ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 11, November 2016

Performance Analysis and Comparative Study of Process Migration Using Genetic Algorithm

Sandhya S, Revathi S, Dr. N.K Cauvery

Abstract— **Process migration is the act of transmitting the state of the process from overloaded node to a pre-determined destination node for completion of its execution. Process migration implicitly achieves load balancing by the distributing the processes. Genetic Algorithms (GA) help in deriving optimized solutions from a group of available predetermined solutions. Therefore, a genetic algorithms-based approach for process migration approach is analyzed and studied. This paper presents an analysis on process migration with genetic algorithm and compares the traditional migration approaches with the various metric used for evaluation.**

Index Terms—**Process Migration, Genetic Algorithm, Load Balancing**

I. Introduction

a. Process Migration: Process migration is an act of transferring an active process from one machine to another machine, thereby increasing the performance of distributed systems. The main aim of process migration is load distribution among the available machines, by which it solves the problem of load balancing. There are two types of process migration algorithms: preemptive process migration [9] and non-preemptive process migration. Preemptive migration [1] involves transfer of an executing process by suspending the process on the source node and migrating it to the destination node where it resumes its execution. Capturing the process's state is cumbersome while the execution is in progress, hence this approach to migration is an expensive operation. On the other hand, Non-preemptive migration involves the transfer of processes that have not started their execution. This approach of migration hence does not require the transfer of the process's state. This is mostly scheduling or allocating of processes to other nodes in the system.

There are a variety of preemptive migration algorithms namely: (1) Total Copy migration[1]- This algorithm would first suspend the running process on the source host. The source then transfers the complete state information to the destination node where the process is resumed. This approach is used more often for process migration. (2) Precopy[1] –the execution of the process proceeds in parallel with the transfer of the address space from the source host to the destination, the process on source host is suspended only after the complete transfer of the address space. (3) Lazy copy[2]-this mechanism migrates only the minimal necessary information required for resuming of process on the destination host, rest of the information would be transferred based on demand.(4) Flushing[1] –in this approach the process on source host is suspended, and all dirty pages are flushed on to the file server(third party between the source and destination which keeps the backup of process state information), the process later resumes its execution at the destination by retrieving the pages from the file server. (5) Postcopy[1]- this algorithm suspends the process execution on the source host and transfers the subset of process state information to the destination host. The process execution resumes at the destination host, after which the source host transmits the remaining state information in parallel to the process execution at the destination node.

b. Genetic Algorithm: Genetic Algorithm GA is an evolutionary based optimization algorithm with a global search potential proposed by Holland in 1975. GA is one of

Manuscript received Oct , 2016.

Sandhya S, *Department of CSE, RVCE, Bengaluru, India.*

Revathi S, *Department. of CSE, RVCE, Bengaluru, India.*

Dr. N.K Cauvery, *Professor & Head of Department of ISE, R.V.C.E, Bengaluru, India*,

the most successful class of algorithms under evolutionary algorithms, which is based on the evolutionary ideas of natural selection. These approaches follow the principles of Charles Darwin Theory,- the survival of the fittest. However, because of its outstanding performance in optimizing the solution, GA has been regarded as a function optimizer. The genetic algorithm optimizes the problems by mimicking the principles of biological evolution, repeatedly modifying a population at individual points. It is very efficient and stable in searching for global optimum solutions. There are three basic operation of genetic algorithm namely:

(i)Selection- this operation selects the fittest individuals from a given population, called parents that contribute to the creation of new population for the next generation.

(ii) Crossover- it combines subparts of two parents to form new children for the next generation.

(iii) Mutation- this operation makes random changes to individual parent chromosomes to form new child chromosomes.

Genetic algorithm associated with process migration: Genetic algorithm [7] is one of the widely used algorithms for dynamic load balancing using process migration. GA involves decreasing the unnecessary requests to all of the available nodes and increases the acceptance rate. Genetic algorithm determines the destination node to which the process will be migrated, thereby reduces the difficulties of identifying the destination node. Thus the process migration with GA [11]provides results which are more optimized and thus improves performance.

II. COMPARATIVE ANALYSIS

All traditional process migration algorithms have overheads to consolidate data needed for the transfer which is either prior or post migration called preparation and post-processing time that needs to be kept minimum. Another additional factor that determines the efficiency of such algorithms is the residual dependency on the executing system. Table 2.1 shows the comparison of these factors for the above described process migration algorithms.

