 1 – Allocation of Tasks to VM and related aspects

IMPLEMENTATION, RESULTS AND DISCUSSION

The implementation is done in Eclipse IDE with Cloud and Grid Simulators in the External JAR Libraries for compatibility with Cloud and Grid Infrastructure. It is found and concluded from the results that EDSRTF algorithm is having effective results in cumulative way if we consider all the parameters in investigation as a whole level.

- The existing and base multiprocessor scheduling approaches are having huge pitfalls more execution time
- The basic solution obtained without swarm intelligence is not efficient in terms of the turnaround time and optimal results
- To investigate the drawbacks and shortcomings in the classical multiprocessor scheduling
- To propose and implement a novel technique for the simulation of genetic algorithm based multiprocessor scheduling

PROPOSED ALGORITHM

1. *To design the matrices and blocks of the computation cost as well as communication.*
2. *Assignment of priorities to tasks.*
3. *Calculate EST=0 & EFT=0 of task 1 on each processor, set process avail time for p1,p2,p3=0. Assign processor to task1 on which EFT is minimum*
   either p1=EFT (T1or task1) or p2=EFT (T1) or P3=EFT(T1)
4. *Activation of the Genetic Algorithm and its aspects*
   a. *for t2 to tn repeat above*
5. *Calculate if (parent of t1 executed on p1)*
6. *EST (tn) on p1=max (parent executed)*
   a. *p2=max (parent completion+Communication cost), processor avail time)*
7. *EST (tn) on p3= same as above*
8. *EFT=EST+Computation cost*
9. *Effective Comparison between the classical approach and proposed Approach*

RESEARCH METHODOLOGY

- Collection of the Training Data Set for Analysis

- Implementation of the Computation Cost Matrix
- Applying Genetic Algorithm on the Multiprocessor Scheduling
- Applying the proposed model on the Training data set
- Fetch Results
- Data Interpretation

In the complete implementation and research task, following aspects and reference is used regarding classical and proposed approach

Classical / Existing Approach -

GAMP (Greedy Approach Based Multiprocessor Scheduling)

Proposed Approach -

GAMBS (Genetic Algorithm Metaheuristic Based Scheduling)

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 6 November 2015
International Manuscript ID : 2249054XV5I6112015-02

