# A Disparateness-Aware Scheduling using K-Centroids Clustering and PSO Techniques in Hadoop Cluster

**E. Laxmi Lydia[1],Dr. M.Ben Swarup[2]**

**[1]Associate Professor, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, Andhra Pradesh, India.**

**[2]Professor, Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam,**

**Andhra Pradesh, India.**

**Email:elaxmi2002@yahoo.com**

**Abstract.** Big data storage management is one of the most challenging issues for Hadoop cluster environments, since large amount of data intensive applications frequently involve a high degree of data access locality. In traditional approaches high-performance computing consists dedicated servers that are used to data storage and data replication. Therefore to solve the problems of Disparateness among the jobs and resources a "Disparateness-Aware Scheduling algorithm" is proposed in the cluster environment. In this research work we represent K-centroids clustering in big data mechanism for Hadoop cluster. This approach is mainly focused on the energy consumption in The Hadoop cluster, which helps to increase the system reliability. The Hadoop cluster consists of resources which are categorized for minimizing the scheduling delay in the Hadoop cluster using the K-Centroids clustering algorithm. A novel provisioning mechanism is introduced along with the consideration of load, energy, and network time. By integrating these three parameters, the optimized fitness function is employed for Particle Swarm Optimization (PSO) to select the computing node. Failure may occur after completion of the successful execution in the network. To improve the fault tolerance service, the migration of the cluster is focused on the particular failure node. This can recomputed the node by PSO and the corresponding optimal node is predicted. The experimental results exhibit better scheduling length, scheduling delay, speed up, failure ratio, energy consumption than the existing systems.

**Keywords:** K-Centroids Clustering, Big data, Hadoop Cluster, data access locality, data replication, systemreliability, particle swarm optimization

# 1. Introduction

In recent years, big data has rapidly developed into a hotspot that attracts great attention from academia, industry, and even governments around the world [1 – 2]. Nature and Science have published special issues dedicated to discuss the opportunities and challenges brought by big data [3, 4]. McKinsey, the well-known management and consulting firm, alleged that big data has penetrated into every area of today's industry and business functions and has become an important factor in production [5]. Using and mining big data heralds a new wave of productivity growth and consumer impetus. O'Reilly Media even asserted that "the future belongs to the companies and people that turn data into products" [6]. Some even say that big data can be regarded the new petroleum that will power the future information economy. In short, the era of big data has already been in the offing.

What is big data? So far, there is no universally accepted definition. In Wikipedia, big data is defined as "an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications" [7]. From a macro perspective, big data can be regarded as a bond that subtly connects and integrates the physical world, the human society, and cyberspace. Here the physical world has a reflection in cyberspace, embodied as big data, through Internet, the Internet of Things, and other information technologies, while human society generates its big data-based mapping in cyberspace by means of mechanisms like human – computer interfaces, brain – machine interfaces, and mobile Internet [8]. In this sense, big data can basically be classified into two categories, namely, data from the physical world, which is usually obtained through sensors, scientific experiments and observations (such as biological data, neural data, astronomical data, and remote sensing data), and data from the human society, which is often acquired from such sources or domains as social networks, Internet, health, finance, economics, and transportation.

Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license. It has been used by many big technology companies, such as Amazon, Facebook, Yahoo and IBM. Hadoop [1] is best known for MapReduce and its distributed file system (HDFS). MapReduce idea is mentioned in a Google paper [11], to be simply the task of MapReduce is another processing of divide and concur. Hadoop [8] is aimed at problems that require examination of all the available data. For example, text analysis and image processing generally require that every single record be read, and often interpreted in the context of similar records. Hadoop uses a technique called MapReduce to carry out this exhaustive analysis quickly. HDFS gives the distributed computing storage provides and support. They are the two main subprojects for Hadoop platform. .Hadoop set FIFO algorithm as its default algorithm. According to our research of algorithm for Hadoop, we found that it unable to satisfy the demand of users. We cannot only keep the idea of first come first served. We need to think about the requirement form that has the higher priority, but at the same time we also can keep the fairness to other users. Then we announced K-centroids Clustering algorithm in Big Data-Hadoop Cluster.

## *1.1 Hadoop And Hdfs Overview*

Hadoop Distributed File System (HDFS)[4] is the primary storage system used by Hadoop applications. The Hadoop distributed file system is designed to handle large files (multi-GB) with sequential read/ write operation. Each file is broken into chunks, and stored across multiple data nodes as local OS track of overall file directory structure and the placement of chunks. DataNode reports all its chunks to the NameNode at bootup. Each chunk has a version number which will be increased for all update. Therefore, the NameNode know if any of the chunks of a DataNode is stale those stale chunks will be garbage collected at a later time. To read a file, the client API will calculate the chunk index based on the offset of the file pointer and make a request to the NameNode. The NameNode will reply which DataNodes has a

copy of that chunk. From this points, the client contacts the DataNode directly without going through the NameNode.

