Hadoop Job Scheduling Using Improvised Ant Colony Optimization

G. Joel Sunny Deol[1], Dr.O. NagaRaju[2]

[1]Research Scholar, Dept of CSE, Acharya Nagarjuna University
[2]Asst. Professor, Department of Computer Science, Government Degree College, Macherla-522426,
sunnydeolgosu@gmail.com.

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: Hadoop Distributed File System is used for storage along with a programming framework MapReduce for processing large datasets allowing parallel processing. The process of handling such complex and vast data and maintaining the performance parameters up to certain level is a difficult task. Hence, an improvised mechanism is proposed here that will enhance the job scheduling capabilities of Hadoop and optimize allocation and utilization of resources. Significantly, an aggregator node is added to the default HDFS framework architecture to improve the performance of Hadoop Name node. In this paper, four entities viz., the name node, secondary name node, aggregator nodes, and data nodes have been modified. Here, the aggregator node assigns jobs to data node, while Name node tracks aggregator nodes. Also, based on the job size and expected execution time, an improvised ant colony optimization method is developed for scheduling jobs.In the end, the results demonstrate notable improvisation over native Hadoop and other approaches.

Keywords-Hadoop, Virtualization, MapReduce, Job Scheduling, Improved Ant Colony Optimization.

1. Introduction

Hadoop clusters have acquired great acceptance for their efficiency in computation, thereby helping save time and cost.Hadoopcomprises an HDFS and MapReduce as its two mainstays. The users are endowed with the facility of distributed storage access by the HDFS, while distributed processing is offered by MapReduce.The name node (master node) and the data nodes (slave nodes) build up the HDFS, which ensure that distributed environments and storage facilities are efficiently managed. The tasks are run on a cluster in MapReduce which allows data to be managed in a distributed data storage system.In fact, MapReduce splits input dataset and creates many blocks, measuring 64 or 128 MB, before storing in an HDFS.The two functions used by MapReduce component are as follows:

- **Mapper: E**ach block requires a map operation to run that is kept separate from rest of the blocks, while ensuring that the data node is exactly place in the data storage site. The mapping functions, i.e., < key, value > for each term, such as < xyz, 1 > are performed by this operation at this phase.
- **Reducer:** The results supplied by different mappers are combined with the reducer operation for producing a single final output.

MapReduce consists two components (1) Job Tracker (A central component) (2) Task Tracker (Distributed Component).

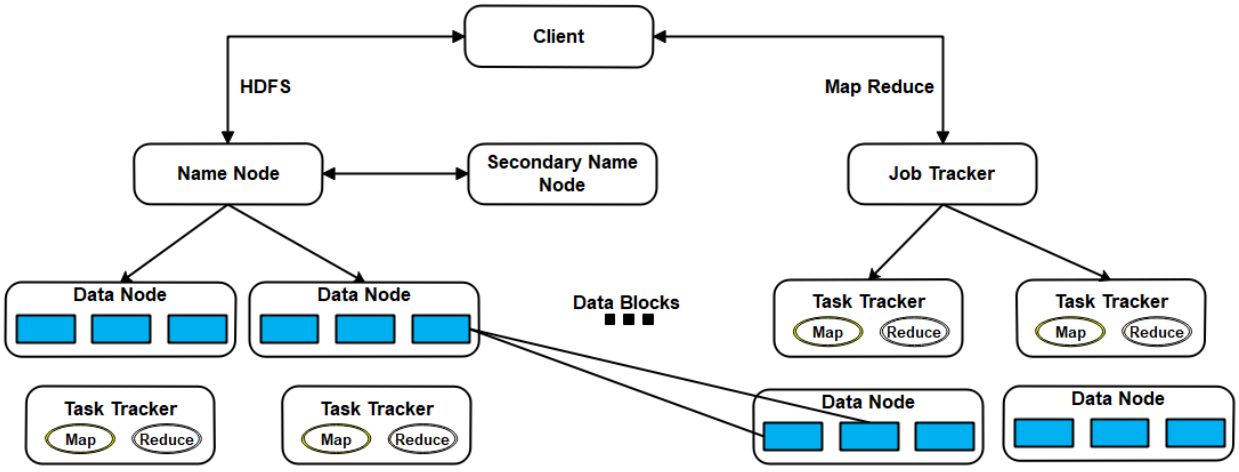

Fig.1. Hadoop System Architecture

Job Tracker: Job tracker primarily handles scheduling andprocessing of allthe jobs. Whenever job tracker receives jobs, task tracker tends to be assigned that particular job. The task tracker then has the responsibility of coordinating the execution process of the job. It is run onMaster node i.e. Name node.