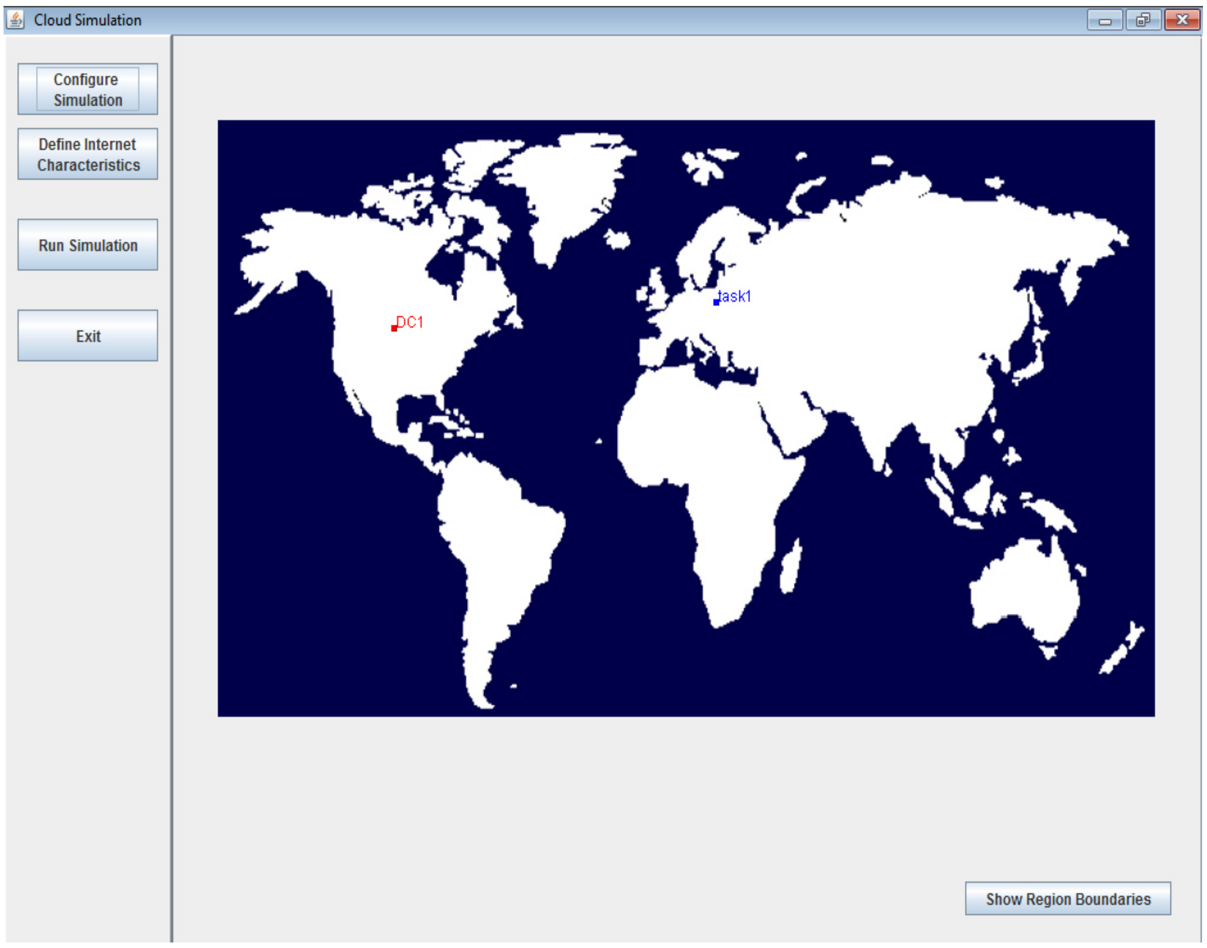
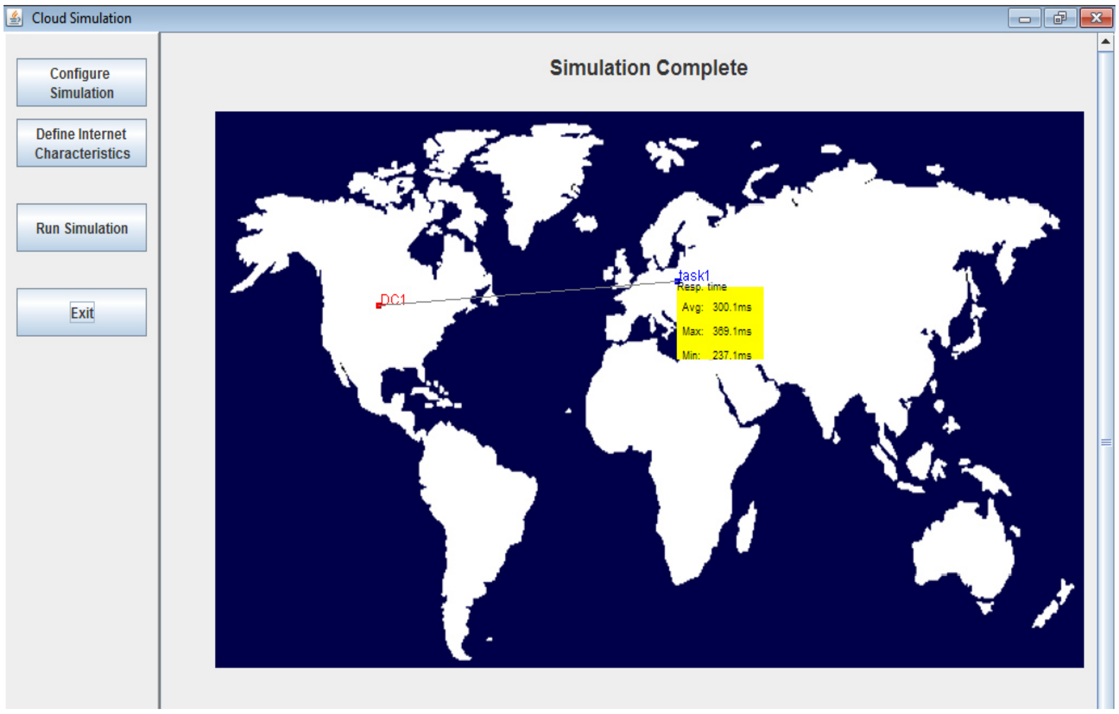