To write a file, client API will first contact the NameNode who will designate one of the replica as the primary (by granting it a lease). The response of the NameNode contains who is the primary and who are the secondary replicas. Then the client push its changes to all DataNodes in any order, but this change is stored in a buffer of each DataNode. After changes are buffered at all DataNodes, the client send a "commit" request to the primary, which determines an order to update and then push this order to all other secondaries. After all secondaries complete the commit, the primary will response to the client about the success. All changes of chunk distribution and metadata changes will be written to an operation log file at the NameNode. This log file maintain an order list of operation which is important for the NameNode to recover its view after a crash. The NameNode also maintain its persistent state by regularly check-pointing to a file. In case of the NameNode crash, a new NameNode will take over after restoring the state from the last checkpoint file and replay the operation log.

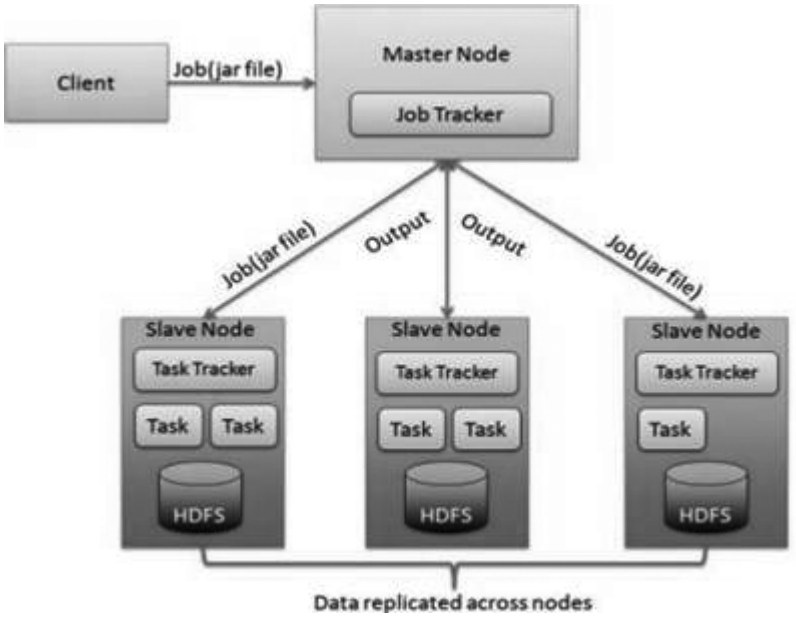


Fig.1

## *1.2 Mapreduce Overview*

The MapReduce frame work [3] consists of a single master JobTracker and one slave TaskTracker per cluster node. The master is responsible for scheduling the jobs' component tasks in the slaves, monitoring them, and re-executing any failed tasks. The slaves executed the tasks as directed by the master. As mentioned, MapReduce applications are based on a master-slave model [6]. This part describes the various operations that are performed by a generic application to transform input data into output data according to that model. The user defined map and reduce functions [5], the map function processes a key/value pairs and return a list of intermediate key/value pairs

Map(k1,v1)—-list(k2 v2)

The reduce function merges all intermediate values having the same intermediate key:

Reduce (k2, list(v2))---list(v2)

The JobTracker will first determine the number of splits (each split is configurable, ~16-64MB) from the input path, and select some TaskTracker based on their network proximity to the data sources, then the

Job Tracker send the task requests to those selected Task Trackers.
Each TaskTracker will start the map phase processing by extracting the input data from the splits. For each record parsed by the "Input Format", it invokes the user provided "map" function, which emits a number of key/value pair in the memory buffer. A periodic wakeup process will sort the memory buffer into different reducer node by invoke the "combine" function. The key/value pairs are sorted into one of the R local files (suppose there are R reducer nodes).
When the map task completes (all splits are done), the TaskTracker will notify the JobTracker. When all the TaskTrackers are done, the JobTracker will notify the selected TaskTrackers for the reduce phase. Each TaskTracker will read the region files remotely. It sorts the key/value pairs and for each key, it invokes the "reduce" function, which collects the key/aggregated Value into the output file (one per reducer node).