Task Tracker: Theperform map and reduce functions are performed by a task tracker for a particular job assigned by job tracker. The job tracker receives report about the status of an assigned work. Each task tracker is assigned many slots as part of 'map andreduce function' required in performing a task. The balance of map to

reducetask is also an important consideration which is managed by JVM. The Task trackerrun on slave node i.e. Data node.

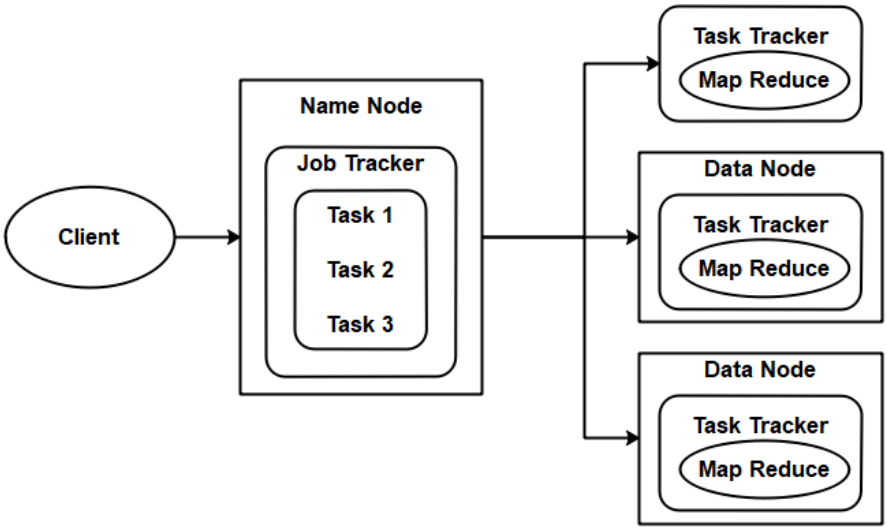

Fig 2: Job and Task Tracker

The task tracker is assigned a job by the jobtracker which is submitted by a user and that job is processed in task tracker. The Task tracker will update the job Tracker with current status.

1.2. Virtualization

Virtualization is a concept of dividing the resource of the same hardware tomultiple operating system (same or different) and applications at same time,achieving higher efficiency and resource utilization.Hypervisor or virtual machine monitor is the manager who carries out the process of virtualization.

- **Bare metal hypervisor** run directly on top of hardware to have more control power overhardware.
- **Hosted hypervisor** run on the top of conventional operating system along with some native processes.

It will differentiate the guest and host operating system running on the single hardware.

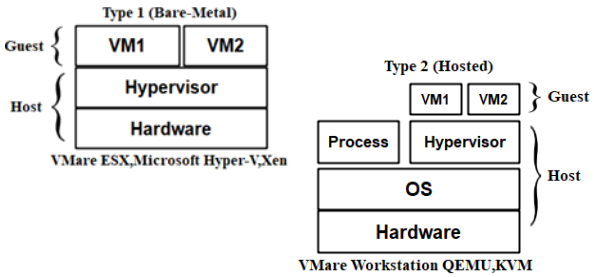

Fig.3.Virtual Machine Monitors

1.2.1 Types of Virtualization

Virtualization can be utilized in different areas to enhance performance and resource sharing. So various types of virtualizations are as following:

Server Virtualization: Server virtualization is a process in which multiple OS and applications run on a single server so that resources of that single host will be divided among all the operating system and the various applications. It brings the resource utilization and the cuts the cost by having only one server in place of many.

Application Virtualization: Application virtualization enables the end user to access the application from remotely located server. Application is not installed on every user desktop but is available on user ‘s demand. So, it is very cost efficient for the organizations.

Desktop Virtualization: It is similar to server virtualization. In desktop virtualization the workstation with all its application is virtualized. The hypervisor contains the customization and preferences of an application. It runs on a centralized mechanism so user can access the desktop from any location. It provides higher efficiency and the cost reduction.

