International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 11 (2018) pp. 9636-9645
© Research India Publications. http://www.ripublication.com

Fault-Aware Advance Reservation Scheduling in Heterogeneous Computing Systems

Harmanpreet Kaur and Amit Chhabra

Department of Computer, Engineering and Technology, Guru Nanak Dev University, India.
Department of Computer, Engineering and Technology, Guru Nanak Dev University, India.

Abstract

Resource availability is an important issue nowadays due to the possibility of occurrences of faults in heterogeneous computing systems such as grid and cloud systems. Traditional advance reservation scheduling techniques do not consider resource availability while making resource reservations. There are many high priority jobs that need to be executed as quickly as possible but others jobs lead to the starvation of such jobs. To overcome this problem advance reservation scheduling is considered to allocate the resources to such jobs when needed which consider deadline constraint. Such jobs could not be risked to be executed on the machines that could fail very easily. The fault awareness is very important to consider which has not been done so far with advance reservation. Reliability is handled in the proposed approach by considering mean time between failures metric. Deadline sensitive jobs are compared against finish time of jobs which is calculated in advance and AHP is applied to determine their completion time. AHP matrix indicates if jobs can satisfy deadline or not. To resolve the faults, task execution state at current machine is migrated to other machine using checkpointing approach which leads to reduction of the execution time. The result of the proposed fault-aware scheduling policy showed improvement in terms Makespan and Flowtime by 10% and 14% respectively.

Keywords: AHP, Deadline, Makespan, Flowtime, Reliability

INTRODUCTION

Resource availability and job execution becomes need of the hour in case of resource constraint environment. **(Khoshkholghi et al. 2017)**Cloud computing with homogeneous environment may not be sufficient to execute jobs in time critical and deadline sensitive environment. **(Patel & Jethva 2013)**To tackle the issue, heterogeneous cloud consisting of multiple providers must be considered for evaluation. Resources in such a situation are collaborated within the pool known as resource pool. (Pei et al. 2015)As the jobs arrive within the system, their machine requirements are matched against the pool. In case pool has the sufficient resources then jobs are allotted to the resources and resources are decreased from the pool. This process continues until all the jobs are allotted. The problem of deadline sensitive jobs become vigorous since time criticality is considered. **(Li et al. 2012; Li et al. 2014)**In order to satisfy time or deadlines, advance reservation becomes need of the hour. Advance reservation ensures that job enters into the system only if its requirements will be satisfied otherwise jobs are not accepted for scheduling. Overall throughput is considerably decreased using advance reservation.

Mean time between failure must be incorporated within the job allocation in order to ensure execution of maximum number of jobs without fault or failure.**(Schroeder & Gibson 2007; Guermouche et al. 2011)** Fault tolerance alludes to right and nonstop operation even within the sight of non-functional resources. It is the craftsmanship and art of building computing framework that keep on operating attractively within the sight of faults. A fault tolerant framework must have the capacity to endure at least one fault composes including - transient, irregular or perpetual component faults, programming mistakes, administrator mistakes, or remotely actuated surprises or physical damage. **(Bautista Gomez et al. 2010; Salehi et al. 2016)**In constant cloud applications, preparing on computing hubs is done remotely which has a high likelihood of event of blunders. These occasions increment the requirement for fault tolerance methods to accomplish dependability for the constant computing on cloud framework.

With the increase in cloud computing services, there is a possibility that faults may occur which adversely affects the cloud performance. These faults can be of different kinds including:

- transient, intermittent or changeless equipment faults;
- software bugs and plan mistakes;
- Operator blunders;
- Externally actuated faults and blunders.

The proposed system considers federated cloud environment to determine resource availability, MTBF for faults and failures during job execution and allotting jobs to resource with maximum MTBF. Checkpointing approach is used to ensure backup of progress made at current machine. AHP matrix is used to determine job completion. Early finish time is calculated to preempt the resources from the jobs which are already finished. This ensures availability of resources as and when required by the jobs. Rest of the paper is organised as under: section 2 presents literature survey of various job scheduling mechanisms along faults and failure tackling mechanism. Section 3 describes the methodology, section 4 deals with evaluation of performance comparison, section 5 exhibits the conclusion and future scope and last section gives references.

RELATED WORK

This section includes analysis of techniques used to ensure compatible execution of jobs over the resources. Fault tolerant strategy checkpointing is discussed as it becomes part of the proposed system as well. This section includes the literature survey of different aspects of the proposed system is discussed one by one.

(Xhafa et al. 2011) proposed hybridization of genetic and tabu search mechanism for job allocation and execution**.(Rodger 2016; Elghirani et al. 2008)** To execute the jobs genetic approach is followed and to locate the resource tabu search is used. Fitness function is defined in terms of cost. The fitness function thus has to be minimised and is achieved through said literature. **(Switalski & Seredynski 2014)** proposed a generalized external optimization (GEO) which is enhancement of genetic approach. The discussed approach consists of two phases. In the first phase, optimal virtual machine out of the available machines is selected. In the second phase, batches are scheduled to execute on selected virtual machine. **(Kliazovich et al. 2013)** proposed an energy aware job scheduling within the data centers. Energy efficiency and network awareness is being presented in this literature for achieving optimization in terms of Makespan and Flowtime.