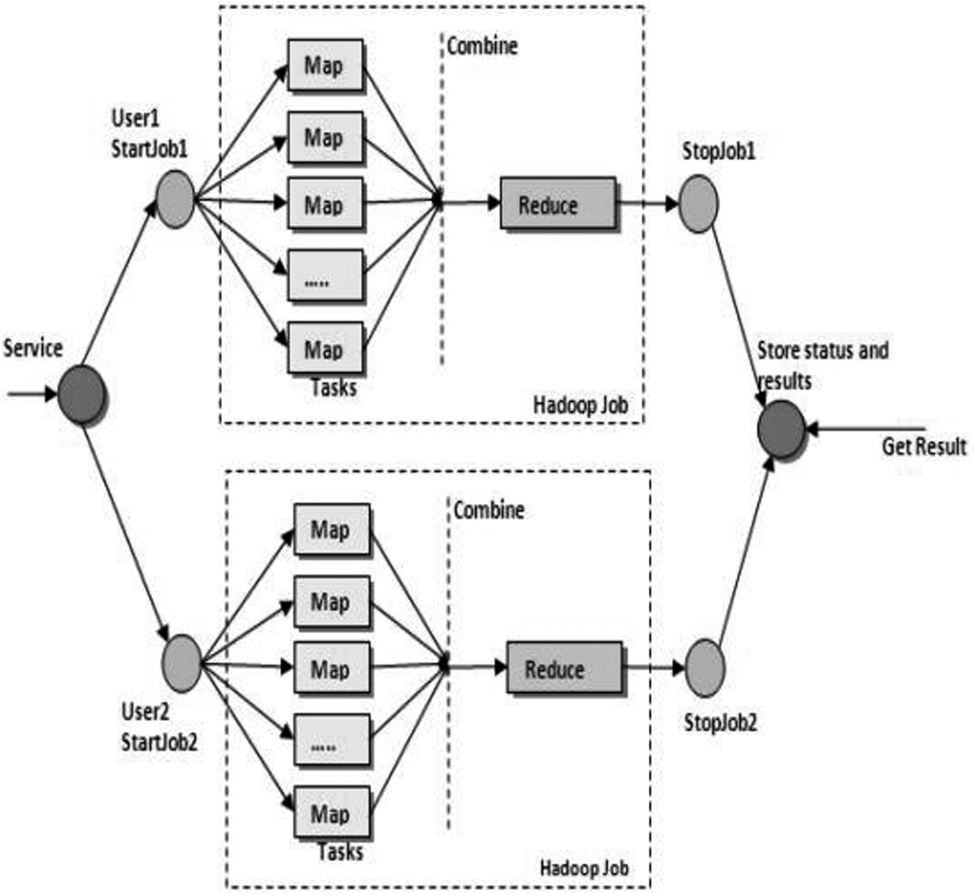


Fig.2

## 2. Related Work

Hadoop's MapReduce operation is initiative requesting from taskTracker to jobtracker .The principle is similar to ordinary, non-preemptive scheduling operating system, which is cannot be interrupt once the task is assigned. As what I have learned about the Hadoop algorithms, there are four classic algorithms.

### *2.1 First In First Out (FIFO):*

This expression describes the principle of a queue processing technique or servicing conflicting demands by ordering process by first come, first served behavior, what comes in first is handled first, what comes in next waits until the first is finished. This is the default algorithm in Hadoop.

### 2.2 Round Robin (RR):

In computer operation, one method of having different program process take turns using the resources of the computer is to limit each process to a certain short time period, then suspending that process to give another process a turn (or "time-slice"). This is often described as round-robin process scheduling.

### 2.3 Height Priority First (HPF):

The algorithm scheduling process, each will be assigned to handle the highest priority ready process. Priority setting for the number of static when it can be dynamic. Static priority number is in the process of creation is based on the initial characteristics of the process or user requirements identified in the process cannot be changed during operation. Dynamic priority number refers to the process and then create an initial priority to determine when the first number, after running in the process as the process characteristics change.

### 2.4 Weighted Round Robin (WRR):

Weighted Round Robin is a scheduling discipline. Each packet flow or connection has its own packet queue in a network interface card. It is the simplest approximation of generalized. While GPS serves infinitesimal amounts of data from each nonempty queue, WRR serves a number of packets for each nonempty queue.

