Received: March 20, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Fuzzy Resource Pre-processing and Compress and Join Gang Polling Evaluation Scheduling in Cloud Computing

Surendran Sharon Priya[1]* **Wahab Aisha Banu**[2]

[1]*Department of Computer Science and Engineering, B.S. Abdur Rahman University, Chennai, Tamil Nadu, India*
* Corresponding author's Email: sharonpriyaphd@gmail.com

Abstract: Throughout the numerous processing in cloud computing, an essential concern is to scheduling jobs in parallel to cloud data centers. An aim of this paper is to design a Fuzzy logic system. Gang forming with scheduling policy improves the performance of computation cost and time of different cloud workloads. In this work a novel framework for cloud workload management is modelled. In which clustering of cloud workload is done based on Manhattan distance based fuzzy clustering and scheduling of workload is done based on Compress & Join Gang Polling Evaluation scheduling algorithm (C&JGPESA). In order to provide effective utilization of resources, Gang scheduling algorithm is used based on their performance to fit the same number of applications in less time slots. Finally proposed work comparison is compared with computation cost, make span and response time and it achieves around 60%, 97% and 80% greater performance than both existing works.

Keywords: Cloud computing, Dynamic scheduling, Polling evaluation algorithm, Gang scheduling, Manhattan fuzzy clustering, Compress & Join.

1. Introduction

Cloud computing is one of the creative Information System (IS) design envisioned as what might be the outcome of computing. A main motivation is re-arrange the comprehension of working frameworks and client–server models. [1, 2]. Cloud computing is used for accomplishing the task scheduling and it has some particular features like computing power. The job administration capacity into various sub-tasks and it is possible when a distributed computing job arrives [3].

Cloud is an arrangement of empowered administration and it gives adaptable QOS for the systems [4]. In industry these capacities are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) individually [5, 6]. Distributed computing environment is exceptionally troublesome from the scheduling environment [7, 8].

For the most part, enthusiastic information accumulation and pre-processing are acknowledged in the cloudlet, and here acknowledgment and administration are carried out in the remote cloud [9, 10]. To access some pre-handling, for example, image resizing and Gaussian separating to dispense with the noise and improve the image quality. At that point the pre-prepared information are conveying to the remote cloud [11 - 13].

Scheduling of gang is one of the strategy in distributed computing that is depends on time space sharing [14]. In this Gang scheduling, the calculation limit of a node is for sharing jobs. The task scheduling calculation handles every features of a job so one process won't be in rest state when another procedure requires processing with it [15 - 17].

Security issues in distributed computing are difficult in reliable associations [18]. Investigative pre-handling of the information might be conveyed through the portable cloud services for the reasons of protection and security [19]. Scheduling is the practice is to group the tasks of an identical job, [20]. Principle issue in this strategy is number of tasks in a group can't surpass the quantity of accessible assets.

In order to overcome the complexities of the previous work, our proposed method introduces the Manhattan distance based fuzzy clustering. Here scheduling of workload will be done based on Compress & Join Gang League championship scheduling algorithm. This planned technique comprise of three stages.

1. In this first stage resource pre-processing is performed based on Manhattan distance based Fuzzy Clustering (MFC) method.

2. Second stage demonstrates the Gang Polling Evaluation algorithm (GPE), this algorithm evaluates the scheduling priority of the Gangs. Thus based on the priority of the resource gang with the individual task gang and parallel scheduling is carried out in to the ousterhout matrix.

3. Third stage exhibits the Compress & Join Gang Scheduling algorithm, in which task is scheduled parallel to all the virtual machines and hence the number of time slots is also reduced. So the complexity of the process will be reduced. So this process may be used for the system with the non-parallel task scheduling.

The remaining section of this manuscript is given below. Section 2 exhibits the works related to the dynamic task scheduling. Section 3 explains the proposed methodology. Section 4 demonstrates the experimental results and. At last conclusion of this work is explained in the section 6.

2. Related work

Some of the recent related works associated to dynamic scheduling in cloud computing is given below:

Fan Zhang et al [21] proposed a technique which was producing more effective arrangements from a worldwide point of view over quite a while. They demonstrated through overhead investigation and the points of interest in time and space proficiency for utilizing the technique. Because of noise intrusion, the performance of scheduling was degraded. Wei Wang et al [22] outlined a multi-asset assignment instrument, called DRFH that sums up the idea of Dominant Resource Fairness (DRF) from a one server to numerous heterogeneous servers. DRFH gave various much craved properties. Compared to our work responsible time for 100th task is very low. RizwanaIrfan et al [23] suggested Mobi computing, a hybrid cloud-based Bi-Objective Recommendation Framework (BORF) for versatile informal communities. The Mobil Context uses multi target enhancement methods to create customized proposals. To deliver the issues identified with information meagre condition, the BORF accomplish information pre-preparing by using the Hub-Average (HA) induction demonstrate. To measure time complexity of this work takes more steps and since computation cost is high. Sun Yuan Hsieh et al [24] proposed a job scheduler called the job portion scheduler, organized to adjust task usage. The JASL built up the utilization of hubs and the execution of Hadoop in heterogeneous computing situations. In conclusion, two parameters were added to the JASL to distinguish wrong space settings and make a dynamic job assignment scheduler with region (DJASL). This system is not suitable for large amount of extraneous network transformation.