Advance Reservation policy proposed by **(Bhartee et al. 2015)** suggests that without the application of advance reservation, resources within the community cloud may not be available as and when desired causing starvation. In order to resolve the problem, Advance reservation is implemented which implies that every VM requiring resource must request it before execution of task. In case resource is not available then job execution will be suspended. **(Sindhu & Mukherjee n.d.)** proposed advance reservation scheme for task scheduling within cloud. Advance reservation is implemented in cloud system suggested using lease connection. The cost is the biggest problem associated with lease connection.

Advance reservation scheme using multi heuristic criteria as proposed by (Bama 2009). The entire scheduling mechanism is divided into phases. The first phase is a request phase. In the request phase cloud users request the resources from service provider. Second phase takes the requirement request from the first phase and judge the credibility and length of the task. And final phase allocate the resources. The resource allocation is on the basis of complexity and priority of the task. The delay in allocation is considerably reduced by the application of advance reservation.

Advance reservation and architecture of grid computing is discussed by (Prajapati & Shah 2014). Ganglia monitoring system for grid computing system tackle the issues of resource provisioning in the proposed literature.

The algorithm proposed by (Computing 2014) suggested the advance reservation mechanism for workflow scheduling. ECA rule based workflow scheduling suggested includes advance reservation of resources before executing the task. Once task is allotted to VMs, execution no longer can be delayed due to shortage of resources. Hence execution of task became faster due to least waiting period.

Advance reservation in grid computing using GridSim is proposed by (Sulistio et al. n.d.). The simulation environment consists of five distinct resource types with multiple instances. The resource availability ensures avoidance of deadlock situation. First come first system is supported using GridSim API.

Scheduling with advance reservation in grid is proposed by (Min & Maheswaran n.d.). Priority assignment with grid computing is used in the studied literature. Priority assignment is user specific. User specific input driven environment makes overall processing slower.

Advance reservation with flexible start time is proposed by (Nirmala et al. 2016). Experiment in OpenNibula environment was conducted and jobs with deadline constraint environment found to be stringent and less effective in such environment. Start time flexibility provided with the compensation of performance in terms of finish time. Request acceptance rate however is improved greatly through suggested literature.

(Manikandaprabhu & Sivasenthil 2013) Proposed adaptive reservation scheme to minimize cost of resource provisioning in cloud computing. Reservation technique using over provisioning (RTUOP) is used to improve the performance in terms of cost in the studied literature.

The algorithm proposed by (Chaisiri & Member 2012) efficiently provision resource among regular cloud users. The algorithm optimal cloud resource provisioning (OCRP) includes application of advance reservation along with partitioning the jobs into parts known as modularization and deterministic equivalent formulation strategy for minimizing the cost associated with the resource allocation.

Advance reservation scheme for scheduling complex jobs within community cloud is proposed by **(Bhartee et al. 2015)** Work is being done towards availability of resources using advance reservation with lack of reliability standards. Deadline constraint jobs require reliable resources which are ensured by considering reliability metrics. None of the existing work focused on reliability aware advanced reservation scheduling within federation of cloud. According to **(Armbrust et al. 2010)** the elasticity of resource pool is unprecedented mechanism provided to help out IT industry without paying a heavy amount to service providers. This allows reduction is cost during maintenance and deployment. The resources are provided using the mechanism of reliability discussed by **(Sharma et al. 2016).** Reliability of cloud enhances its popularity. In other words through reliability, it is ensured that more and more users utilize the services of the cloud. Reliability degradation results in loss of consumer that directly have business profitability issue. Quality of Service (QoS) must be ensured to maintain or enhance the popularity of cloud that is accomplished using Reliability Metrics. **(Zhou et al. 2014)** proposed reliability enhancement of cloud by reducing the storage resource utilization. The identical services provided by the similar VMs are identified and then check pointed, thus separate checkpoint image is discarded causing the reduction in network and storage requirements. In case of failure, recovery from check pointed image is made that establish reliability.

(B. Egger, Y. Cho, C. Joe, E.Park 2016; El-sayed & Schroeder 2014) The full checkpointing mechanisms retain system's complete running states frequently on a storage platform, but in incremental checkpointing the complete running states of a system are included in first checkpoint and succeeding checkpoints only retain pages that are updated since the last checkpoint. **(Zhou et al. 2017)**Checkpointing data is saved on local disk storage in local checkpointing, therefore transient failure can be identified from local checkpointing whereas checkpointing data is stored on global storage which can be retrieved in new storage node from global checkpointing in case of permanent failure.

(Palaniswamy n.d.; B. Egger, Y. Cho, C. Joe, E.Park 2016) Coordinated checkpointing mechanisms generally depend upon a collaboration of operating system or user level runtime library support for checkpointing whereas uncoordinated checkpointing mechanisms depend upon logging messages. **(Salehi et al. 2016)**Check-pointing can be Disk & Diskless, which is done with the help of MPI. Disk based stores data on global disk storage and is used for node or network failure while diskless stores data on local storage and is used for process or application failure.

PERFORMANCE METRICS-

(Kumar et al. 2014; Singh et al. 2012) The current fault tolerance system in cloud computing consider following parameters: throughput, response - time, adaptability, execution, accessibility, usability, reliability, security and related over - head.