## 3. Disparateness Aware Scheduling Approach In Hadoop Cluster Environment

This section explains the overall flow description of the proposed Disparateness-aware scheduling approach in cluster environment. Initially, the Disparateness cluster environment is created along with the properties of resource such as resource type, processing speed, and the memory. In order to avoid the scheduling delay, the system needs to form a cluster using the K-centroids clustering. Depending up on higher priorities, the node will move to the cluster. Furthermore, the cosine similarity is finding out to compute the clusters. After accomplishing the cluster, the fitness function is estimated with the consideration of load, energy, and time for each cluster. Thus, the clusters are scheduled and then executed after uploading the load. Once any failure occurs during the process, the value must recomputed using PSO and predicted another optimal node. Figure 3 depicts the overall flow diagram of the proposed methodology. The major components of the proposed system are briefly discussed as follows:

Table. 1 Symbols and its descriptions

| SYMBOL | DESCRIPTION |
|---|---|
| $T_{(i,r)}$ | Time required for $i^{th}$ cluster completion in $r^{th}$ cluster resource |
| $L_{(i,r)}$ | Load of $r^{th}$ cluster resource at $i^{th}$ cluster submission |
| $E_{(i,r)}$ | Energy required for $i^{th}$ cluster in $r^{th}$ cluster resource |
| $IT_r$ | Invoking Time of $r^{th}$ cluster resource |
| $ET_{(i,r)}$ | Executing Time of $i^{th}$ cluster in $r^{th}$ cluster resource |
| $RT_{(i,r)}$ | Retrieving Time of $i^{th}$ cluster in $r^{th}$ cluster resource |
| $CS_i$ | $i^{th}$ Cluster Size |
| $CPS_r$ | Cluster Processing Speed of $r^{th}$ cluster resource |
| $CS_j$ | $j^{th}$ Cluster Size |
| $CRSi_r$ | Size of $r^{th}$ Cluster Resource |
| $IE_r$ | Invoke Energy of $r^{th}$ cluster resource |
| $EE_{(i,r)}$ | Execution Energy of $i^{th}$ cluster in $r^{th}$ cluster resource |
| $CPE_r$ | $r^{th}$ Cluster Resource Processing Energy |

### *3.1 K-Centroids Clustering*

The well-known clustering problem can be solved by K-means, which is one of the simplest unsupervised learning algorithms. Assume the K number of clusters for classifying a given cluster processor in a simple and easy way. K-means clustering does not have a guarantee for optimal solution as the performance is based on an initial centroids. Thus, the proposed system uses the partitioning clustering, say, K-centroids clustering as described in the following algorithm. Table I shows the notation and its description that employed in the proposed system

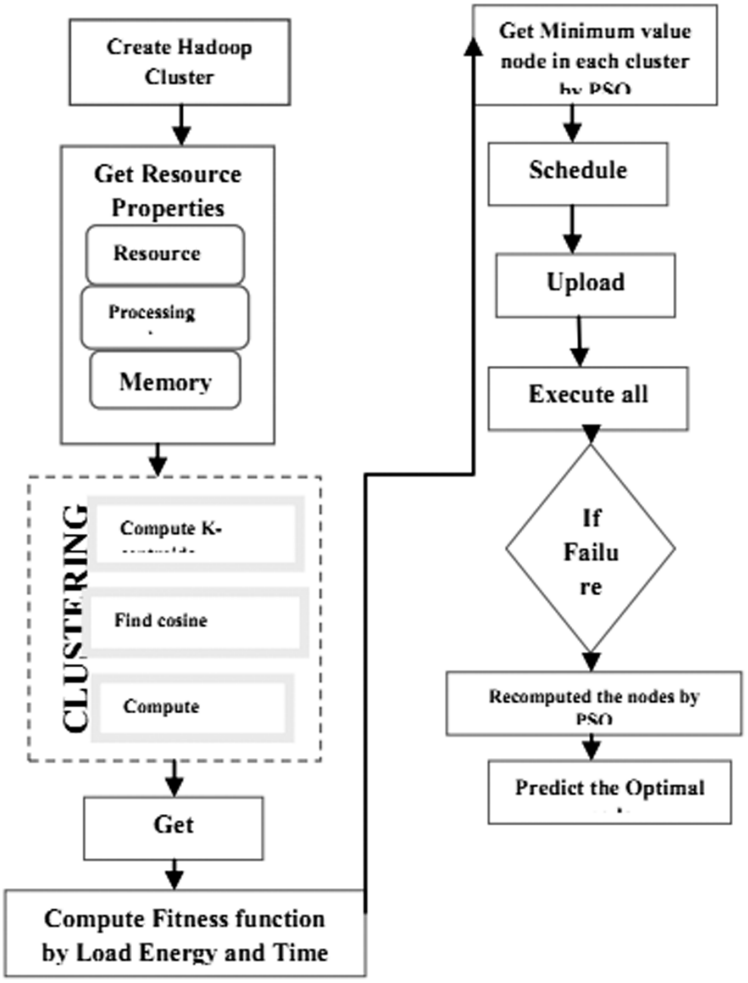


Fig. 3 An overall flow diagram of the proposed K-centroids clustering In Hadoop Cluster.