Yongsheng Hao et al [25] arranged standard nonlinear relating jobs in Cloud. It incorporates a particular equality that may exclusively compose at the beginning of the execution. They show the adequacy and intensity of our arranging task through re-enactments using WRF (Weather examination and predicting model) that is wide utilized in logical computing. Our investigation come 100% diminishment at execution time. This technique is not suitable for parallel scheduling

3. Proposed methodology

The issue identified with the multi resource task scheduling executes task at once. This procedure is not satisfactory for synchronizing dynamic scheduling. A substitute way to deal with scheduling is to execute multiple tasks at a unit time. Despite the fact that it will execute one job at once, our proposed strategies give scheduling of parallel task to all the resources.

Volume of integrities building as polling gangs participates in an artificial league for more than a few weeks (iterations). As per the outcome, overhead commenced through the context switch among slots must be proficiently condensed.

The main objective of this method is to schedule the task to all the resource with dynamical manner. In order to schedule task in parallel, the planned methodology introduces Compress & Join Gang Polling Evaluation scheduling algorithm

(C&JGPESA) in which resources are grouped founded on Manhattan based Fuzzy clustering (MFC). The Gang Polling Evaluation (GPE) algorithm is a novel algorithm premeditated depends on the polling gangs. GPE can be emblematically enlightened as follows:

Received: March 20, 2017

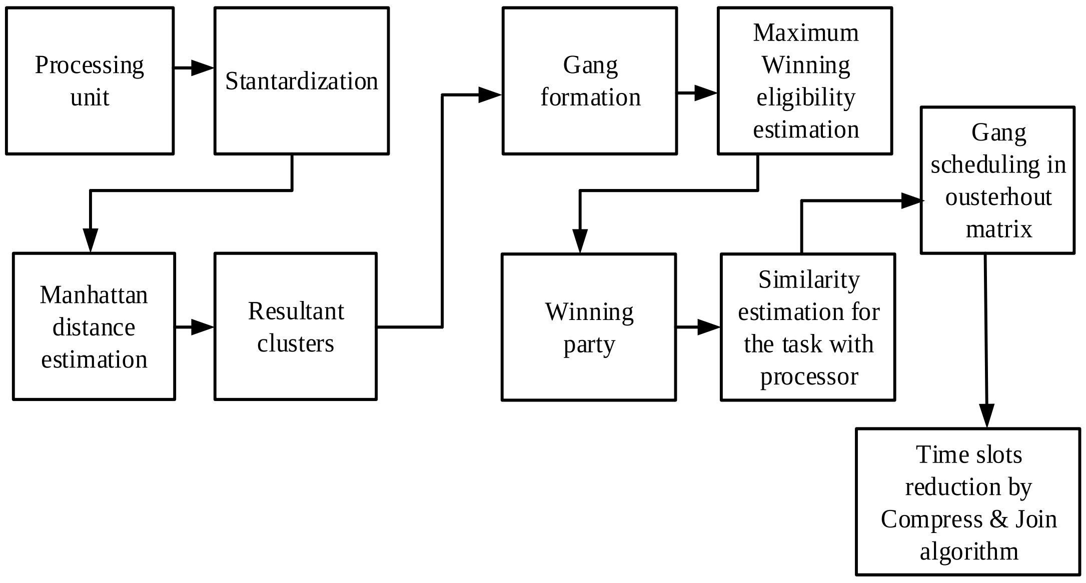

Processing unit
Stantardization
Manhattan distance estimation
Resultant clusters
Gang formation
Maximum Winning eligibility estimation
Winning party
Similarity estimation for the task with processor
Gang scheduling in ousterhout matrix
Time slots reduction by Compress & Join algorithm

Figure.1 schematic representation of proposed pre-processing and Gang Polling Evaluation scheduling

Figure 1 shows the schematic representation of proposed method in which pre-processing epitomizes the clustering process based on Manhattan distance. Gang Polling Evaluation algorithm sufficiently identifies the gang with higher priority and the scheduling is initially processed for gang (party) with most precedence. So the task is allocated in the Ousterhout matrix in alternative with the priority scheduling. Objective function of this work is given by the below equation:

$$(E_{sort} => R_{Priority}) \epsilon \xi_{matrix} \quad (1)$$

Equation (1) describes the scheduling capacity of the Ousterhout matrix. In which sorted task *(E_{sort})* will be scheduled to resource with higher priority *($R_{priority}$)*. In this it should be lies between dimensions of the Ousterhout matrix *(ξ_{matrix})*.