Storage Virtualization: Storage virtualization is a technique in which the storage from the multiple hardware devices will be combined to act as a single storage device. In this the storage space by SAN and NAS

is virtualized along with the disk file of virtual machine. So, it provides the disaster recovery management by replication.

Network Virtualization: Network virtualization is a concept in which the all the network from different network devices are combined into a single network called asa virtual network so that its bandwidth will be divided to channels which will be given to the server or device.

Hadoop Map Reduce is used to compute the huge amount of complex data in tolerable elapsed time, because all the processes performed to analyze data will be done in parallel on different nodes. Although Hadoop provides many functionalities but when it comes to manage and providing resources for each upcoming request, it will become a difficult task to handle it. So, with the built-in features of Hadoop there is a need to virtualize it. Virtualizing the Hadoop clusteradds new features to it.

- It brings elasticity as cluster can be expanded or reduced by adding or removing the nodes on demand. The whole process is very fast.
- A physical cluster will be shared between multiple virtual clusters so that the physical cluster will be reused which enables resource utilization.
- Roles of task tracker and data nodes will be separated into different machines for achieving high security as they both have their own access authorization.
- After virtualization of physical cluster, cloning of single image (for e.g. cloning of data node) can be performed which reduce the cost and enhance the performance.It also adds new features to it.

When their functioning was analyzed, a majority of the known schedulers, including LATE and FCFS failed in performing better. In fact, they were found not being able to utilize resources in balanced ways. Moreover, the workload of a job is neglected by schedulers, thereby causing imbalances in resource utilization. In order to enhance Hadoop performance,new methodsfor job scheduling, resource utilization and allocation have been proposed in this paper.The Amazon EC2 nodes are applied wherein a particular node is designated as the master node, while other nodes are designated as slave nodes. Every master node in the proposed HDFS cluster is found to be populated by many aggregator nodes, while in the slave nodes, map, reduce, and shuffle functions are assigned as a three-phased process.

2. Proposed Work

2.1. Scheduling Based on Improved Ant Colony Optimization Algorithm

Let the task set $\mathbf{T}=\{\mathbf{t}_1,\mathbf{t}_2...,\mathbf{t}_n\}$, $\mathbf{VM}=\{\mathbf{vm}_1,\mathbf{vm}_2,\mathbf{vm}_m\}$ stands for $\mathbf{n}$ tasks and $\mathbf{m}$ virtual. The matrix $\mathbf{K}$ is used to signifies the relation of a task on a VM [20]

$$\mathbf{K}=\begin{pmatrix} \mathbf{k}_{11} & \mathbf{k}_{12} & \dots & \mathbf{k}_{1n} \\ \mathbf{k}_{21} & \mathbf{k}_{22} & \dots & \mathbf{k}_{2n} \\ \dots & \dots & \dots & \dots \\ \mathbf{k}_{m1} & \mathbf{k}_{m2} & \dots & \mathbf{k}_{mn} \end{pmatrix}$$

Here, $\mathbf{K}_{ij}$ stands for the procedure in which jobs between $\mathbf{i}^{th}$ job and the $\mathbf{j}^{th}$ virtual machine has been allocated. The matrix $\mathcal{ET}$ represents the time required to execute the job on the virtual machine as given below[20]:

$$\mathcal{ET}=\begin{pmatrix} \mathbf{et}_{11} & \mathbf{et}_{12} & \dots & \mathbf{et}_{1n} \\ \mathbf{et}_{21} & \mathbf{et}_{22} & \dots & \mathbf{et}_{2n} \\ \dots & \dots & \dots & \dots \\ \mathbf{et}_{m1} & \mathbf{et}_{m2} & \dots & \mathbf{et}_{mn} \end{pmatrix} \text{ where } \mathbf{et}_{ij}=\frac{\mathbf{t}_{length\_i}}{\mathbf{vm}_{comp\_j}}$$