Algorithm 1: K-Centroids Clustering
Input: Cluster Processors $CP_n$, k value
Begin:
Initialize k centroids $C_k$
For i = 0, 1, 2⋯n then // Cluster Resource Property

$V_1 = \{RT_i, CPS_i, CPSi_i\};$
For j = 0, 1, 2…k then // K-Centroid's Property
$V_2 = \{RT_j, CPS_j, CPSi_j\};$
$Sim[i][j] = CS(V_1, V_2) = \frac{\overrightarrow{V_1}.\overrightarrow{V_2}}{|V_1|.|V_2|};$
End For j
Min_Index = 0;
Min = Sim[i][0]
Curr_Index = 0;
While (Curr_Index < k) then
IfSim [i] [Curr_Index] < Min then
Min = Sim [i] [Curr_Index]
Min_Index = Curr_Index;
End If
Curr_Index++;
End While
SetClusterID (Min_Index);
End for i;
End Begin;

The inputs taken for the above K-centroids clustering algorithm are the Cluster processors and the K value. At first, the K number of centroids are initialized and the vectors V1and V2 are defined with respect to the retrieving time, Cluster processing speed, and size of Cluster resource. Depending up on the vectors, the cosine similarity is estimated. Then, the similarity measures verifies for the minimum value of the current index. If the value is said to be less, then it sets as a minimum value. It checks until the current index is less than the k number of centroids. The minimum index is set as the cluster ID. Further, the fitness function is estimated until all jobs are scheduled by the consideration of load, energy, and time for each cluster. It is defined as:

$$f(n) = T_{(i,r)} + L_{(i,r)} + E_{(i,r)} \quad (1)$$

### 3.1.1 Time Computation

The time computation for ith cluster completion in rth Cluster resource is described as:

$$T_{(i,r)} = IT_r + ET_{(i,r)} + RT_{(i,r)} \quad (2)$$

Where, the Executing Time of ith cluster in rth Cluster resource and Retrieving Time of ith cluster in rth Cluster resource are given respectively as in equation (3) and equation (4).

$$ET_{(i,r)} = \frac{CS_i}{CPS_r} \quad (3)$$

$$RT_{(i,r)} = \frac{CS_i}{BW_{sr}} + \delta_{sr} \quad (4)$$

Herein, $BW_{sr}$ is the bandwidth for the scheduler to the receiver and $\delta_{sr}$ mentions the delay that occurs between the scheduler and the receiver for network communication.

### 3.1.2 Load Computation

The load computation of rthCluster resource at ithcluster submission is as follows:

$$L_{(i,r)} = \sum_{j=0}^{m} CS_j \Big/ MAX(CPSi_r) \quad (5)$$

### 3.1.3 Energy Computation

An energy required for ith cluster in rth cluster resource is defined as:

$$E_{(i,r)}=IE_r + EE_{(i,r)} \quad (6)$$

Where, the Execution Energy of ith cluster in rth Cluster resource is given as:

$$EF_{ir} = CS * CPE_r \quad (7)$$

Based on the consideration of time, load, and energy, the proposed estimation of fitness function has the following advantages:

- The scheduling delay can be avoided in the initial stage.
- Minimize the computation cost
- Decrease the execution time

## 3.2 Pso Based Disparateness-Aware Scheduling Model

The heuristic optimization algorithms are widely used for solving a wide variety of NP-complete problems. PSO is considered as the latest evolutionary optimization techniques, which has fewer algorithm parameters than the other approaches.

Algorithm 2: Disparateness-Aware Scheduling using PSO
Input : Cluster List Jn , Cluster Resource CRm
Output : Allocated Cluster List ALn
Begin
For x = 0, 1, 2 ⋯ n then
TGR = Ø ; // Temporary Cluster Resource list
For y= 0, 1, 2 ⋯ m then
If (Jx . Rtype(). Equals( CRy . Rtype() ) then
TCR.add (CRy)
End IF;
End Fory;
SCR = PSO (TCR, Jx ) // Selected Cluster resource
$J_x$ . setclusterResource SCR
ALx ← SGR
End For x;
End Begin;