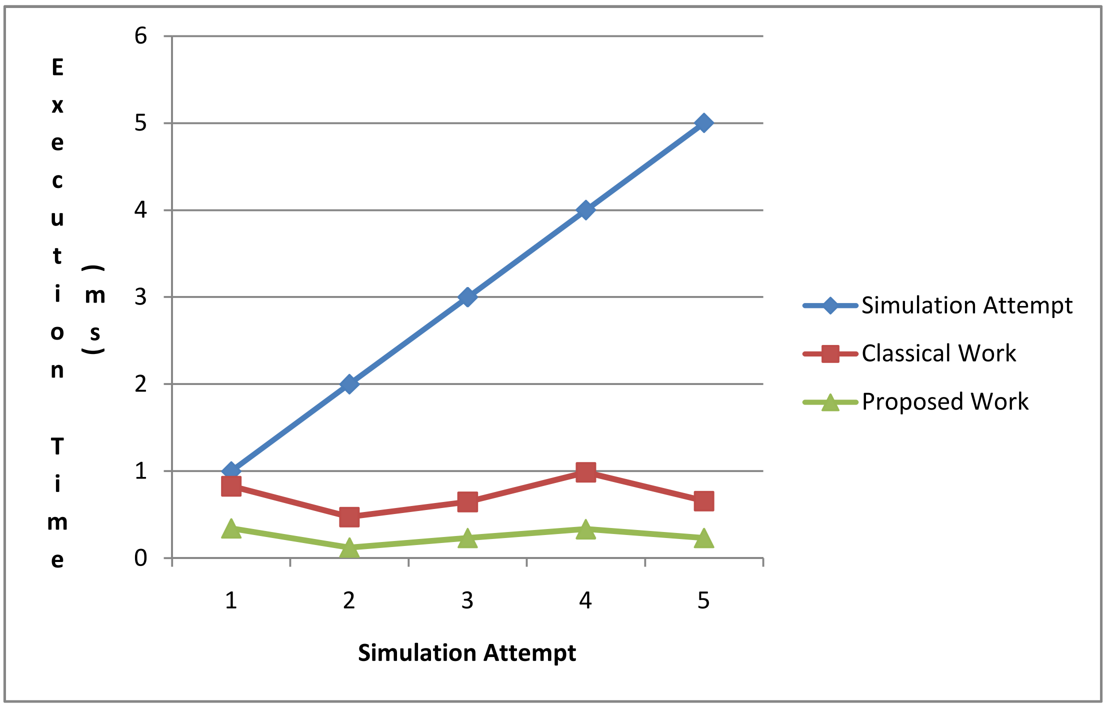