The design objective of this paper is to minimize makespan.

- Scheduling resources to all the virtual machines in parallel manner and also reducing number of time slots.
- To attain minimum computation cost and time.

3.1 Resource pre-processing using MFC

In order to achieve the enhanced resource scheduling enactment in cloud computing, this paper presents a novel Manhattan distance based Fuzzy Clustering (MDFC). Sorting out all resources into a number of clusters this is done by the resources with the related computing ability is assembled into one cluster.

Initially the mathematical modelling starts with the resource pre-processing. The processing units are clustered based on the minimum Manhattan distance. The ith characteristic value of kth process unit is denoted by g_{ki} and $c_k$in set *C= {C_1, C_2, C_k}* has pattern vector *C'= {C_{k1}, C_{k2}... C_{ni}}* then the ith value should be lies between *($1 \leq i \leq m$)*.

The clustering of target cloud systems is based on their properties and it can be improve by resources scheduling performance. By using qualitative characteristics, the corresponding properties or qualities of cloud computing.

By using mean and standard deviation with the data *C* in target system only we can achieve standardization data *C'*. The standardized value of C_{ik} is given by Eq. (2).

$$C'_{ik} = \frac{(C_{ik} - \mu_{ik})}{\sigma_{ik}} \quad (2)$$

$$\mu_{ik} = \frac{1}{m} \sum_{i=1}^{m} C_{ik} \quad (3)$$

Where μ_{ik} denotes the k^{th} eigenvector of original data and μ_{ik} is the mean value of cik then cik is the kth eigenvector of original data.

$$\sigma_{ik} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (C_{ik} - \mu_{ik})^2} \quad (4)$$

σ_{ik} is the standard deviation of C_{ik}. Normalization factor depends on the extreme standardized method. Since the resulting standardized data C_{ik}' is not yet necessarily in [0, 1]. The extreme standardized procedure is given by Eq. (5) and U_{kmin}' is the minimum value in $U_{1k}', U_{2k}', U_{Nk}'$ and $U_{kmax'}$ is the maximum value in $U_{1k}', U_{2k}', U_{Nk}'$.

$$U_{ik}'' = \frac{(C_{ik}' - C_{kmin}')}{(C_{kmax}' - C_{kmin}')} \quad (5)$$

Wherever, C_{kmin}' is the smallest rate in $C_{1k}', C_{2k}' \ldots C_{mk}'$ and $C_{kmax'}$ is the greatest value in $C_{1k}', C_{2k}' \ldots C_{mk}'$. This standardization is for make the data as fuzzifier output.

The Manhattan distance can be calculated by the sum of difference among the two items. So Eq. (6) represents the distance formula of centre point and standardized data.

$$D = (min\{d_t\}) \quad (6)$$

$$d_t = \sum_{i=1}^{m} F_{ij} \, \|C_{ik}' - P_t\| \quad (7)$$

$$F_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - P_t\|}{\|x_i - P_k\|} \right)^{\frac{2}{l-1}}} \quad (8)$$

Where $P_t = \sum_{i=1}^{m} \frac{F_{ij} \cdot C_{ik}''}{m}$ (9)

In the fuzzification (F_{ij}) formula p_t represents the middle point of the characteristic value and l represents the fuzziness index then its value varies from 0 to ∞. It converts the data with in zero to one limit.

Next to the gang formation we have to select which gang will execute the task. After forming the Gang we've got to make your mind up that gang can execute the task. Therefore we've got to separate characteristics of every gang and kind it within the declivitous order. Based upon the arrangement we are able to decide that which gang will execute the task.

3.2 The compress & join gang polling evaluation scheduling algorithm (C&JGPESA)

To achieve dynamic task scheduling in cloud computing, the susceptible (C&JGPESA) is introduced and hence overhead introduced by the context switch between the slots is actively reduced. Polling Evaluation based scheduling algorithm mainly used for find out the winning team among the number of teams. At first, resource size and the number of gang will be initialized. Then the winning eligibility will be calculated for the number of gang. Based on the winning eligibility, the winning team will be calculated for every task with the related resources. From that only, the task will be scheduled to the identical resources.

Job organisation: Ousterhout matrix

In order to make an effective scheduling, gang formation is performed on the task in the cloud computing. The number of task assigned in this process is, $T = \{t_1, t2 \ldots t_k\}$ and it has the pattern vector of $T' = \{t_{k1}, t_{k2} \ldots t_{ki}\}$. The value of i should be within the range of [1, m].