- **Throughput:** It characterizes the quantity of assignments whose execution has been finished. Throughput of a framework ought to be high.
- **Response Time:** Time taken by a calculation to react and its esteem ought to be made limited.
- **Scalability:** Number of hubs in a framework does not influence the fault tolerance limit of the calculation.
- **Performance:** This parameter checks the viability of the framework. Execution of the framework must be upgraded at a sensible cost e.g. by permitting worthy defers the reaction time can be lessened.
- **Availability and MTBF:** Availability and mean time between failures ensures system is available as and when desired. Availability of a framework is specifically master proportional to its dependability. The likelihood a thing is working at a given occurrence of time under characterized conditions.
- **Usability:** The degree to which an item can be utilized by a client to accomplish objectives with adequacy, effectiveness, and fulfilment.
- **Reliability and MTTR:** This viewpoint means to give right or worthy outcome inside a period limited condition. MTTR specified the time required to recover the system to its original state.
- **Overhead Associated:** It is the overhead related while executing a calculation. Overheads can be forced as a result of assignment developments, inter process or bury - processor correspondence. For the effectiveness of fault tolerance strategy the overheads ought to be limited.
- **Cost Adequacy:** Here the cost is just characterized as a monitorial cost.

The discussed literature highlight the terms which are considered for improvement in our work. The proposed system is discussed in the next section along with the experimental setup.

PROPOSED SYSTEM

a**. SYSTEM MODEL**- The proposed system consists of clusters which are heterogeneous in nature. These clusters contain processors which are to be assigned to the jobs. We have changed the no: of machines in different clusters according to our need to check the performance of our system. The resources before allocation is passed through advance reservation scheme. In other words, a special variable is associated with the machines indicating whether they are already reserved or not. In case resources are already reserved, then jobs must wait. Early finish time becomes critical in the scenario since it will be used to release the resources held by jobs which are finished. AHP matrix is maintained to determine the finish time of jobs. Checkpointing is established to enhance reliability and measurement metric which is used is mean time between failures.

b. APPLICATION MODEL- In proposed system i.e. fault-aware AHP (FA-AHP), we have considered jobs exhibiting the quadruple properties of itself. These four are- Arrival Time, Burst Time, Deadline Time and No: of Processors required to execute that job. The job type in our proposed system is of rigid nature. The arrival time of the job is the time when the job has entered into queue and has asked for the resources for its execution. Burst Time is the actual execution time needed for the completion of the job. Deadline Time is the time before which the execution must be finished. User has previously associated each job with its deadline time and according to that deadline time; we have done advance reservation so that the job is executed before the end of the deadline time of each job. No: of Processors is the count of processors required for the job execution.

c. ALGORITHM-The fault-aware AHP technique (FA-AHP) is listed as follows:

Input: Stream of jobs, Heterogeneous clusters of processors.

Output: Advance Reservation satisfying deadline and fault-tolerance.

```
1. Input job stream
        a. Select the job with FCFS strategy.
        b. Set K value
    2. Job Selection process(Advance Reservation)
       Perform Job ordering by checking job requirement
       against available machines and reject the jobs not
       lying within sequence.
    3. Selection of processor
        a. Check for Deadline and Max MTBF if found
           goto step b.
        b. Check for Security parameter
           (Max(Security(VM)) if found goto step c
        c. Processor selection on the basis of
           Machine_available.
           If
           $processor_i$_available>$processor_{i+1}$_Available_Cl
           uster
           Max_Available_Cluster=$processor_i$
        d. Processor selection on the basis of MTTF
           If $processor_i$_MTTF>$processor_{i+1}$_MTTZF
           Min_MTTF_Processor=$processor_i$

    4. Allocate job to Max_speed_processor and
       Min_MTTF_processor
    5. Check for deadline meet condition
        a. $Deadline_i$=$DeadLine_i$+Job_Arrival+K*Jobs_Giv
           en_Burst_Time
        b. Actual_Deadline=Current_Time+Burst_Time/Sp
           eed
        c. If Dead_line_$job_i$>=$Actual\_Deadline_i$
        d. Obtain result in terms of Makespan and
           Flowtime
           Else
           Go to step 2
    6. Check for availability processor
        a) This is performed to allocate next job in
           sequence to optimal processor
        b) $Availability_i = \frac{Burst_{time}}{Speed}$
        c) If processor_$available_i$==true
           Allocate the job and go to step 5.
        d) Perform step 3 to 6 until all the jobs finish
           execution
```

Results through this methodology are obtained in terms of Makespan and Flowtime. The performance analysis and results is given in the next section.

Experimental setup of fault-aware ahp-

Experiments corresponding to the proposed system consist of 5 clusters with 128, 96 and 64 machines. 'K' is used as a constant parameter whose value is in between 0.1 to 2. The jobs are fetched from a dataset. The configuration corresponding to the proposed system is given as under

Table 1. Experimental Setup

Parameters	Values
Jobs	200,100,50
Clusters	Five
Machines	128,96,64
K	0.1 to 2
Speeds	1,2,3,4,5
Flowtime	Initially 0
Makespan	Initially 0

PERFORMANCE ANALYSIS AND RESULTS

This section gives the performance analysis in terms of Makespan and Flowtime. Makespan is the maximum of all the times taken by jobs to complete the jobs and Flowtime is the total time taken to complete total job schedule. The performance analysis is conducted by varying the values of the parameters such as number of jobs, processor and constant parameter k. Result is obtained which is better as compared to existing literature without considering AHP, MTBF and deadline constraint. Obtained results are given as under-

a. As the load increases Makespan corresponding to existing and proposed techniques also increases. The plots demonstrate the same.