Table 2.1 Comparison for Migration Overhead

Migration Algorithms	**Preparation Time**	**Post-processing Time**	**Residual Dependency**
Total Copy migration	NO	NO	NO
Precopy	YES	NO	NO
Lazy copy	NO	YES	YES
Flushing	NO	YES	YES (on third party entity)
Postcopy	NO	YES	NO

Another significant factor for evaluating the performance of process migration algorithm is the down time. Down time is the time period from the suspension of the process on the source node along with any communication interruption to the process. Along with this the total migration time is also an important metric considered for evaluation.

Total copy algorithm exhibits total migration time same as/equal to the down time. The down time is the largest for this algorithm as compared to any other algorithm whereas the total migration time is the lowest since there is no residual dependency (refer table 2.1) and data need not be transmitted later.

As indicated in table 2.1, *Precopy Algorithm* has to prepare for transmission before the process migration, reducing the amount of data transmitted during the downtime. Hence the downtime is noticeably shorter in precopy as compared to total copy but slightly higher than the post copy algorithm. The total migration time cannot be drawn any inference since this depends on the amount of data that gets dirtied and the number of pages that needs to be transmitted. If the dirty pages are more then the total migration time of precopy will the largest of all the migration algorithms.

The *lazy copy algorithm* has residual dependency on source and some volume of post processing and hence the downtime is lower than precopy. Since there is post processing done after the process resumes on the destination node, the total migration time cannot be drawn conclusion as this depends on the dirtied page rate and these being referred during the processing on the destination node.

Flushing Algorithm is implemented on Sprite and it is seen that there are no other migration algorithms implementation on sprite. Hence a comparison for this algorithm cannot be made.

The *post copy approach* migrates the minimum process's state hence the downtime is measurably short but needs to work on both the transferred process and the algorithmic demands from source host due to residual dependency increasing the total migration time which can compared with the total copy approach.

The table 2.2 [1] gives the results of migrating light-weight process with either high-speed or low-speed network connection. It can be inferred for table 2.2, the post copy algorithm is best choice to migrate heavy-weight process with low network speed and high network speed, total copy and pre copy is not suitable to migrate the heavy-weight process.

Table 2.2: Comparative analysis of process migration algorithms

Algorithms	**Downtime**	**Total Migration Time**	**Performance / Throughput**
Total Copy[1]	High Downtime	Total migration time is low	Good
Precopy [1]	Minimal downtime but increase in number of cores results in infeasible	Increase in total migration time as increase in preparation time	Acceptable but comparatively lesser then hybrid
	downtime		
Lazy copy[2]	Minimum downtime	Increase in total migration time	Accessible performance
Flushing [3][16]	Reduced downtime	Moderate total migration time	High performance
Copy-on-reference [3]	Shortens downtime	Total migration time is low	Performance degrades
Post copy[1]	Shortens downtime	Increase in total migration time	Degrades the response time which affects the performance
Hybrid[5]	Shortens the system downtime	Increase in total migration time	Good performance
Generic Process Migration[6]	Reduce in downtime	Decrease in migration time	Good performance

III. PROCESS MIGRATION USING GENETIC ALGORITHM:

Using process migration load is balanced between two communicating hosts (source & destination).These hosts are termed as overloaded and underloaded nodes, where the overloaded node initiates the process of migration for distribute its load to the available destination (underloaded) nodes. If any node in the system becomes overloaded it sends the migration request to all available nodes in the distributed environment. This increases traffic and result in overhead, resulting in reduced performance. Hence Genetic algorithm based load balancing scheme was proposed [15] for reducing the number of request messages. The learning procedure formulates an appropriate fitness function whose

implementation plays a major role in selecting the subset nodes as destination / underloaded nodes that would receive the request messages for migration. The reduced mean response time is plotted in the graph in figure 2.1. It is evident from the graph that GA is an effective approach for reducing the traffic in the system which in turn reduces the response time.

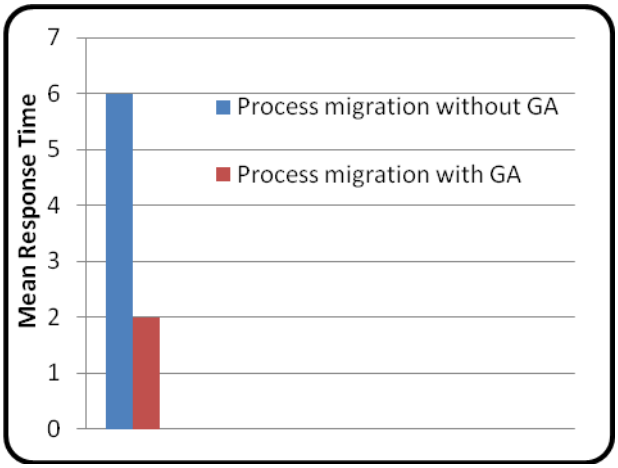

Figure 2.1: Behavior of Mean Response Time

The migration of process distributes the load, further improving the system performance[12]. GA reduces the communication traffic by the bringing down the number of migration request thus optimizing the performance. Minimizing the response time decreases the overall execution time resulting in the increase of system throughput. This is illustrated in the graph in figure 2.2 where the downtime remains constant [24] for migration even with the use of genetic algorithm.