Represents the execution time of task $\mathbf{i}$ on virtual machine $\mathbf{j}$. $\mathbf{t}_{length\_i}$ indicates the length of job $\mathbf{i}$. $\mathbf{vm}_{comp\_j}$ represents the processing power of the virtual machine $\mathbf{j}$. $\mathbf{vm}_{comp\_j}$ is calculated as [20]:

$$\mathbf{vm}_{comp\_j}=\mathbf{vm}_{mips\_j}\times\mathbf{vm}_{cpu\_j}$$

$\mathbf{vm}_{mips\_j}$ represents the computing power of the virtual machine $\mathbf{j}$.For a given virtual machine $\mathbf{j}$. $\mathbf{vm}_{cpu\_j}$ represents the no.of CPUs.For a given task the transmission time can givenby $\mathcal{ER}$ matrix[20].

$$\mathcal{ER} = \begin{pmatrix} \mathbf{er}_{11} & \mathbf{er}_{12} & \dots & \mathbf{er}_{1n} \\ \mathbf{er}_{21} & \mathbf{er}_{22} & \dots & \mathbf{er}_{2n} \\ \dots & \dots & \dots & \dots \\ \mathbf{er}_{m1} & \mathbf{er}_{m2} & \dots & \mathbf{er}_{mn} \end{pmatrix} \text{ where } \mathbf{er}_{ij} = \frac{\mathbf{t}_{\mathbf{inputfilesize\_i}}}{\mathbf{vm}_{\mathbf{bw\_j}}}$$

where $\mathbf{er}_{ij}$ indicates the time when the $\mathbf{i}$ task is transferred to the $\mathbf{j}$ virtual machine.For task $\mathbf{i}$ $\mathbf{t}_{\mathbf{inputfilesize\_i}}$ represents the size of the data $\mathbf{i}$. The bandwidth of the virtual machine $\mathbf{j}$ is defined by $\mathbf{vm}_{\mathbf{bw\_j}}$. The completion time of a particular mission is, thus[20],

$$\mathbf{Job\,RunTime}_{ij} = \mathbf{et}_{ij} + \mathbf{er}_{ij}$$

Then the virtual machine completion time $\mathbf{j}$ is the sum of the completion times of all the tasks assigned to the virtual machine[20]

$$\mathbf{vm}_{\mathbf{completetime\_j}} = \sum_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{JobRunTime}_{\mathbf{ij}}$$

Since any virtual machine is running at the same time, the time to complete the mission is the time when the last virtual machine is running. It can be representedas[20]

$$\mathbf{C\_Time(I) = max(vm}_{\mathbf{completetime\_j}}\mathbf{)} \quad \mathbf{j \in [1, n]}$$

Assuming the unit cost of virtual machine $\mathbf{j}$ is $\mathbf{U}$, the cost of completing all subtasks is defined as[20]

$$\mathbf{TotalCost(I)} = \sum_{\mathbf{j=1}}^{\mathbf{n}} (\mathbf{vm}_{\mathbf{completetime\_j}}) \times \mathbf{UCost}_{\mathbf{j}}$$

In the model, each particlemust be allocated the virtual machine of its choice to perform a designated task. The optimized target is used by the ants to identify an optimal matching scheme.The optimal solution is found by the ants in parallel, which allows them to communicate and pass information to each other. Here, the pheromone was adjusted to the task aligned on the virtual machine path, which determines which ant to be selected for the other ant as well as readies the next iteration for the ant [20].

Algorithm: *Improved Ant Colony Optimization for Scheduling*

1.Set the pheromone in motion. Set as many iterations as possible, the energy component of the pheromone α ,the heuristic factor c β ,the unpredictable factors $\mathbf{r}$ and p1 $\mathbf{r}_1$, the number of ants $\mathbf{m}$, and $\mathbf{p}_0$

2. Randomly position all the ants at the start $\mathbf{vm's}$.

3. Each ant computes on a set of optional Virtual Machines the likelihood of the current mission chosen on each virtual machine.

$$\mathbf{RS} = \begin{cases} \mathbf{argmax}_{\mathbf{j \in allowed_k}} \{ \tau_{ij}^{\alpha}(t) \eta_{ij}^{\beta} \}, & \mathbf{if\ q < q_0} \\ \mathbf{S_1} & \mathbf{otherwise} \end{cases}$$

$$\mathbf{S_1} = \mathbf{p}_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}(t)]^{\beta}}{\sum_{\mathbf{j \in allowed_k}} [\tau_{ij}(t)]^{\alpha} * [\eta_{ij}(t)]^{\beta}} & \mathbf{if\ j \in allowed_k} \\ \mathbf{0} & \mathbf{otherwise} \end{cases}$$

4.If an ant completes a solution, upgrade the pheromone on the direction of the ant

$$\tau_{ij}(t+1) = (1 - r)\tau_{ij}(t) + r\Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t) = \begin{cases} \dfrac{1}{L}, & \mathbf{if\ (i,j) \in T^k(t)} \\ 0 & \mathbf{otherwise} \end{cases}$$

Compare and change the optimum solution with the previous optimal solution.