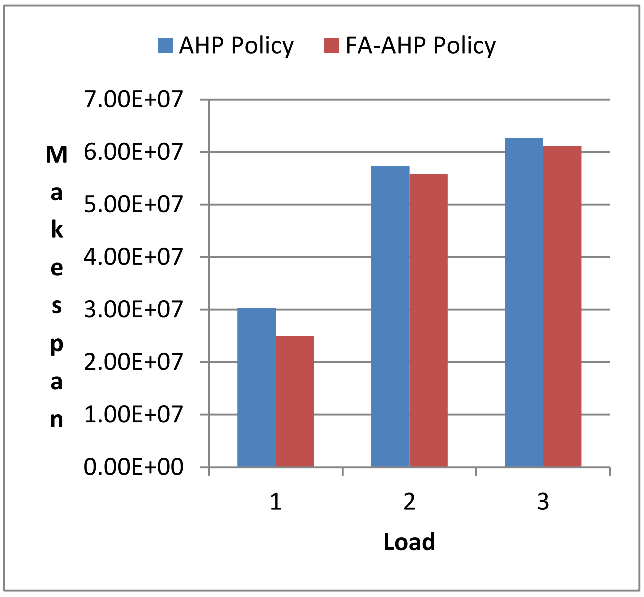


Figure. 1(a)

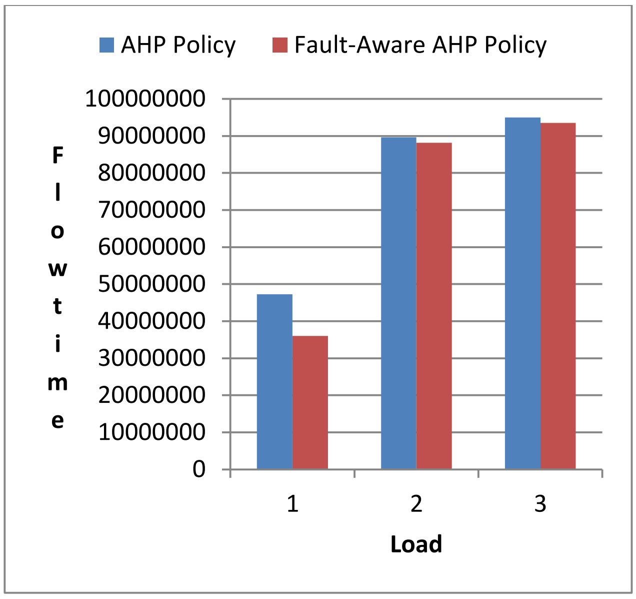


Figure. 1(b)

Figure 1(a) and 1(b): Plots corresponding to Makespan and flowtime when load is increased with No: of processors=64

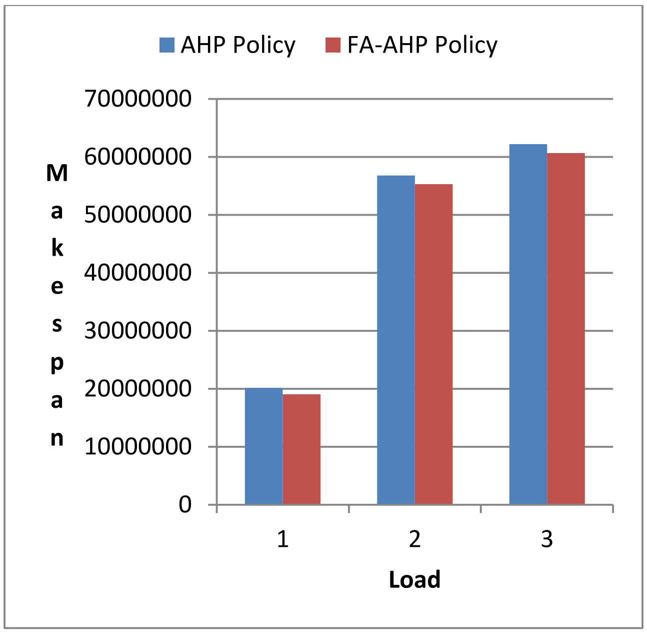


Figure. 2(a)

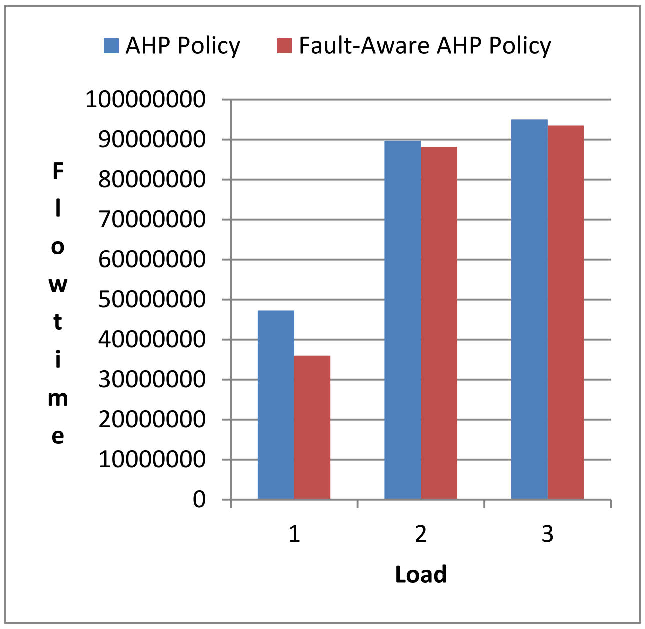


Figure. 2(b)