In our process, the number of task will be scheduled on an Ousterhout matrix. So the task will be scheduled on parallel manner to the corresponding resources. Gang Scheduling oversees a two dimensional matrix wherever individual aspect embodies the processors and the other is time (this arrangement remains recognized as the Ousterhout matrix). Among the resources and the slots time sharing is performed. So the number of time slots must be equal or lesser than the amount of processors. Every single feature is a time slot, poised through a gradient of more than numerous teams.

Gang Polling Evaluation

Gang Polling Evaluation algorithm is a priority estimation algorithm in which the party with majority vote is selected as the winning party. In this process, 'gang' is mentioned as the 'party' and fitness is mentioned as the 'winning eligibility'. For the number of task winning eligibility is calculated to estimate the priority of the party.

Figure 2 deliberate the process flow of GPE algorithm in which scheduling of the task to the corresponding resource is carried out based on winning eligibility of the task in accordance with the resource. Then the compress & Join algorithm reduces the number of time slots by find out the efficiency of the each task.

Let us consider initial party assignment and the party size for evaluating gang from the random number of task. Gang formation on the task is performed by Eq. (10).

$$t \geq \sqrt{A_{ik} \times (L-1)} \quad (10)$$

In this equation t represents the number of party and A_{ik} represents the ith value of kth attribute.

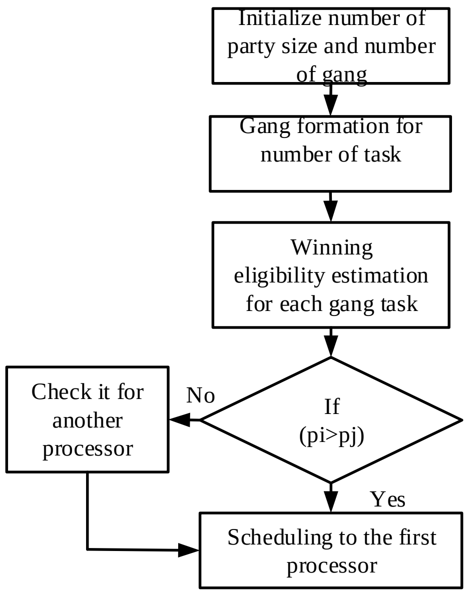

Initialize number of party size and number of gang
Gang formation for number of task
Winning eligibility estimation for each gang task
If (pi>pj)
No
Yes
Check it for another processor
Scheduling to the first processor

Figure.2 Process flow of Gang Polling Evaluation Scheduling algorithm

The condition which is satisfied Eq. (10) means that form as one gang and the else it form as another one gang. Since the complexity of the algorithm can be reduced by forming gang. In case of Polling Evaluation scheduling algorithm, each team should have equal number of task. Hence, the sorting operation is performed for the task gang to make the task in equal number.

This process makes the task gang as equal with precedential manner. After the Gang formation the next step is to scheduling the task to the corresponding resource. Winning eligibility calculation is given in the below Eq. (11). For that the winning eligibility will be calculated for determining the similarity among the number of resource and the number of task.

$$\frac{(E_i)-(v)}{(E_j)-(v)} = \frac{P_j}{P_i} \quad (11)$$

$$P_i + P_j = 1 \quad (12)$$

From Eq. (11)

$$\frac{(E_i)-(v)}{(E_j)-(v)} = \frac{1(1-P_i)}{P_i} \quad (13)$$

$$\frac{(E_i)-(v)}{(E_j)-(v)} + 1 = \frac{1}{P_i} \quad (14)$$

$$P_i = \frac{(E_j)-f(v)}{(E_j)+(E_i)-f(v)} \quad (15)$$

Equation (15) comprises the fitness evaluation of various tasks with the interrelated resources. Where *(E_j)* and *(E_i)* are the average value of the task attributes.

$$(E_i) = (E_j) = Execution\ time\ of\ the\ task \quad (16)$$

$$(v) = \frac{t_3}{n} \quad (17)$$

$$(E_{sort} \Rightarrow R_{priority}) \in \xi_{matrix} \quad (18)$$

Fitness evaluation in this Eq. (15) deliberates the number of task present in the nth gang. In the same way *f (v)* is the fitness evaluation of the number of resources. Equation (18) describes the scheduling capacity of the Ousterhout matrix. In which sorted task *(E_{sort})* will be scheduled to resource with higher priority *($R_{priority}$)*. In this it should be lies between dimensions of the Ousterhout matrix *(ξ_{matrix})*. So the similarity is estimated based on the above equation. Then the job will be allocated to the ousterhout matrix depend on the winning eligibility priority.

At last, compress & Join algorithm compresses the number of time slots by efficiency estimation. Hence the time slot finally reduces the number of time slots, complexity and the time consumption of the process will be reduced.