5. If all the ants finish their tour, calculate $\mathbf{N = N + 1}$ (**N** is the number of iterations), calculate the optimal solution globally and change the pheromone otherwise, repeat step3.

$$\tau_{ij}(t+1) = (1 - r_1)\tau_{ij}(t) + r_1 \Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L_{best}}, & \text{if } (i,j) \in T^{+} \\ 0 & \text{otherwise} \end{cases}$$

6.If the current number of iterations is less than the limit, go back to Step 2. If not, the iteration will stop, and the best solution will be given.

2.2. Artificial Neural Network for Node Usage Prediction

The suggested ANN is based on the wording of the node and the node of aggregation. Accept a set of input variables and determine the weights to reach the output variable of the input variables. This procedure can be defined as an entry, an activation, and an output package [19].

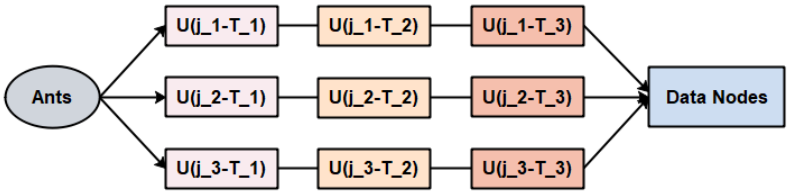

Fig.4 Ant execution job strategy based on data nodes

In the hidden layer the feedback and bias of other neurons in the input layer is weighted by each neuron [19].

$$Z_i = \left(\sum_{K=1}^{N_j - 1} x_k^{j-1} w_{k,i} - b_k \right)$$

If $w_{k,i}$ is the value of the relation between the node K and all the nodes in the input layer, x_k^{j-1} is the input of the node K^{th} in the j^{th} layer, $N_j - 1$ is the total number of nodes in the $j-1$ layer, and b_k is the bias of the node. The sum of these weighted values with the activation function is forwarded to determine the output of the node. The output is determined as follows [19].

$$y_i = f(Z_i)$$

For the activation function, the sigmoid function can be used and is formulated accordingly [19].

$$f_{Z_i} = \frac{1}{1 + e^{-Z_i}}$$

This equation is used to model the respective non-linear conducts [19].

3. Results and Experiments

A simulation platform is applied to implement the experimentwith 2 Datacenters and 50-250 tasks. The task runs within a range of 5000 MI (Million Instructions)-100000 MI. The cloud simulator is set as per the following parameters as shown in Table 1, while virtual machine are 10 in number.

Table 1. CloudsimSetting

Entity Type	Parameters	Values
Task (Cloudlet)	Task Length	5000-10000
	Task in number (Total)	50-250
	File Size	300-5000
Virtual Machine (**vm**)	**pe** (MIPS)	514-1028
	vm in total	0-10
	pe with **vm**	01-03
	Bandwidth of the System	500-1000
	Memory allocated	1024-2048
	Storage allocated	1000-4000

	vm per unit	1-100

This paper predicted the above algorithm parameters. In addition, before selecting the right collection of parameters as parameters to experiment, the efficiency of 10 groups of different parameters α , β and $\mathbf{r}$ was evaluated and compared.

Table.2. Parameters setting of purposed model

Parameters	Values
α	1
β	5
$\mathbf{r}$	0.4
$\mathbf{r_1}$	0.5
No.of ants(m)	10
$\mathbf{q_0}$	0.9
Max Number of iterations ($\mathbf{N_{max}}$)	30

The makespan is analyzed on all categories of synthetic dataset for batch sizes of100 to 1000 with the difference of 100 in batch sizes.

