# Parallel Cluster Resource and Job Allocation Using Hybrid AHP Approach

Divya Dogra[1], Mohit Trehen[2]
[1]Dept. of CSE, GCET, Gurdaspur, PTU Kapurthala, Punjab, India
[2]Dept. of CSE, GCET, Gurdaspur, Punjab, India

*Abstract-* **Resource availability and usability becomes critical as mass user becomes need of the hour for advanced and heterogeneous computing. Federated cloud computing becomes building block of current environment as resources are limited and requirements are heavy. To overcome the problem deadline constraint scheduling mechanism is proposed under heterogeneous cloud computing. Reliability is a concern which is tackled in the proposed literature considering metric mean time between failures. Deadline sensitive jobs are compared against finish time of jobs which is calculated in advance and AHP is applied to determined there completion time. AHP matrix indicates if jobs satisfy deadline or not. Advance reservation is also used within the proposed system to handle deadline constraint jobs. Early finish time calculated in the scenario is used to preempt the resources in case resources become critically low and jobs cannot be allotted to the clusters. To resolve the faults, progress made at current machine is migrated to other machine using check pointing approach. The result of the proposed system shows improvement in terms Make span and Flow time by 10%.**

**Index Terms- AHP, Deadline, federated cloud, Make span, Flow time.**

## 1. INTRODUCTION

Resource availability and job execution becomes need of the hour in case of resource constraint environment. (Khoshkholghi et al. 2017)Cloud environment with homogeneous environment may not be sufficient to execute jobs in time critical and deadline sensitive (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 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. system only if its requirements will be satisfied otherwise jobs are prompted from the system. 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 might 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 gives literature survey of various job scheduling mechanisms along faults and failure tackling mechanism. Section 3 gives the experimental setup, section 4 gives the methodology, section 5 gives the performance comparison, section 6 gives the conclusion and future scope and last section gives references.

## 2. LITERATURE SURVEY

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.

### 2.1 Checkpointing for fault tolerance

(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.

In addition to checkpointing strategy, job scheduling is also critical to ensure load balancing. These strategies including multiple objectives are given as under:

### 2.2 Job Scheduling

(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 extremal 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 a 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.

### 2.3 Metric for Measurements

(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 failure 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.

## 3. EXPERIMENTAL SETUP

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

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

Table 1: Experimental Setup

## 4. PROPOSED SYSTEM

The proposed system consists of clusters which are heterogeneous in nature. These clusters contain machines or resources which are to be assigned to the jobs. 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. Deadline is matched against the finish time to determine fitness of the job and resource. Checkpointing is established to enhance reliability and measurement metric which is used is mean time between failures. The proposed scheme is listed as follows:

```
1. Input job list
        Job list can be obtained through dataset
        Or through user input
        Or through direct initialization
        K=0.5 or 2
Job Selection process(Advance Reservation)
Perform Job ordering by checking job requirement against available machines and reject the jobs not lying within sequence
Selection of processor
Check for Deadline and Max MTBF if found goto step b.
Check for Security parameter (Max(Security(VM)) if found goto step c
Processor selection on the basis of Machine_available.
If
processori_available>processori+1_Available_Cluste
r
```

```
Max_Available_Cluster=processori
Processor selection on the basis of MTTF
If processori_MTTF>processori+1_MTTZF
Min_MTTF_Processor=processori
Allocate job to Max_speed_processor and Min_MTTF_processor
Check for deadline meet condition
Deadlinei=DeadLinei+Job_Arrival+K*Jobs_Burst_Time
If  Dead_line_jobi>=Actual_Deadlinei
Obtain result in terms of Makespan and Flowtime
Else
Go to step 2
Check for availability processor
This is performed to allocate next job in sequence to optimal processor
Availability_i=(Burst_time)/Speed
        If processor_availablei==true
Allocate the job and go to step4
```

c) 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.