Similarity estimation

Case 1:

For equal number of tasks in the gang, the similarity estimation is carried out to the cluster of resources. So there is there is no need for individual resource priority estimation. The major step in the Gang Polling Evaluation algorithm is to estimate the priority for each task which is given in Fig. 3. In which every task in the gang is compared its characteristics with the processor characteristics in order to evaluate the priority of the task. So, higher priority task is given to the corresponding resource. Hence this process is repeated for the number of processor for estimating priority for the tasks.

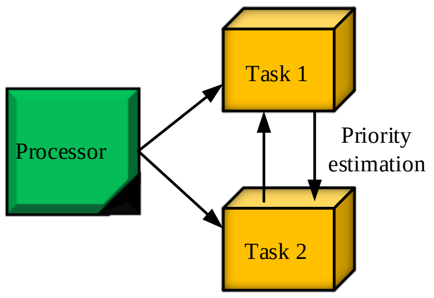


Figure.3 Schematic representation of similarity estimation of the task with resource gang

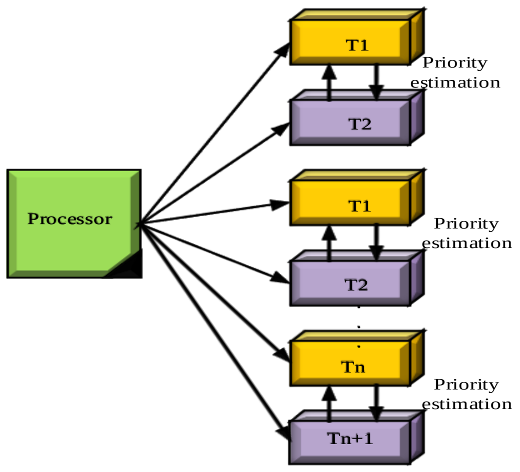


Figure.4 Schematic representation of similarity estimation of the task with individual resource

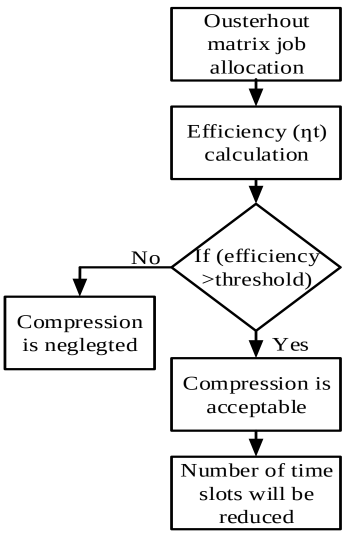
Case 2:

In this type of analysis the priority estimation is processed for the individual resources which are given in figure 4. Since intended for each resource, the priority is computed in the basis of every task in the ascending order manner.

An individual resource with task priority estimation may incorporate with the dynamical scheduling. Consequently, avoiding complexity of the parallel scheduling process, Compress & Join gang scheduling is used. Thus the number of time slots and the will reduce.

Compress & Join Gang scheduling algorithm

The major goal of Compress& Join algorithm is to fill the free space in the ousterhout matrix and reducing time slots.



Ousterhout matrix job allocation
Efficiency (ηt) calculation
If (efficiency >threshold)
No
Yes
Compression is neglegted
Compression is acceptable
Number of time slots will be reduced

Figure.5 Flow chart for Compress & Join algorithm

The above Fig. 5 shows the flow chart for the Compress and Join algorithm. Initial process of the compress & Join algorithm is to clear the complete ousterhout matrix. By reducing the number of time slots, an amount of delay will be reduced. Predominantly, the time slots allotment is set to zero. Then the job will be compressed based on the efficiency, if the efficiency should be greater than the threshold. If the efficiency is less than the threshold means the compression is not acceptable.

$$Efficiency(\eta_t) = \frac{T_{sp}\times 100}{m} \tag{19}$$

$$T_{sp} = \frac{Execution\ time\ of\ the\ task}{Avg\ commn\ ability\ of\ the\ processor} \tag{20}$$

In Eq. (19) speed up time *(T_{sp})* is calculated for n number of task with m number of processors. Since for each processors efficiency is compared with the threshold to ensure the compression with in the time slots.

4. Experimental results and discussions

The performance analysis is compared with the metrics such as makespan, idle time, energy consumption and average computation cost. The simulation for the projected technique is process on JAVA. The simulation results illustrate that our proposed method provides satisfied performance than the existing method.

In this work, the data set consist of (i) Arrival time: It is the time to arrive the task for further processing, (ii) Execution time: It is the time when the execution of the task is to be complete, (iii) Memory size: It is the memory size of the message that is to be displayed. So totally m number of task is to be scheduled on to the n number of processor which is shown in the below figure.

Case 1:

In this case the tasks are dynamically scheduled based on gang similarity. Since, it has an equal number of tasks in each gang. So parallel scheduling is possible in this case. Based on the pre-processing two kinds of resource clusters are formed. They are (P0, P2, P6, P7, P9, P10, P11 and P12) and (P1, P3, P4, P5 and P8). So the tasks are scheduled separately on those gang depend on the priority estimation criteria. Table 1 represents the task scheduling model to the C1 Ousterhout matrix and table 2 demonstrate the task scheduling model to the C2 Ousterhout matrix.