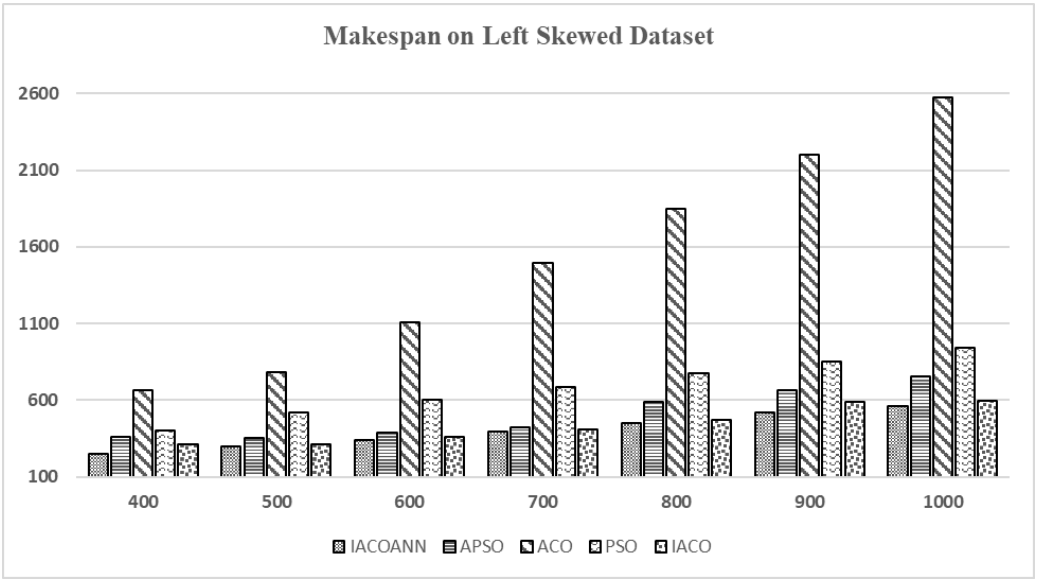

Fig.5. Makespan on Left Skewed Dataset

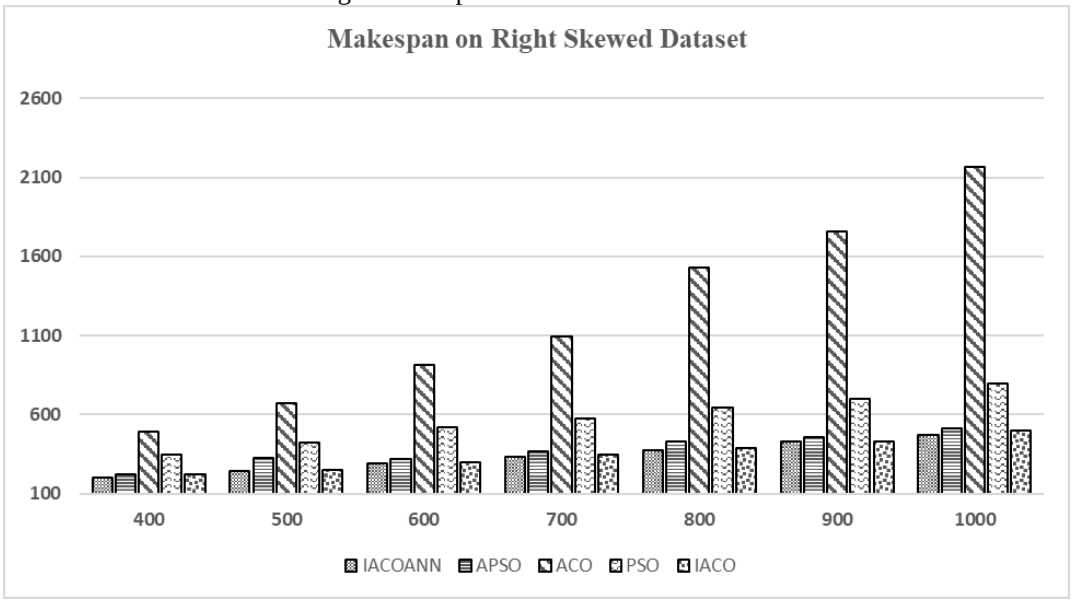

Fig.6. Makespan on Right Skewed Dataset

The Fig.6. shows the makespan analysis on right skewed dataset. It is evidentthat in most of the batch sizes theIACOANNshowed better makespan.