Figure 2(a) and 2(b): Plot comparison of Makespan and Flowtime with load with no: of processors=96

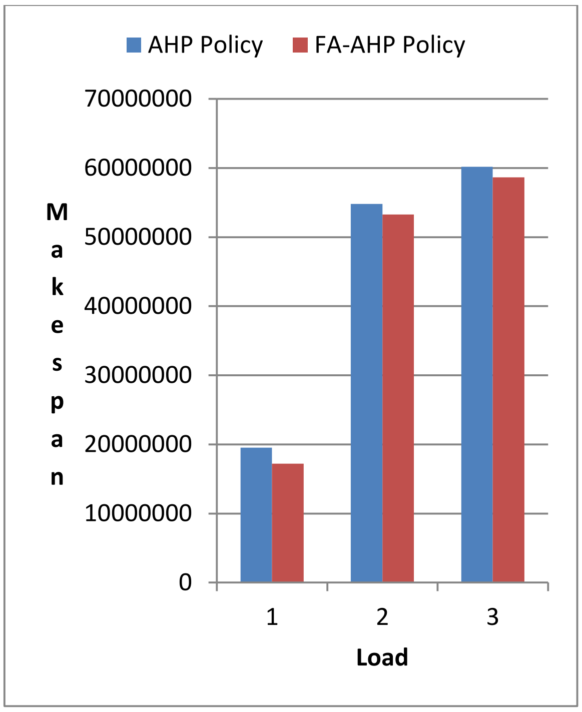


Figure. 3(a)

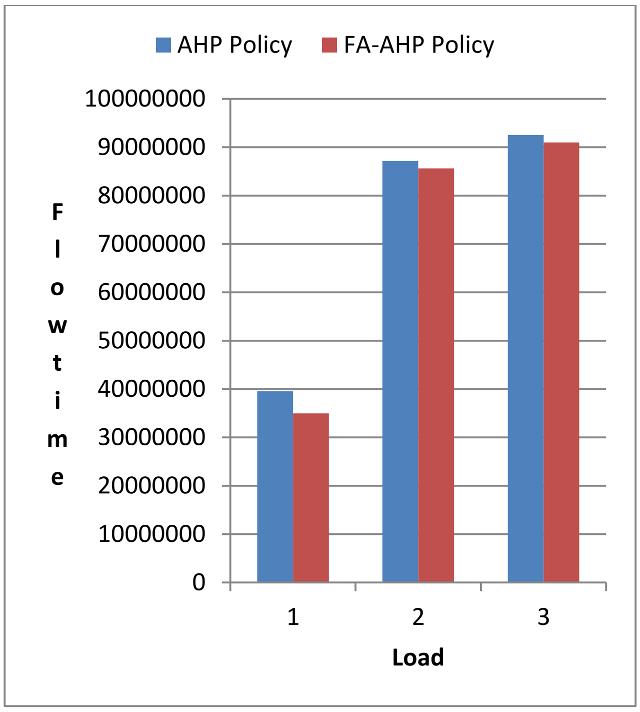


Figure. 3(b)

Figure 3(a) and 3(b): Plot comparison of Makespan and Flowtime with load with no: of processors=128

Different observations inferred from the graphs of Fig1(a), 1(b), 2(a), 2(b), 3(a) and 3(b) are as under

a. As the number of processors increases, makespan and flowtime decreases. This could be understood from the simple fact that as there will be more no: of processors, the job could be distributed to more no: of processors and the execution time decreases and therefore the turnaround time decreases and with the decrease of turnaround time, makespan and flowtime also decreases.

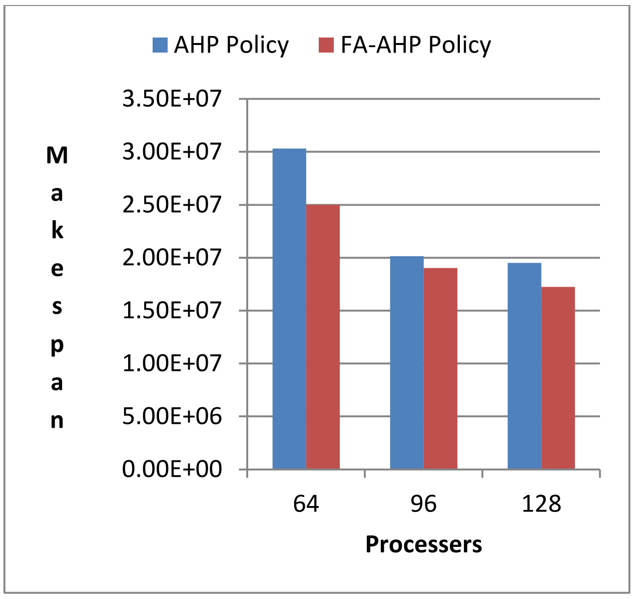
b. Also we can see that with increase in amount of load, makespan and flowtime increases. The amount of load increased is multiplied with the burst time of the job as so the execution time and turnaround time increases, and with this increase makespan and flowtime also increases.