Result for Polling Evaluation algorithm:

Table 1. Task scheduling in to the C1Ousterhout matrix

P0	P2	P6	P7	P9	P10	P11	P12
T3	T11	T12	T7	T18	T16	T5	T8
T19	T14	T3	T11	T12	T7	T18	T16
T5	T8	T19	T14	T3	T11	T12	T7
T18	T16	T5	T8	T19	T14	T3	T11
T12	T7	T18	T16	T5	T8	T19	T14
T3	T11	T12	T7	T18	T16	T5	T8
T19	T14	T3	T11	T12	T7	T18	T16
T5	T8	T19	T14	T3	T11	T12	T7

Table 2. Task scheduling in to the C2 Ousterhout matrix

P1	P3	P4	P5	P8
T1	T1	T1	T1	T1
T2	T2	T2	T2	T2
T4	T4	T4	T4	T4
T6	T6	T6	T6	T6
T9	T9	T9	T9	T9

Table 3. Compress & Join Gang scheduling

P0	P2	P6	P7	P9	P10	P11	P12
T2	T16	T19	T11	T12	T14	T12	T16
T18	T7	T5	T14	T3	T8	T3	T7
T12	T11	T18	T8	T12	T16	T19	T11
T3	T14	T12	T16	T3	T7	T5	T14
T11	T8	T3	T7	T16	T11	T18	T8

Table 4. Characteristics for the simulation

Characteristics	value
Number of processors	13
Number of tasks	20
Processor frequency	200HZ
Cycles per instruction	0.9997105
Message size	16 Kbytes

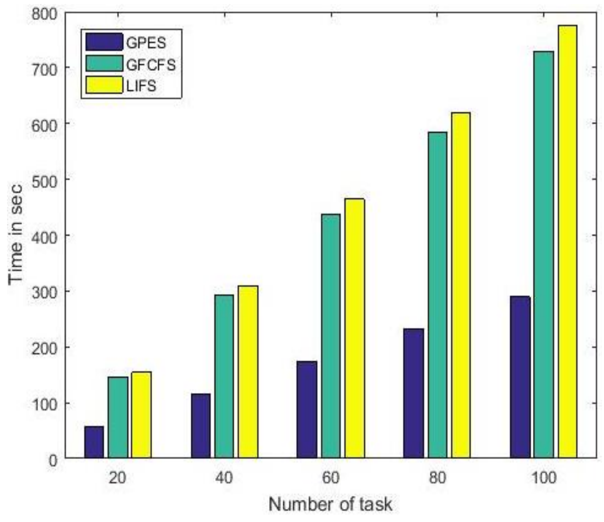


Figure.6 Make span comparison analysis

Case 2:

In case of non-parallel scheduling Compress & Join algorithm efficiently reduces the time slots and hence the gap between the slots also meritoriously compressed. Since for non-similar task gang this process will be effectively utilized to produce 100% utilization.

Result after Compress & Join:

Table 3 deliberates the compress & Join output. From the result the time slot would compressed and the complexity will be reduced. In case of C2 Ousterhout matrix there is no need for the compress & Join algorithm. Since the speed up time and efficiency of the C2 based processor is low.

4.1 Simulation environment

The simulation is performed on the platform of JAVA and the below table shows the parameter needed for the simulation.

4.2 Performance evaluation

Performance of the proposed Gang Polling Evaluation algorithm (GPES) is compared with the existing methods. This evaluation is carried out to show the performance enhancement of the proposed method.

Makespan analysis

It is the maximum completion time of the recent job.

$$makespan = \max_{j \in m} completion\ (i) \qquad (21)$$

Completion (i) designates the interval while the resource concluded the assigned job.

The result demonstrates that the GPES algorithm is 60.27% better than the GFCFS (Gang First Come First Serve) and 62.58% better than LJFS (Largest Job First Served) technique. From the figure for varying number of task, the proposed method yields better performance than existing algorithms.

Response time

Response time is demarcated as the dissimilarity between task execution time t_i^f and the task arrival time t_i^a.

$$t_i^{\,y} = t_i^{\,f} - t_i^{\,a} \qquad (22)$$

If the number of task will vary means the response time comparison of the proposed method contributes enhanced enactment.