Fig. 2 – Simulation Scenario



Avg: 300.1ms
Max: 369.1ms
Min: 237.1ms

Fig. 3 – Simulation Completion Status



International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 6 November 2015
International Manuscript ID : 2249054XV5I6112015-02

Figure 4 - Line Graph Analysis of the Classical and Proposed Approach

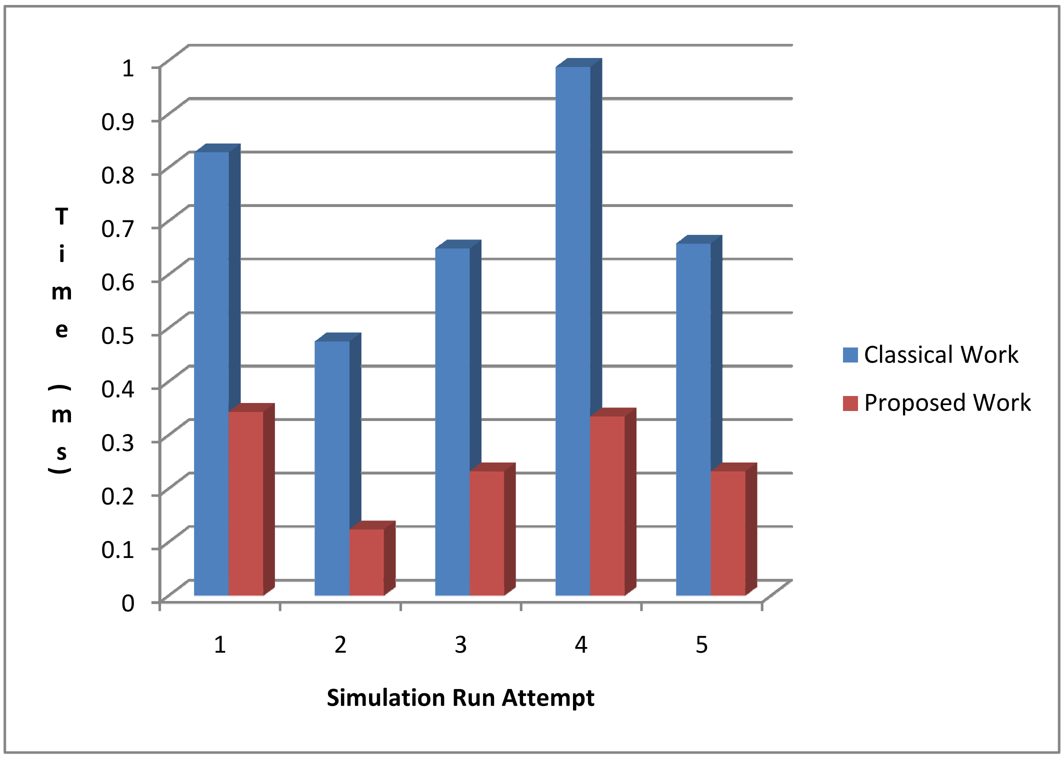


Figure 5 – Bar Graph Analysis of the Classical and Proposed Approach

Simulation Attempt	Classical Work	Proposed Work
1	89	70
2	78	60
3	67	59
4	76	40
5	49	20

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 6 November 2015
International Manuscript ID : 2249054XV5I6112015-02

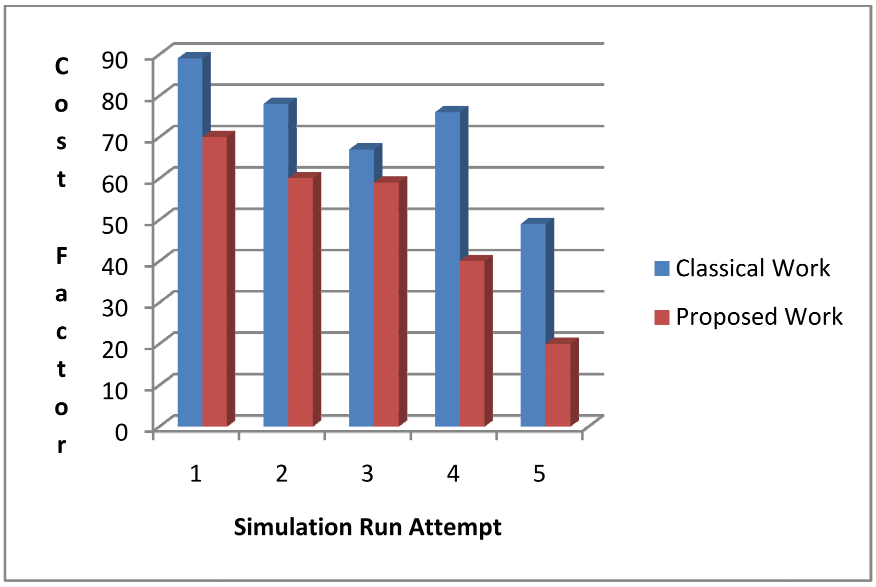


Figure 6 – Cost Factor Graph Analysis of the Classical and Proposed Approach

Classical / Existing Approach is not having effectiveness and efficiency as compared to GA based approach. The classical work is taken as the implementation without integration of metaheuristic based simulation. Tasks executed without GA and then with the integration of GA to evaluate the efficiency and related cost factor.