c. Now we have seen that in each case, FA-AHP policy has lower makespan and flowtime as compared to AHP policy makespan and flowtime. As it is already discussed that in FA-AHP, checkpointing with reliability factor has been added but in AHP policy, no such technique is used so whenever there is an error, in AHP technique, the execution of the job was started from the start no matter what amount of execution was completed. This loophole was considered in the FA-AHP and checkpointing was included by which in case of any fault the execution resumes from the last checkpoint and not from starting and so the turnaround time decreases, and so does makespan and flowtime.

d. Also its important to mention here that even if checkpointing technique is added in the AHP technique, yet FA-AHP results will come out better than the AHP technique, because in FA-AHP, MTTR and MTBF metrics are considered and the execution of the job is assigned to the processor with max MTBF and min MTTR which is missing in AHP technique, so FA-AHP exhibits better performance from AHP (with and without checkpointing).

Variation is also observed as the number of processor varied. The variation when processor varied is given as under

The results indicates processor increase decreases the Makespan which is also elaborated through the plots as



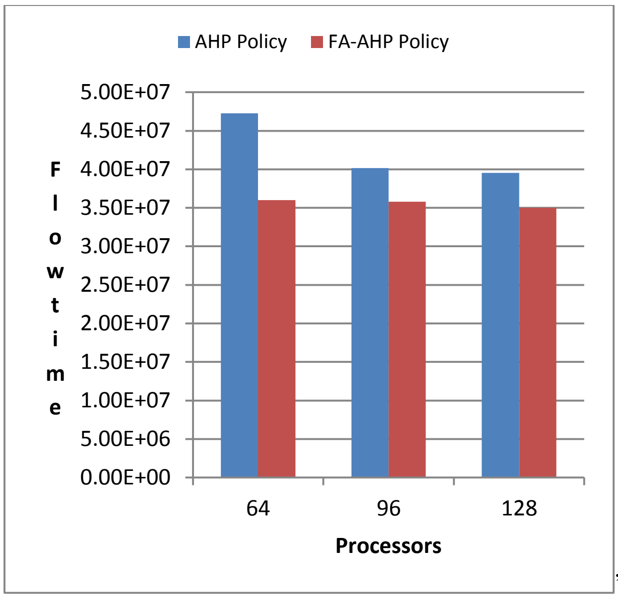


Figure. 4(a) **Figure. 4(b)**

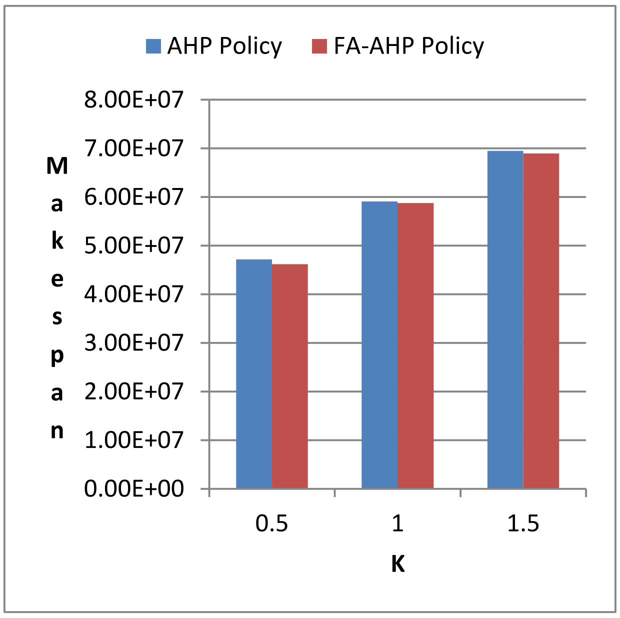
Figure 4(a) and 4(b): Comparison in terms of Makespan and Flowtime when processor increases

Observations of graphs 4(a) and 4(b) are –

i. With increase in number of processors makespan and flowtime decreases. This could be understood by the fact that as the no: of processors increases, there are more machines to execute the same job and so the turnaround time decreases, and which it the makespan and flowtime also decreases.

ii. We can notice that the makespan and flowtime of FA-AHP technique are less than AHP technique in each case. This is because the AHP technique has not considered the checkpointing as well as MTTB and MTTR metrics and so the execution time of it is larger than the FA-AHP technique in which these are considered and faults are handled smartly.

b. Constant metric K when varies between 0.1 to 2 also yield distinct results in terms Makespan and Flowtime. Result in terms of plots is given as under