The inputs taken for this model are cluster list and the cluster resource. Initially, the temporary cluster resource list is empty and the similarity should be further verified. It checks whether the resource type of cluster list is similar to the resource type of cluster resource. If the verification is similar, then the cluster resources are added to the temporary cluster resource list. The technique of PSO is applied for the temporary cluster resource list and the cluster list to accomplish the selected cluster resource. The final output obtained in this scheduling model is the allocated cluster resource along with the selected cluster resource. When a failure occurs during this process, it can be recomputed by PSO and the other optimal node will be predicted. The advantages over the heterogeneous-aware scheduling model are:

- Minimize the number of failures
- Increases the resource utilization

## 4. Performance Analysis

This section compares the performance of the proposed Disparateness-Aware Scheduling (DAS) algorithm with two existing scheduling algorithms: Height Priority First (HPF)[12] and Weighted Round Robin (WRR)[12].The performance metrics used for the analysis are: system reliability, scheduling delay, scheduling length, speed up, energy consumption, and failure ratio with respect to the number of clusters. Due to the dynamic resource availability, the behavior of scheduling algorithms on real cluster platforms is not practical. The simulation is the optimum choice for testing and comparing the scheduling algorithms, where the experiments on real platforms are often non-reproducible. Thus, an extensive simulation environment of cluster system is built as in Table 2.

Table. 2 Simulation parameters

| METRICS | PARAMETER | VALUE |
|---|---|---|
| Computing Node | Number of Cluster Resource | 5 - 50 |
| | Number of Host per Cluster Resource | 5 - 10 |
| | Processing Speed (MIPS) | 1000 - 5000 |
| | Number of Resource Types | 2 - 5 |
| Communication | Bandwidth (MBPS) | 100 - 500 |
| | Latency(ms) | 5 - 10 |
| Clusters | Number of Clusters | 10 - 100 |
| | Size of Clusters (MI) | 10000-50000 |

### *4.1 System Reliability Vs. Number Of Clusters*

The reliability of the system can be calculated by the average reliability of all clusters and is mathematically defined as follows:

$$SR = \frac{\sum_{k=1}^{n} R[E_{A_k}]}{n} \quad (8)$$

Here, $R[E_{AK}]$ is the distribution of the reliability probability of clusters $A_K$ , and n is the number of clusters. Fig.2 describes the system reliability in terms of the number of clusters for the existing HPF and WRR and the proposed DAS model. The system reliability of DAS model is greater than the other two existing approaches, where its value is gradually decreased with respect to the number of clusters.

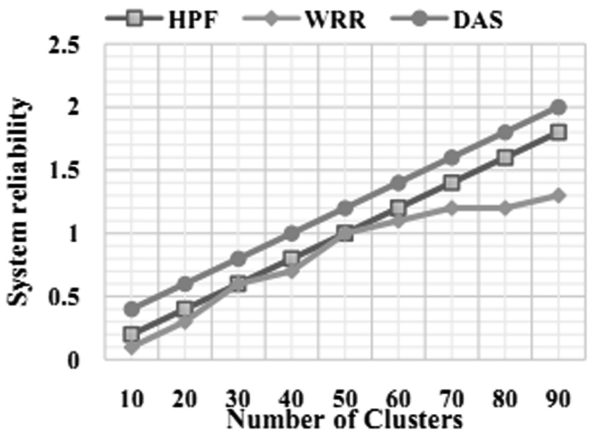


Fig. 4 The result of system reliability with respect to the number of clusters

## *4.2 Scheduling Length Vs. Number Of Clusters*

Schedule length is measured as:

$$SL=\max\{SL(A_1), SL(A_2),\cdots, SL(A_n)\} \quad (9)$$

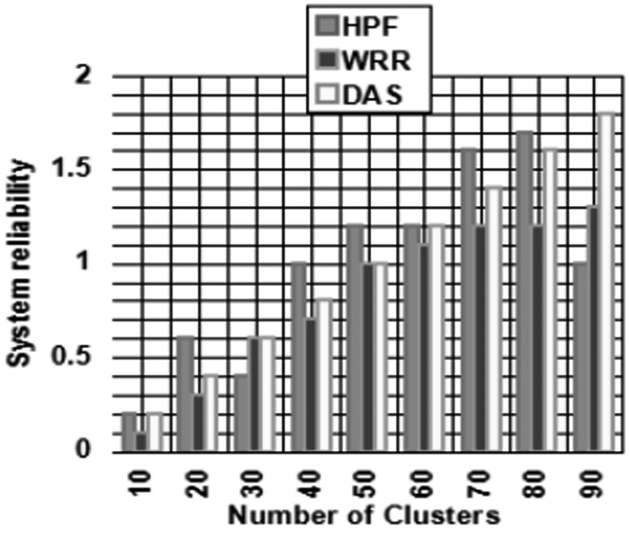


Fig. 5 The result of schedule length with respect to the number of Clusters.