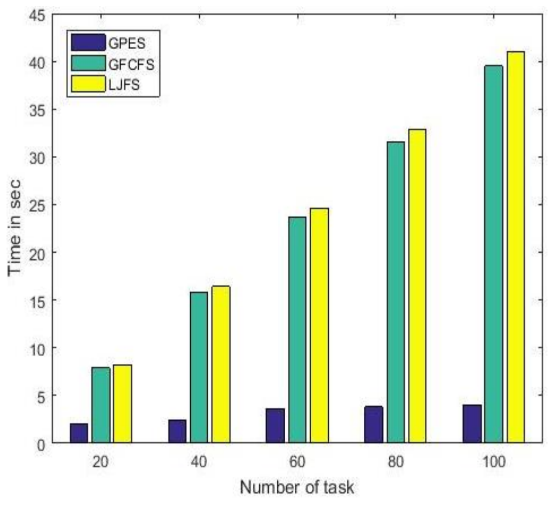


Figure.7 Responsibility comparison analysis

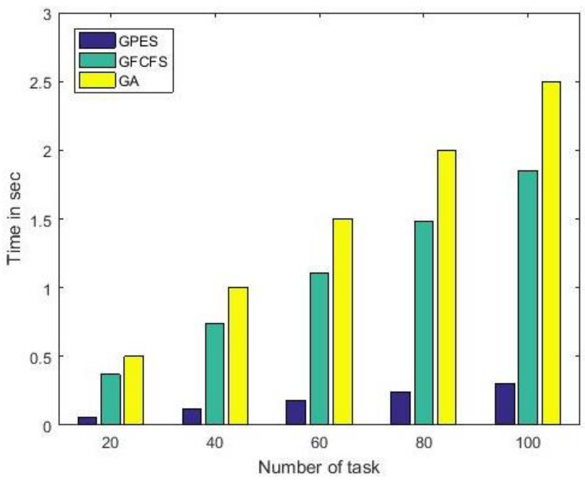


Figure.8 Computation cost analysis

The above figure deliberates the proposed method is 97.4% better than the LJFS (Largest Job First Served) and 97.5% better than the GFCFS (Gang First Come First Serve) technique.

Average computation cost analysis

It is the ratio of predictable execution time to accomplish the task t_i on resource p_j to the amount of resource.

$$Computation\ cost = \sum_{j=1}^{q} \frac{e_t}{q} \tag{23}$$

e_t designates the predictable execution time to fulfilled task t_i on processor p_j and q is the number of processors.

Average computation cost of the suggested technique is 83.65% better than GFCFS (Gang First Come First Serve) and 87.95% better than Genetic Algorithm (GA).

5. Conclusion

Efficient scheduling algorithm in cloud computing system works vigorous role for sharing the resources. For that a scheduling optimization tactic centred on Manhattan distance based fuzzy clustering and the scheduling of workload will be done based on Polling Evaluation scheduling algorithm. In which we dynamically distribute resources to satisfy the necessities of corresponding virtual machines. In this paper, initially resource pre-processing is processed based on Manhattan fuzzy clustering (MFC) for gang forming and the Compress & Join Gang Polling Evaluation scheduling algorithm (C&JGPESA) efficiently find out the priority of the gang and hence dynamic task scheduling is performed based on the priority scheduling strategy for discovering most priority task to allocate in to the Ousterhout matrix. The experimental result shows 60%, 97% and 80% better result than the previous techniques in terms of make span, response time and average computation cost analysis. Thus effective utilization of resources with the parallel task scheduling is done within the same number of applications in less time slots.

For advance studies, the preemptive Virtual machine scheduler working through self-sufficient and heterogeneous tasks on Cloud computing will be focused. Future studies also focused on minimum energy consumption systems on cloud computing.

References

[1] D. Zissis, and D. Lekkas, “Addressing cloud computing security issues”, *Future Generation computer systems*, Vol.28, No.3, pp.583-592, 2012.

[2] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, “Cloud computing-The business perspective”, *Decision support systems*, Vol.51, No.1, pp.176-189, 2011.

[3] J. Ma, W. Li, T. Fu, L.Yan, and G. Hu, “A Novel Dynamic Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing”, In: *Wireless Communications, Networking and Applications*, Vol.1, No.1, pp.829-835, 2016.

[4] L. Wang, G. V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, “Cloud computing: a perspective study”, *New Generation Computing*, Vol.28, No.2, pp.137-146, 2010.

[5] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”, *Future generation computer systems*, Vol.28, No.5, pp.755-768, 2012.

[6] Q.Wang, C. Wang, K. Ren, W. Lou, and J. Li, “Enabling public auditability and data dynamics for storage security in cloud computing”, *IEEE transactions on parallel and distributed systems*, Vol.22, No.5, pp.847-859, 2011.

[7] A. losup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T.Fahringer, and D.Epema, “Performance analysis of cloud computing services for many-tasks scientific computing“, *IEEE Transactions on Parallel and Distributed systems,* Vol.22, No.6, pp.931-945, 2011.

[8] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, “Green cloud computing: Balancing energy in processing, storage, and transport“, *Proc. of the IEEE,* Vol.99, No.1, pp.149-167, 2011.