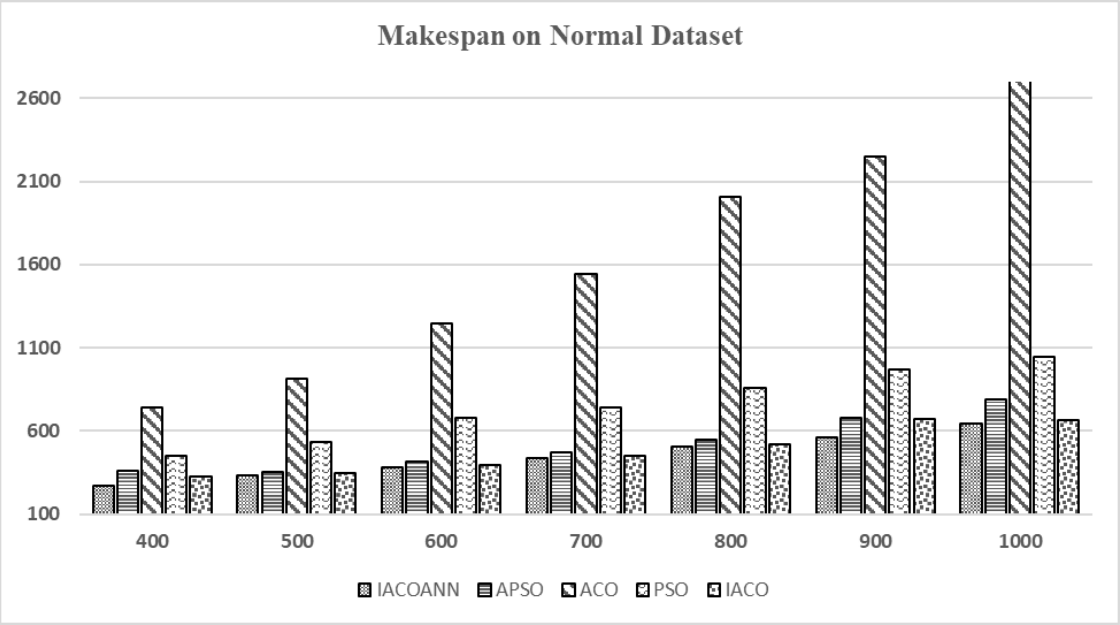

Fig.7. Makespan on Normal Dataset

The Fig.7. shows average percentage of improvement of IACOANN against other techniques. IACOANNhas achieved better makespan on all batch sizes of normal dataset.