Fig. 5 depicts the result of the schedule length in terms of number of clusters for the existing HPF and WRR and the proposed DAS model. The schedule length is lower than the other two existing systems.

## *4.3 Speed Up Vs. Number Of Clusters*

The speed up is the ratio of the sequential execution time to the schedule length of the output schedule. It is computed as follows:

$$SU = \frac{\sum_{A_k \epsilon A} \sum_{v_n \epsilon A_k} \frac{w(v_n)}{w(p_i)}}{SL} \quad (10)$$

Fig.6 shows the result of speed up with respect to the number of clusters for the existing HPF and WRR and the proposed DAS model. The speed up of HAS model is higher than the other two existing approaches, where its value is gradually increasing with regard to the number of clusters.

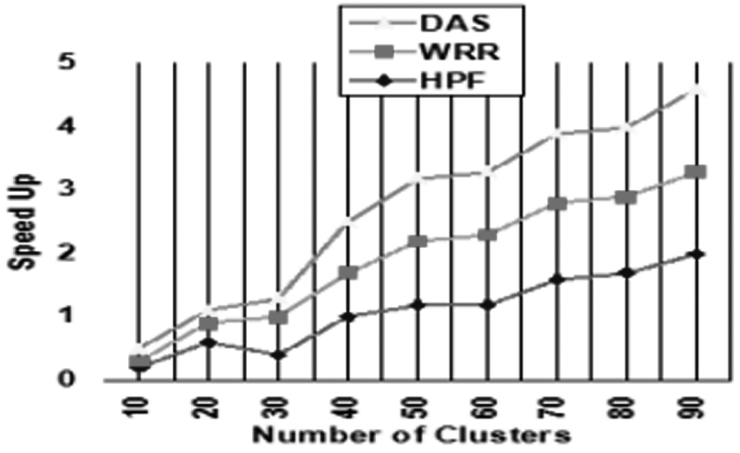


Fig. 6 The result of speed up with respect to the number of clusters.

### 4.4 Scheduling Delay Vs. Number Of Cluster Resource

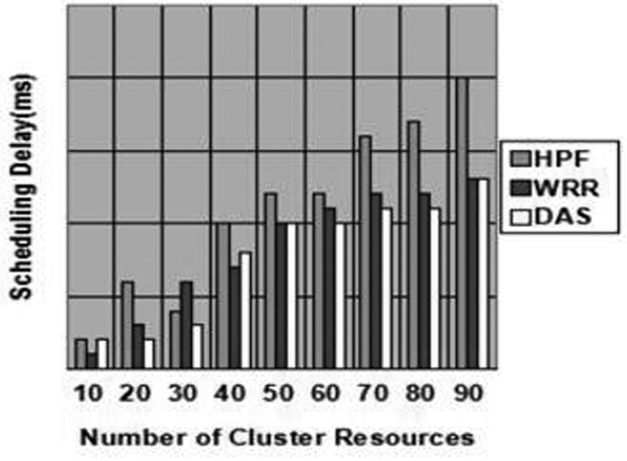


Fig. 7 The result of scheduling delay with respect to the number of cluster resources.

Fig.7 shows the result of scheduling delay with respect to the number cluster resource for the existing HRDS2 and MCMS and the proposed HAS model. The proposed system has a lower scheduling delay than the other two scheduling algorithms.

### 4.5 Energy Consumption Vs. Number Of Cluster Resource

Fig.8 shows the result of energy consumption with respect to the number cluster resource for the existing HPF and WRR and the proposed DAS model. The proposed system consumes less energy than the other two scheduling algorithms. Its value gradually increases in regards to the number of cluster resource.