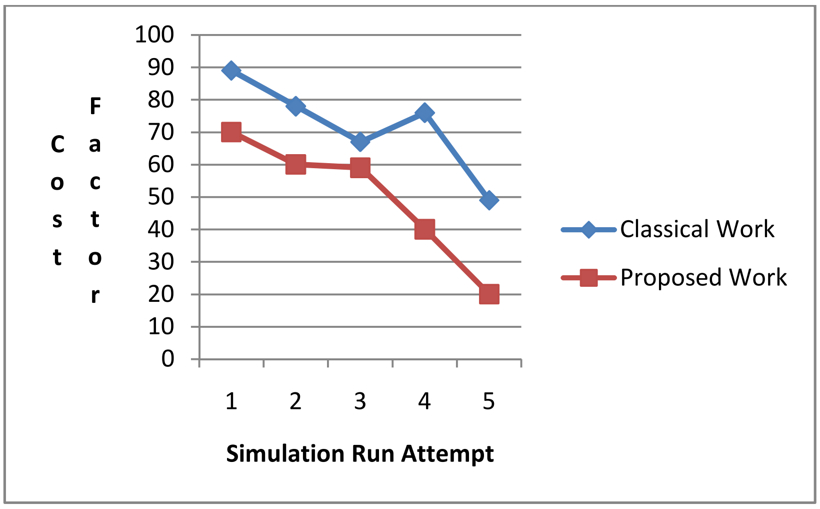


Figure 7 – Cost Factor Line Graph Analysis of the Approaches

It is evident from the graphical results that the cost factor in the proposed approach is *very less as compared to the classical approach. The execution time in the classical work is taking higher units as compared to the proposed work.*

Tasks executed without GA and then with the integration of GA to evaluate the efficiency and related cost factor.

Table 1 - Tabular Comparison of the Results Obtained

	W	T	E	P	C	CF
FCFS	10	50	75	85	690	35
LJF	14	550	20	40	20	300
EDSRTF	8	20	70	87	680	20
GA	7	18	95	96	670	18

It is evident from the simulation results and Table 1 that the cumulative result based on all the parameters are effective and better in the proposed approach name GA.

W – Waiting Time
The amount of time a process has been waiting in the ready queue in the process of execution.

T – Turnaround Time
Amount of time taken to complete a particular process.

E – Efficiency
The number of process that completes its execution per time unit.

P – Performance
Performance (P) is directly associated with the degree of Efficiency (e)

C – Complexity

Complexity of a structured program is defined with reference to the control flow graph of the program, a directed graph containing the basic blocks of the program, with an edge between two basic blocks if control may pass from the first to the second.

The complexity C is then defined as

$$C = E - N + 2P,$$

where

E = the number of edges of the graph.

N = the number of nodes of the graph.

P = the number of connected components.

n alternative formulation is to use a graph in which each exit point is connected back to the entry point. In this case, the graph is strongly connected, and the cyclomatic complexity of the program is equal to the cyclomatic number of its graph (also known as the first Betti number), which is defined as

$$C = E - N + P.$$

This may be seen as calculating the number of linearly independent cycles that exist in the graph, i.e. those cycles that do not contain other cycles within themselves. Note that because each exit point loops back to the entry point, there is at least one such cycle for each exit point.