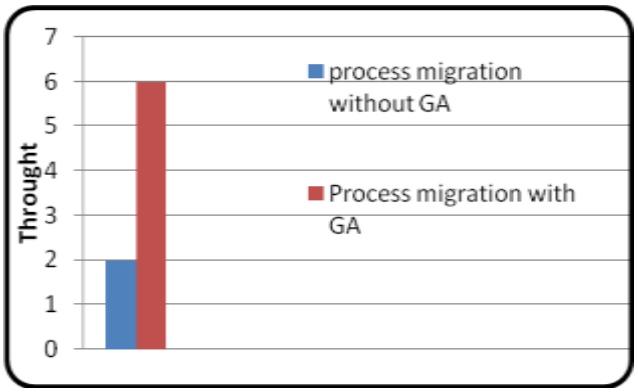

Figure 2.2: Throughput behavior with GA

IV. CONCLUSION:

Process migration with genetic algorithm for load balancing effectively reduces the imbalance of workload across the multiple processors. Genetic algorithm helps in finding the destination processor for the sender to migrate the process. The migrated process resumes execution at the chosen destination yields better throughput as compared to the migration without GA. Thus genetic algorithm works efficiently in achieving the minimum completion time, minimum mean response time, maximum resource utilization and maximum throughput.

V. REFERENCES

[1] NOACK, M. "Comparative evaluation of process migration algorithms". Master's thesis, Dresden University of Technology - Operating Systems Group, 2003.

[2] Michael Richmond and Michael Hitchens." A new process migration algorithm". In proceeding of ACM SIGOPS Operating Systems Newsletter, Volume 31 Issue 1, Jan. 1997.

[3] Dejan S. Milojicic, Fred Douglis, Yves Paindaveine, Richard Wheeler, " Process Migration", ACM Computing Surveys, Vol. 32, No. 3, September 2000.

[4] M. Kozuch and M. Satyanarayanan, "Internet suspend/resume". In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, 2002.

[5] Syed Asif Raza Shah, Amol Hindurao Jaikar, Seo- Young Noh, " A performance analysis of Precopy, Postcopy and Hybrid live VM Migration Algorithms in scientific Cloud Computing Environment", IEEE, 2015.

[6] Amirreza Zarrabi, " A Generic Process Migration Algorithm". International Journal of Distributed and Parallel Systems(IJDPS) Vol.3, N0.5, September 2012.

[7] Dam, Scintami, et al. "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing." Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on. IEEE, 2015.

[8] Wenzheng Li, Hongyan Shi. "Dynamic Load Balancing Algorithm Based on FCFS", Fourth International Conference on Innovative Computing, Information and Control, IEEE, 2009.

[9] Sandhya . S, Cauvery N K, *"An Improved and Optimized approach for Pre-emptive Migration of Video Process"*,proceedings of the International Conference on Contemporary Computing and Informatics IC3I,SJCE, Mysore, Conference proceeding indexed in IEEE Digital Library, November 2014, pp 894 - 898.

[10] Ramon Lawrence, "A survey of process migration mechanism". May 29 1998

[11] Ranjitha K N, Sandhya S, N K Cauvery, *"A Survey on: Pre-Emptive Migration of a video process using Genetic Algorithm on Virtual machine",* International Journal Of Engineering And Computer Science, Volume 3 Issue 5, may 2014, ISSN:2319-7242, pp 5897-5900.

[12]M. Munetomo, Y. Takai and Y. sato, " A Stochastic genetic Algorithm for Dynamic Load Balancing in Distributed Systems". IEEE, 1995,pp. 3795-37999.

[13] Albert Y. Zomaya and Yee- HweiTeh, " Observation on Using Genetic Algorithms for Dynamic Load Balancing", IEEE, Vol. 12. No. 9. September 2001.

[14] Kulvinder Singh and ChetnaKukreja, "A Sender-Initiated Load Balancing Approach with Genetic Algorithm", IJSWS 2012.

[15] Masaharu Munetomo, Yoshiaki Takai and Yoshiharu Sato, "A Genetic Approach to Dynamic Load Balancing in a Distributed Computing System", IEEE , 1995, P. 418-421.