## 5. PERFORMANCE ANALYSIS AND RESULTS

This section gives the performance analysis in terms of Makespan and Flowtime. Makespan is the time taken to complete entire schedule of jobs and Flowtime is the time taken to complete individual job. 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

| Load | Existing Makespan | Proposed Makespan |
|---|---|---|
| 1 | 7.07E+04 | 6.09E+04 |
| 2 | 8.25E+04 | 7.65E+04 |
| 3 | 2.47E+05 | 2.45E+05 |

Table 2: Comparison with the variation of load

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

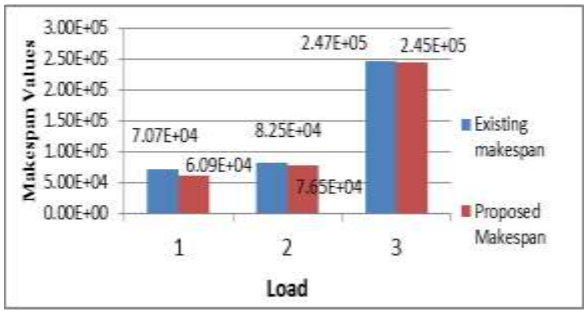

Figure 1: Plot corresponding to Makespan when load is increased

Load impact on Flowtime is also observed. Flowtime also increases as load increases. This is given in terms of following table.

| Load | Existing Flowtime | Proposed Flowtime |
|---|---|---|
| 1 | 2.28E+04 | 2.17E+04 |
| 2 | 2.68E+04 | 2.63E+04 |
| 3 | 5.05E+04 | 4.95E+04 |

Table 3: Flowtime comparison

The comparison of Flowtime also show hike as load increases. This variation is also depicted through the plots as

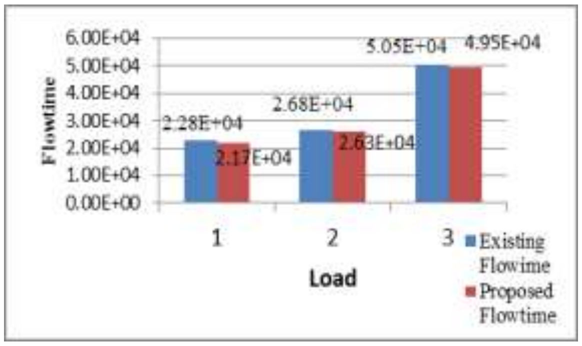

Figure 2: Plot comparison of Flowtime with load

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

| Processor | Existing Makespan | Proposed Makespan |
|---|---|---|
| 64 | 4.73E+07 | 3.60E+07 |
| 96 | 40146410 | 35813194 |
| 128 | 39523525 | 35013194 |

Table 4: Comparison of Makespan when processor varies

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

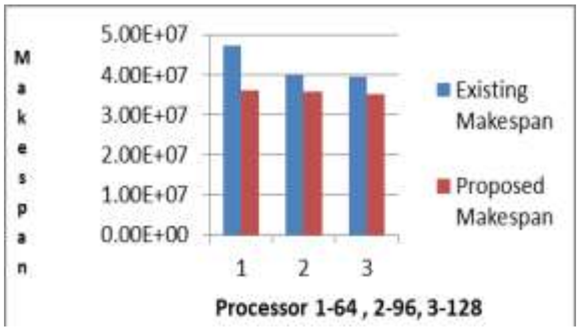

Figure 3: Plots in terms of Makespan when processor varies

The results in terms of Flowtime when processor increases is given as under

| Processor | Existing Flowtime | Proposed Flowtime |
|---|---|---|
| 64 | 33.03 E+07 | 2.50E+07 |
| 96 | 20146410 | 19029194 |
| 128 | 19523525 | 17229194 |

Table 5: comparison in terms of Flowtime when processor changes

As the processor increases Flowtime decreases considerably shown through the plots also