CF – Cost Factor

r => n * (1/t) * rnd

n -> Length of the Input

t -> Execution Time

rnd -> Random Fuzzy Random

c = 1/(r * (p + e)) * 100

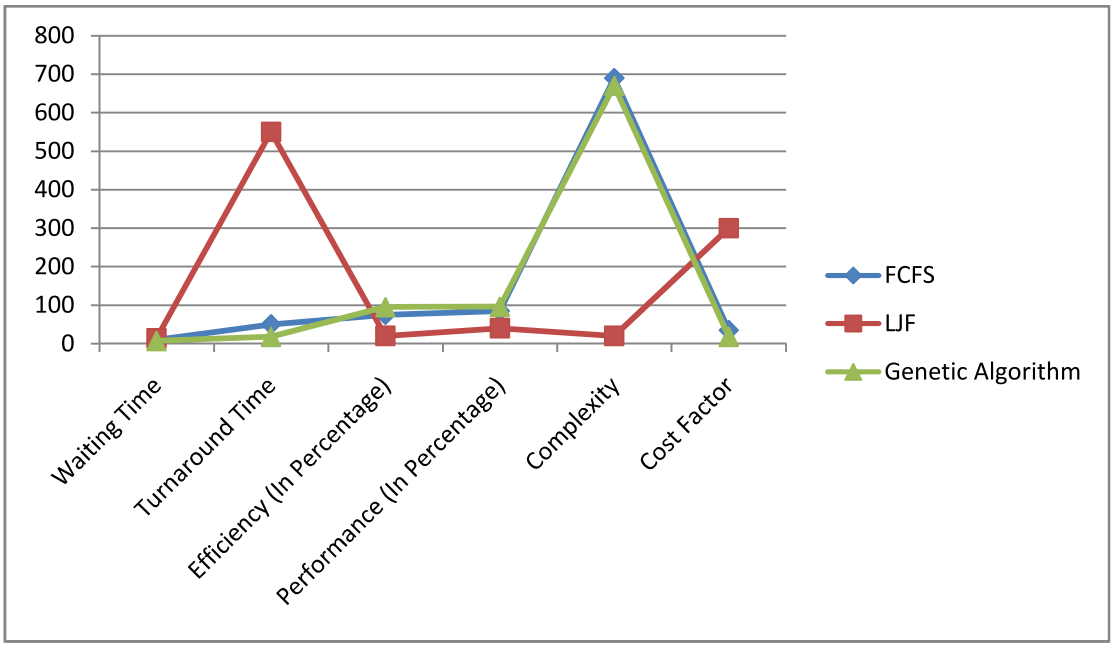


Fig. 8 – Cumulative Analysis of the Paramters

CONCLUSION AND FUTURE WORK

Because of the advances in the technology, the issues and further scope in science and engineering are becoming more complicated than ever before. To solve these complicated problems, grid computing becomes a popular tool. A grid environment collects, integrates, and uses heterogeneous or homogeneous resources scattered around the globe by a high-speed network. A grid environment can be classified into tw types: computing grids and data grids. This research work focuses on job scheduling algorithms and their performance on multiple parameters in the grid environment. In computing grid, job scheduling is a very important task. A good scheduling algorithm can assign jobs to resources efficiently and can balance the system load.

For future scope of the work, following techniques can be used in hybrid approach to better and efficient results –

- Particle Swarm Optimization
- HoneyBee Algorithm
- Simulated Annealing
- Genetic Algorithmic Approaches

REFERENCES

[1] Joseph, J., Ernest, M., & Fellenstein, C. (2004). Evolution of grid computing architecture and grid adoption models. *IBM Systems Journal*, *43*(4), 624-645.

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., ... & Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.

[3] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. ACM SIGCOMM Computer Communication Review, 38(4), 63-74.

[4] Robinson, J., Sinton, S., & Rahmat-Samii, Y. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In Antennas and Propagation Society International Symposium, 2002. IEEE (Vol. 1, pp. 314-317). IEEE.

[5] Kune, R., Konugurthi, P. K., Agarwal, A., Chillarige, R. R., & Buyya, R. (2014, December). Genetic Algorithm based Data-aware Group Scheduling for Big Data Clouds. In Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing (pp. 96-104). IEEE Computer Society.

[6] Sailer, R., Valdez, E., Jaeger, T., Perez, R., Van Doorn, L., Griffin, J. L., ... & Berger, G. S. (2005). sHype: Secure hypervisor approach to trusted virtualized systems. *Techn. Rep. RC23511*.

[7] Mathew, T., Sekaran, K. C., & Jose, J. (2014, September). Study and analysis of various task scheduling algorithms in the cloud computing environment. In Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on (pp. 658-664). IEEE.

[8] Singh, S., & Kalra, M. (2014, November). Scheduling of Independent Tasks in Cloud Computing Using Modified Genetic Algorithm. In Computational Intelligence and Communication Networks (CICN), 2014 International Conference on (pp. 565-569). IEEE.

[9] Verma, A., & Kaushal, S. (2014). Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud. International Journal of Grid and Utility Computing, 5(2), 96-106.

[10] Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014, January). Hybrid job scheduling algorithm for cloud computing environment. In Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014 (pp. 43-52). Springer International Publishing.

[11] Rodriguez, M. A., & Buyya, R. (2014). Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds.Cloud Computing, IEEE Transactions on, 2(2), 222-235.