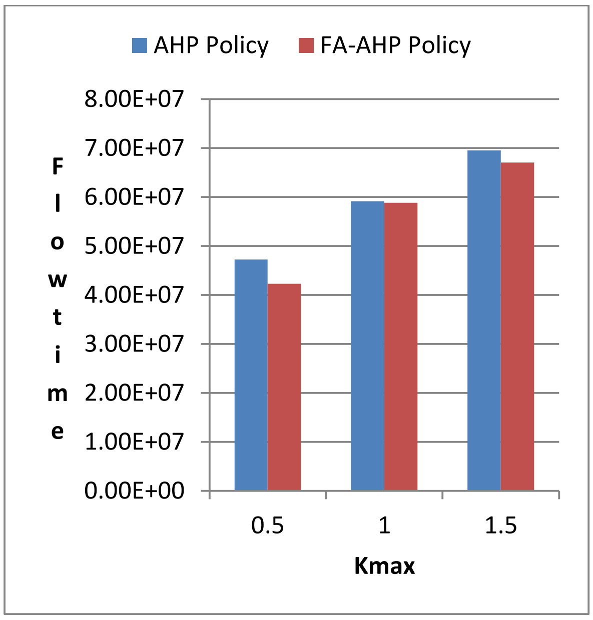


Figure. 5(a) **Figure.5(b)**

Figure 5(a) and 5(b): gives the comparison of Makespan and Flowtime obtained in terms of Kmax

Observations of graphs 5(a) and 5(b) are –

1. It is evident that with the increase in value of k, the makespan and flowtime increases. This is because in calculation of deadline, k factor is multiplied with job's burst time and so the execution time increases and so does the makespan and flowtime.
2. The performance of FA-AHP technique is better than the AHP technique this is due to the lack of checkpointing and MTTB, MTTR metrics in AHP technique.

Now let us consider the effect ok deadline which is determined by k factor on miss rate of the jobs. Graph showing effect is-

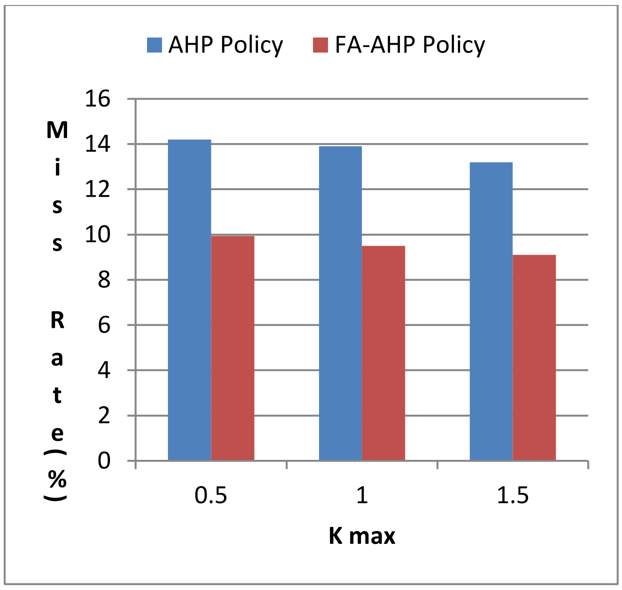


Figure 6. Gives the comparison of miss rate of AHP and FA-AHP.

Observations of graph in fig. 6 is-

1. With increase in k factor, the deadline of the job increases and so number of jobs executed is less and so the miss rate decreases.
2. Miss rate of AHP is more as compared to FA-AHP because it has not considered the fault tolerance and also when fault appears, it started the execution of the job from the starting thus increasing the execution time of job as so missing out other jobs. Whereas in FA-AHP, checkpointing is done, i.e. when fault come the execution starts from the last checkpoint instead of again from the start thus decreasing the execution time and so miss rate decreases.

Overall performance analysis suggests that as resources increases execution time decreases. The random parameter variation considering reliability also ensures least failures within the machines. Even if failures do occur they are tackled using checkpointing strategy. Performance is enhanced as there is 10% improvement in makespan results and 14% improvement in flowtime results.

CONCLUSION AND FUTURE SCOPE

Resource availability and fault awareness is the need of the hour in Heterogeneous computing systems. Traditional resource allocation techniques do not consider fault-tolerance while scheduling jobs. In this paper, we have presented the fault aware advance reservation policy (FA-AHP) which is an extension of AHP. The resources are dynamically released when the job finished. Earlier resources held by the job are not released as long it is in the system causing lack of resource availability. We have compared FA-AHP with AHP by varying the system load, resource heterogeneity, deadline and number of processors. In case of variation of load, the FA-AHP produced improvement in makespan in the range of 10-15% and in case of flowtime it is 7-20%, in case of variation of no of processors the improvement in makespan is in range of 15-20% and in case of flowtime its 17-22%. Lastly in case of deadline variation, improvement range of makespan is 3-5% and flowtime range is 7-12% Performance is enhanced by the margin of 10% and 14% respectively in average which is significant proving worth of the study. In future this approach can be collaborated with multi-heuristic approach like particle swarm optimization to enhance the results in terms of Makespan and Flowtime further.

REFERENCES

[1] B. Egger, Y. Cho, C. Joe, E.Park, J.L., 2016. Effcient Checkpointing of Live Virtual Machine Migration",. *IEEE Transactions on Computers*, pp.3041–3054.

[2] Bautista Gomez, L. et al., 2010. Low-overhead diskless checkpoint for hybrid computing systems. *17th International Conference on High Performance Computing, HiPC 2010*.

[3] Elghirani, A. et al., 2008. Performance enhancement through hybrid replication and genetic algorithm co-scheduling in data grids. *AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications*, pp.436–443.

[4] El-sayed, N. & Schroeder, B., 2014. To Checkpoint or Not to Checkpoint : Understanding Energy-Performance-I / O Tradeoffs in HPC Checkpointing.