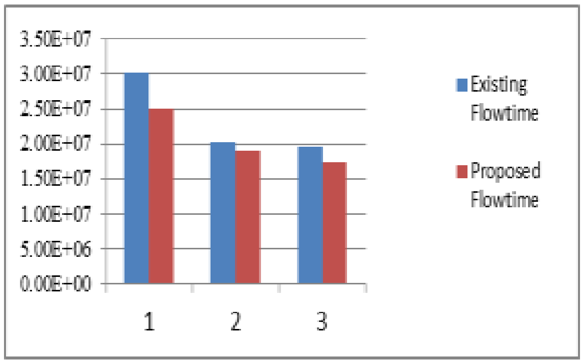

Figure 4: Comparison in terms of Flowtime when processor increases

Constant metric K when varies between 0.1 to 2 also yield distinct results in terms Makespan and Flowtime. These result are given in terms of table as

| K-min/K-max | Existing Makespan | Proposed Makespan |
|---|---|---|
| 0.5 | 4.73E+07 | 4.23E+07 |
| 1 | 59146410 | 58829194 |
| 1.5 | 69523525 | 67029194 |

Table 6: Makespan comparison when K is varied between 0.5 to 1.5

Result in terms of plots is given as under

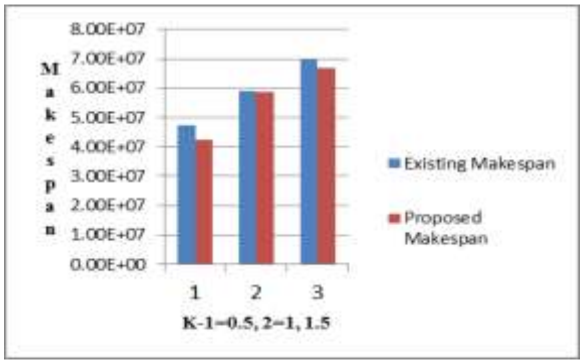

Figure 5: gives the comparison of Makespan obtained in terms of K

Comparison in terms of Flowtime with variation in K is given as under

Table 7: Comparison of Flowtime when K is varied

| Kmin-kmax | Existing flowtime | Proposed Flowtime |
|---|---|---|
| 0.5 | 4.72E+07 | 4.62E+07 |
| 1 | 59096410 | 58779194 |
| 1.5 | 69473525 | 68979194 |

The Flowtime plots when K is distinct is given as under

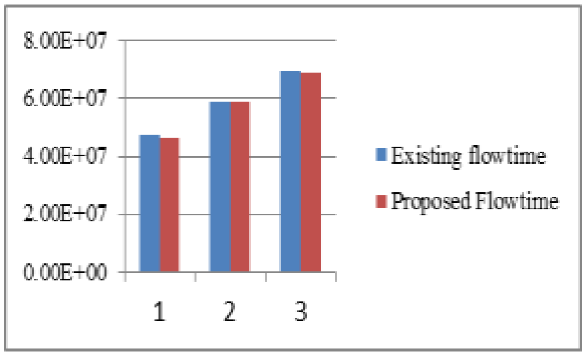

Figure 6: Flowtime comparison when K is varied.

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 enhancement by the factor of 10% is observed through the proposed methodology.

## 6. CONCLUSION AND FUTURE SCOPE

The parallel job allocation along with reliability metric is need of the hour within advanced computing system. This paper uses the Clusters, reliability metrics and job scheduling to achieve optimal results in terms of Makespan and Flowtime. Resource availability is ensured using the early finish strategy. The resources are dynamically prompted when the job becomes finished. Earlier resources held by the job are not prompted as long it is in the system causing lack of resource availability. Load balancing if not successful and machine faulted then checkpointing strategy is in place to restore the system to the stable state. Performance is enhanced by the margin of 10% which is significant proving worth of the study.

In future this approach can be collaborated with multiheuristic 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 Checkpointor 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.