[12] Singh, L., & Singh, S. (2014). A Genetic Algorithm for Scheduling Workflow Applications in Unreliable Cloud Environment. In Recent Trends in Computer Networks and Distributed Systems Security (pp. 139-150). Springer Berlin Heidelberg.

[13] Shojafar, M., Javanmardi, S., Abolfazli, S., & Cordeschi, N. (2015). FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, 18(2), 829-844.

[14] Gu, J., Hu, J., Zhao, T., & Sun, G. (2012). A new resource scheduling strategy based on genetic algorithm in cloud computing environment. Journal of Computers, 7(1), 42-52.

[15] Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on (pp. 400-407). IEEE.

[16] Kune, R., Konugurthi, P. K., Agarwal, A., Chillarige, R. R., & Buyya, R. (2014, December). Genetic Algorithm based Data-aware Group Scheduling for Big Data Clouds. In Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing (pp. 96-104). IEEE Computer Society.

[17] Verma, A., & Kaushal, S. (2014). Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud. International Journal of Grid and Utility Computing, 5(2), 96-106.

[18] Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014, January). Hybrid job scheduling algorithm for cloud computing environment. In Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014 (pp. 43-52). Springer International Publishing.

[19] Ye, H. (2015). Research on Emergency Resource Scheduling in Smart City based on HPSO Algorithm. city, 5, 6.

[20] Pawar, A., Scholar, M. T., & Kapgate, P. D. (2014). A Review on Virtual Machine Scheduling in Cloud Computing. vol, 3, 928-933.

[21] Shojafar, M., Javanmardi, S., Abolfazli, S., & Cordeschi, N. (2015). FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, 18(2), 829-844.

[22] Zhang, F., Cao, J., Li, K., Khan, S. U., & Hwang, K. (2014). Multi-objective scheduling of many tasks in cloud platforms. Future Generation Computer Systems, 37, 309-320.

[23] Quang-Hung, N., Tan, L. T., Phat, C. T., & Thoai, N. (2014). A GPU-Based Enhanced Genetic Algorithm for Power-Aware Task Scheduling Problem in HPC Cloud. In Information and Communication Technology (pp. 159-169). Springer Berlin Heidelberg.

[24] Tsai, C. W., & Rodrigues, J. J. (2014). Metaheuristic scheduling for cloud: A survey. Systems Journal, IEEE, 8(1), 279-291.

[25] Lin, W., Liang, C., Wang, J. Z., & Buyya, R. (2014). Bandwidth-aware divisible task scheduling for cloud computing. Software: Practice and Experience,44(2), 163-174.

[26] Kumar, M., & Doegar, A. (2014). Reliable and Efficient Task Scheduling based on Genetic Algorithm in Cloud Computing Environment.

[27] Frîncu, M. E. (2014). Scheduling highly available applications on cloud environments. Future Generation Computer Systems, 32, 138-153.

[28] Yang, T., & Gerasoulis, A. (2014, June). Author retrospective for PYRROS: static task scheduling and code generation for message passing multiprocessors. In 25th Anniversary International Conference on Supercomputing Anniversary Volume (pp. 18-20). ACM.

[29] Leena, V. A., & Rajasree, M. S. (2016). Genetic Algorithm Based Bi-Objective Task Scheduling in Hybrid Cloud Platform. International Journal of Computer Theory and Engineering, 8(1),

[30] Xu, Y., Li, K., Hu, J., & Li, K. (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues.Information Sciences, 270, 255-287.

[31] Lin, J., Zhong, Y., Lin, X., Lin, H., & Zeng, Q. (2014). Hybrid Ant Colony Algorithm Clonal Selection in the Application of the Cloud's Resource Scheduling. arXiv preprint arXiv:1411.2528.

[32] Zdenek Konfrst, " Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives", 18th International Parallel and Distributed Processing, 2004.

[33] Marin Golub, Leo Budin, "An Asynchronous Model of Global Parallel Genetic Algorithms" Unska 3, HR-10000 Zagreb, Croatia

International Journal of Computing and Corporate Research

ISSN (Online) : 2249-054X

Volume 5 Issue 6 November 2015

International Manuscript ID : 2249054XV5I6112015-02