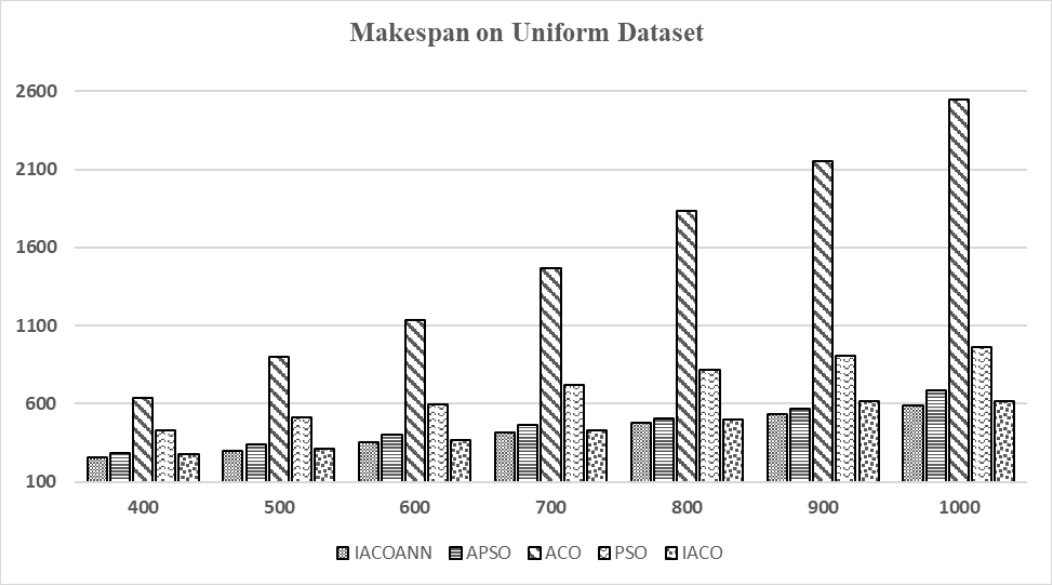

Fig.8. Makespan on Uniform Dataset

4. Conclusion

The task scheduling problem in cloud computing requires the efficient mappingof jobs to virtual resources. Due to the heterogeneity of jobs and resources manypossible mappings can be defined. The heuristic and meta-heuristic schedulers areutilized to map independent jobs. The meta-heuristic has potential to explore thehuge search space of possible solutions. The genetic based IACOANN has been presentedin the research to improve the makespan and resource utilization.

References

1. D. Laney. (2012, 2015). Deja VVVu: Others Claiming Gartner's Construct for Big Data. Available: http://blogs.gartner.com/doug-laney/deja-vvvue-others-claiming-gartners-volume-velocity-variety-construct-for-big-data/
2. M. Isard, et al., "Dryad: distributed data-parallel programs from sequential building blocks," in ACM SIGOPS Operating Systems Review, 2007, pp. 59-72.
3. T. White, Hadoop: The Definitive Guide, Second Edition: O'Reilly Media Inc., 2010.
4. V. K. Vavilapalli, et al., "Apache hadoop yarn: Yet another resource negotiator," in Proceedings of the 4th annual Symposium on Cloud Computing, 2013, p. 5.
5. M. Zaharia, et al., "Spark: cluster computing with working sets," in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010, pp. 10-10.
6. S. Leo. (2012). Hadoop Wiki - PoweredBy. Available: http://wiki.apache.org/hadoop/PoweredBy

A. Verma, et al., "ARIA: automatic resource inference and allocation for mapreduce environments," in Proceedings of the 8th ACM international conference on Autonomic computing, 2011, pp. 235-244.

7. (2012). Capacity Scheduler Guide. Available: http://hadoop.apache.org/docs/stable/capacity_scheduler.html
8. M. Zaharia, et al., "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in Proceedings of the 5th European conference on Computer systems, 2010, pp. 265-278.
9. T. Sandholm and K. Lai, "Dynamic proportional share scheduling in hadoop," in Job scheduling strategies for parallel processing, 2010, pp. 110-131.
10. X. Zhang, et al., "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments," in Cloud and Service Computing (CSC), 2011 International Conference on, 2011, pp. 235-242.
11. Y. Luo and B. Plale, "Hierarchical MapReduce Programming Model and Scheduling Algorithms," in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012, pp. 769-774.
12. (2012). HOD Scheduler. Available: http://hadoop.apache.org/docs/r0.21.0/hod_scheduler.pdf
13. M. Zaharia, et al., "Improving mapreduce performance in heterogeneous environments," in Proceedings of the 8th USENIX conference on Operating systems design and implementation, 2008, pp. 29-42.
14. M. Zaharia, et al., "Job scheduling for multi-user mapreduce clusters," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-55, 2009.
15. M. Hammoud and M. F. Sakr, "Locality-Aware Reduce Task Scheduling for MapReduce," in Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, 2011, pp. 570-576.
16. J. Polo, et al., "Performance-driven task co-scheduling for mapreduce environments," in Network Operations and Management Symposium (NOMS), 2010 IEEE, 2010, pp. 373-380.

A. Verma, et al., "Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance," in Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on, 2012, pp. 11-18.

17. Rayan Alanazi, Fawaz Alhazmi, Haejin Chung,Yunmook Nah. "A Multi-Optimization Techniquefor Improvement of Hadoop Performance with aDynamic Job Execution Method Based onArtificial Neural Network", SN ComputerScience, 2020
18. Qiang Guo" Task Scheduling Based on Ant Colony Optimization in Cloud Environment" 2017 5th International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation (CDMMS 2017) AIP Conf. Proc. 1834, 040039-1–040039-11; doi: 10.1063/1.4981635 Published by AIP Publishing. 978-0-7354-1504-1/$30.00