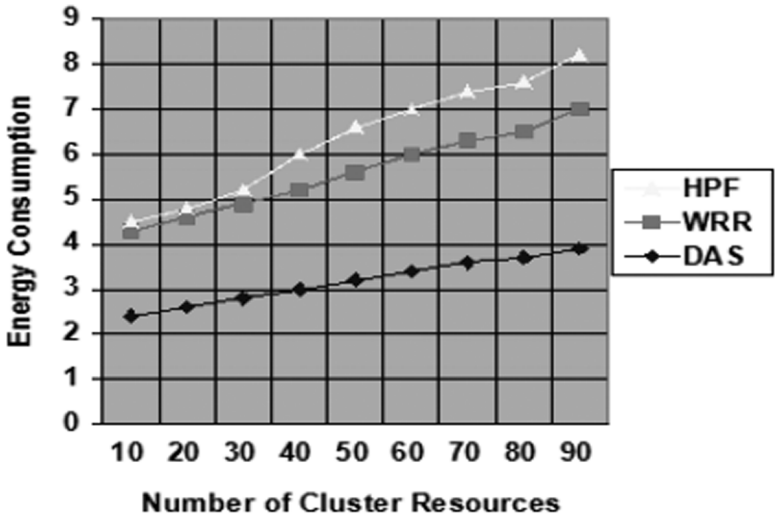


Fig. 8 The result of energy consumption with respect to the number of cluster resource.

### *4.6 Failure Ratio Vs. Number Of Cluster Resource*

The result of the scheduling delay with respect to the number cluster resource for the existing HPF and WRR and the proposed DAS model is shown in Fig.8. The proposed system has a lower failure ratio than the other two existing scheduling algorithms.

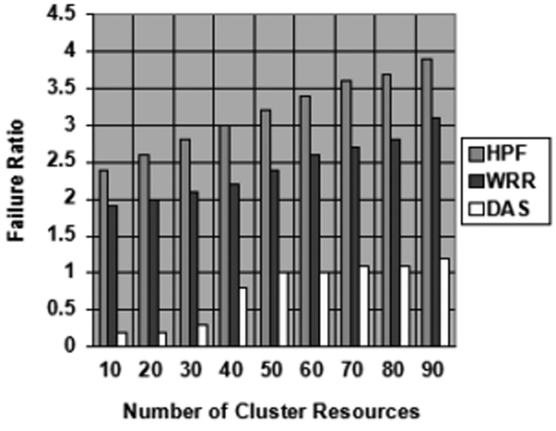


Fig. 9 The result of failure ratio with respect to the number of Cluster resource.

## 5. Conclusion And Future Work

This paper proposes a —Disparateness-Aware Scheduling algorithm in the cluster environment. In this research work we represent K-centroids clustering in big data mechanism for Hadoop cluster. This approach is mainly focused on the energy consumption in the Hadoop cluster, which helps to increase the system reliability. The Hadoop cluster consists of resources which are categorized for minimizing the scheduling delay in the Hadoop cluster using the K-Centroids clustering algorithm. A novel provisioning mechanism is introduced along with the consideration of load, energy, and network time. By integrating these three parameters, the optimized fitness function is employed for Particle Swarm Optimization (PSO) to select the computing node. Failure may

occur after completion of the successful execution in the network. To improve the fault tolerance service, the migration of the cluster is focused on the particular failure node. This can recomputed the node by PSO and the corresponding optimal node is predicted. The experimental results exhibit better scheduling length, scheduling delay, speed up, failure ratio, energy consumption than the existing systems.

## References

[1]V. Mayer-Schonberger, K. Cukier, Big Data: A Revolution That Will Transform How We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.
[2]A. Cuzzocrea, Privacy and security of big data: current challenges and future research perspectives, in: Proceedings of the First International Workshop on Privacy and Securityof Big Data, PSBD '14, 2014.
[3]Big data, Nature 455(7209) (2008) 1 – 136.
[4]Dealing with data, Science 331(6018) (2011) 639 – 806.
[5]C. O'Neil, R. Schutt, Doing Data Science: Straight Talk from the Frontline, O'Reilly Media, Inc., 2013.
[6]Big data, http://en.wikipedia.org/wiki/Big_data, 2014.
[7]G. Li, X. Cheng, Research status and scientific thinking of big data, Bull. Chin. Acad. Sci. 27(6) (2012) 647 – 657.
[8]Y. Wang, X. Jin Xueqi, Network big data: present and future, Chinese J. Comput. 36(6) (2013) 1125 – 1138.
[9]X.-Q. Cheng, X. Jin, Y. Wang, J. Guo, T. Zhang, G. Li, Survey on big data system and analytic technology, J. Softw. 25(9) (2014) 1889 – 1908.
[10] J.Dean,S. Ghemawa. MapReduce: Simplified Data Processing on Large Cluster. OSDI'04, Sixth Symposium on Operating System Design and Implementation, SanFrancisco , CA , December,2004
[11]http://www.vmware.com/appliances/directory/up loaded_files/What%20is%20Hadoop.pdf.
[12] Haiyang Li —PWBRR Algorithm of Hadoop Platform.