[9] M. Husain, J. MCGlothlin, M. M. Masud, L. Khan, and B. M. Thuraisingham, “Heuristics-based query processing for large RDF graphs using cloud computing”, *IEEE Transactions on Knowledge and Data Engineering,* Vol.23, No.9, pp.1312-1327, 2011.

[10] M. Chen, Y. Zhang, Y.Li, S. Mao, and V. C. M. Leung, “EMC: emotion-aware mobile cloud computing in 5G”, *IEEE Network,* Vol.29, No.2, pp.32-38, 2015.

[11] M. Chen, Y. Zhang, Y. Li, M. M. Hassan, and A. Alamri, “AIWAC: affective interaction through wearable computing and cloud technology”, *IEEE Wireless Communications,* Vol.22, No.1, pp.20-27, 2015.

[12] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for secure cloud storage”, *IEEE Transactions on computers,* Vol.62, No.2, pp.362-375, 2013.

[13] B-R. Huang, C.H. Lin, and C-H. Lee, “Mobile augmented reality based on cloud computing”, *IEEE, Anti-counterfeiting, Security, and Identification,* Vol.1, No.1, pp.1-5, 2012.

[14] Z. C. Papazachos, and H. D. Karatza, “The impact of task service time variability on gang scheduling performance in a two-cluster system”, *Simulation Modelling Practice and Theory,* Vol.17, No.7, pp.1276-1289, 2009.

[15] X. Liu, C.Wang, B.B. Zhou, J. Chen, T. Yang, and A. Y. Zomaya, “Priority-based consolidation of parallel workloads in the cloud”, *IEEE Transactions on Parallel and Distributed Systems,* Vol.24, No.9, pp.1874-1883, 2013.

[16] Y. Chen, and H. Hu, “Internet of intelligent things and robot as a service”, *Simulation Modelling Practice and Theory,* Vol.34, No.1, pp.159-171, 2013.

[17] L. Grandinetti, O. Pisacane, and M. Sheikhalishahi, ”An approximate ϵ-constraint method for a multi-objective job scheduling in the cloud”, *Future Generation Computer Systems*, Vol.29, No.8, pp.1901-1908, 2013.

[18] N. Kshetri, “Privacy and security issues in cloud computing: The role of institutions and institutional evolution”, *Elsevier, Telecommunications Policy*, Vol.37, No.4, pp.372-386, 2013.

[19] P. Sharma, S. K. Sood, and S. Kaur, “Security issues in cloud computing”, *High Performance Architecture and Grid Computing*, Vol.1, No.1, pp.36-45, 2011.

[20] M. Gerla, “Vehicular cloud computing”, *Ad Hoc Networking Workshop (Med-Hoc-Net), IEEE,The 11th Annual Mediterranean*, Vol.1, No.1, pp. 152-155, 2012.

[21] F. Zhang, J. Cao, K. Hwang, K. Li, and S. U. Khan, “Adaptive Workflow Scheduling on Cloud Computing Platforms with Iterative Ordinal Optimization”, *IEEE Transactions on Cloud Computing*, Vol.3, No.2, pp.156-168, 2015.

[22] W. Wang, B. Liang, and B. Li, “Multi-resource fair allocation in heterogeneous cloud computing systems”, *IEEE Transactions on Parallel and Distributed Systems*, Vol.26, No.10, pp.2822-2835, 2015.

[23] R. Irfan, O. Khalid, M.U. Khan, C. Chira, R. Ranjan, F. Zhang, S. Khan, B. Veeravalli, K. Li, and A. Zomaya, “MobiContext: A Context-aware Cloud-based Recommendation Framework”, *IEEE Transactions on Cloud Computing*, Vol.99, No.1, pp.1, 2015.

[24] S. Y. Hsieh, C. T. Chen, C. H. Chen, T. H. Yen, H. C. Hsiao, and R. Buyya, “Novel Scheduling Algorithms for Efficient Deployment of Map Reduce Applications in Heterogeneous Computing Environments”, *IEEE Transactions on Cloud Computing,* Vol.1, No,1, pp.1, 2016.

[25] Y.Hao, L.Wang, and M.Zheng, “An adaptive algorithm for scheduling parallel jobs in meteorological Cloud”, *Elsevier, Knowledge-Based Systems*, Vol.1, No.98, pp.226-240, 2016.

[26] V. Kumar, C. P. Katti, and P. C. Saxena, “A Novel Task Scheduling Algorithm for Heterogeneous Computing”, *International Journal of Computer Applications.* Vol.85, No.18, 2014 Jan 1.

[27] B. Kruekaew and W. Kimpan, “Virtual machine scheduling management on cloud computing using artificial bee colony”, In:

Proceedings of the International Multi Conference of Engineers and Computer Scientists, Vol.1, No.1, pp.12-14, 2014.