[5] Guermouche, A. et al., 2011. Uncoordinated checkpointing without domino effect for send-deterministic MPI applications. *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011*, pp.989–1000.

[6] Khoshkholghi, M.A. et al., 2017. Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers. *IEEE Access*, 3536(c), pp.1–13.

[7] Kliazovich, D., Bouvry, P. & Khan, S.U., 2013. DENS: Data center energy-efficient network-aware scheduling. *Cluster Computing*, 16(1), pp.65–75.

[8] Kumar, N. et al., 2014. Achieving quality of service (QoS) using resource allocation and adaptive scheduling in cloud computing with grid support. *Computer Journal*, 57(2), pp.281–290.

[9] Li, B. et al., 2014. Resource availability-aware advance reservation for parallel jobs with deadlines. , pp.798–819.

[10] Li, B. et al., 2012. Scheduling Strategies for Deadline Constrained Coallocation Jobs in Distributed Computing Environments. , 6(February), pp.232–240.

[11] Palaniswamy, C., Analytical and Comparison Incremental of Periodic State Checkpointing Saving *. , pp.127–134.

[12] Patel, P.K. & Jethva, P.H.B., 2013. Priority based scheduling for Lease management in cloud computing. *IEEE*, 1(2), pp.193–196.

[13] Pei, J. et al., 2015. Scheduling jobs on a single serial-batching machine with dynamic job arrivals and multiple job types. *Springer*.

[14] Rodger, J.A., 2016. Informatics in Medicine Unlocked Discovery of medical Big Data analytics : Improving the prediction of traumatic brain injury survival rates by data mining Patient Informatics Processing Software Hybrid Hadoop Hive. *Informatics in Medicine Unlocked*, 1(2015), pp.17–26. Available at: http://dx.doi.org/10.1016/j.imu.2016.01.002.

[15] Salehi, M. et al., 2016. Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems. , pp.1–12.

[16] Schroeder, B. & Gibson, G. a., 2007. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you. *Conference on File and Storage Technologies (FAST)*, (September), pp.1–16. Available at: http://www.usenix.org/event/fast07/tech/schroeder/schroeder.pdf.

[17] Singh, D., Singh, J. & Chhabra, A., 2012. High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms. *Proceedings - International Conference on Communication Systems and Network Technologies, CSNT 2012*, pp.698–703.

[18] Switalski, P. & Seredynski, F., 2014. Scheduling parallel batch jobs in grids with evolutionary metaheuristics. *Journal of Scheduling*, 18(4), pp.345–357. Available at: http://dx.doi.org/10.1007/s10951-014-0382-0.

[19] Xhafa, F. et al., 2011. A GA+TS hybrid algorithm for independent batch scheduling in computational grids. *Proceedings - 2011 International Conference on Network-Based Information Systems, NBiS 2011*, pp.229–235.

[20] Zhou, A., Sun, Q. & Li, J., 2017. Enhancing Reliability via Checkpointing in Cloud Computing Systems. *IEEE*, pp.108–117.

[21] Bhartee, A.K. et al., 2015. Advance Reservation Policy for Inter-Member Workflow Scheduling in Community Cloud. , 4(4), pp.2891–2894.

[22] Alkhanak, E.N. et al., 2015. Cost-aware challenges for workflow scheduling approaches in cloud computing environments : Taxonomy and opportunities. *Future Generation Computer Systems*, 50, pp.3–21. Available at: http://dx.doi.org/10.1016/j.future.2015.01.007.

[23] Armbrust, M. et al., 2010. A view of cloud computing. *Communications of the ACM*, 53(4), p.50. Available at: http://portal.acm.org/citation.cfm?doid=1721654.1721672.

[24] Sharma, Y. et al., 2016. Reliability and energy efficiency in cloud computing systems: Survey and taxonomy. *Journal of Network and Computer Applications*, 74, pp.66–85.

[25] Zhou, A. et al., 2014. On Cloud Service Reliability Enhancement with Optimal Resource Usage. , 7161(c), pp.1–14.

[26] Bama, P.R.K.S., 2009. Heuristics Aware Advance Reservation and Scheduling (HAARS) mechanism in Hybrid (Grid / Cloud) environment.

[27] Prajapati, H.B. & Shah, V.A., 2014. Scheduling in Grid Computing Environment. *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, pp.315–324. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6783470.

[28] Nirmala, S.J., Tajunnisha, N. & Bhanu, S.M.S., 2016. Service provisioning of flexible advance reservation leases in IaaS clouds. , 3(3), pp.154–162.

[29] Sindhu, S. & Mukherjee, S., Efficient Task Scheduling Algorithms for Cloud. , pp.79–80.

[30] Sulistio, A., Buyya, R. & Engineering, S., A GRID SIMULATION INFRASTRUCTURE SUPPORTING ADVANCE RESERVATION.

[31] Min, R.U.I. & Maheswaran, M., SCHEDULING ADVANCE RESERVATIONS WITH PRIORITIES IN GRID COMPUTING SYSTEMS.

[32] Manikandaprabhu, M. & Sivasenthil, R., 2013.

Resource Provisioning Cost of Cloud Computing by Adaptive Reservation Techniques. , 4(May), pp.1118–1124.

[33] Chaisiri, S. & Member, S., 2012. Optimization of Resource Provisioning Cost in Cloud Computing. , 5(2), pp.164–177.