IT in Industry, Vol. 9, No.2, 2021

PublishedOnline12April 2021

AN ADAPTIVE RESOURCE SCHEDULING FRAMEWORK WITH SCHEDULE INTERVAL FILLING FOR CLOUD SERVICES

[1] B.SivaRama Krishna, **[2]Dr. E. Srinivasa Reddy,**

[1]Research Scholar, [2]Professor&Dean,

[1,2]Dept. of Computer Science and Engineering,

[1,2]ANU College of Engineering and Technology

[1,2]Acharya Nagarjuna University,Nagarjuna Nagar – 522510

Guntur, Andhra Pradesh, India.

Abstract

Application level resource scheduling in distributed cloud computing is a significant research objective that grabbed the attention of many researchers in recent literature. Minimal resource scheduling failures, robust task completion and fair resource usage are the critical factors of the resource scheduling strategies. Hence, this manuscript proposed a scalable resource-scheduling model for distributed cloud computing environments that aimed to achieve the scheduling metrics. The proposed model called " Modified Resource Scheduling with Schedule Interval Filling " schedules the resource to respective task such that the optimal utilization of resource idle time achieved. The proposed model performs the scheduling in hierarchical order and they are optimal idle resource allocation, if no individual resource is found to allocate then it allocates optimal multiple idle resources with considerable schedule intervals filling. The experimental results evincing that the proposed model is scalable and robust under the adapted metrics.

1. Introduction

The Modified Resource Scheduling with Schedule Interval Filling (MRS-OSIF) is proposed in this manuscript functions as frontend to Resource Allocation Controller. Initially, the set of similar tasks triggered are pooled as a window. The schedule interval filling can be defined as usage of the interval time between the pair of resource scheduled times in sequence. The scheduling strategy performs the search for optimal resource for a given tasks window in a hierarchical order. The hierarchical order of the search for optimal resource is as follows:

- A control frame respective to each triggered tasks window (here after referred as window) carries the requirements such as expected resource, time to engage that resource, the size of the window, window arrival time and its completion time.
- The arrival time of request window is the aggregate value of time required to reach resource allocation controller, volume of time required to process a control frame.

  $$\tau_{w_i} = t_{cf}(w_i) + p_{cf}(w_i) + t_{w_i} + \beta$$

  the aggregate value of arrival time $t_{cf}(w_i)$ of the control frame cf, process time $p_{cf}(w_i)$, time t_{w_i} required for the window to reach

ISSN(Online):22031731
ISSN (Print): 2204-0595

PublishedOnline12April 2021

resource allocation controller and elapsed threshold β defined.

2. MRS-OSIF Scheduling Strategy

Resource Allocation Controller executes MRS-OSIF to perform resource allocation to the window that represented by the control packet arrived, which is as follows:The adaptable to the requirements and idle time of the resource that suits to accomplish the completion of the taskswindow are two standards followed by proposed resourcescheduling strategy.RS-OSIF, upon failure to identify an individual resource that meets the scheduling criteria, then pools minimal set of resources to meet this scheduling criteria, if failed then selects one or more resources with maximal scheduling intervals (idle time between pair of schedule times in sequence) and schedule them to fulfill the requirements of the window to be arrived. If either of these cases succeeds, then segments the window in to minimum number of windows such that resource scheduling succeeds under specified factors. The resource allocation to the target window at scheduling intervals, which is the third level of the proposed scheduling hierarchy, is explored in following steps.

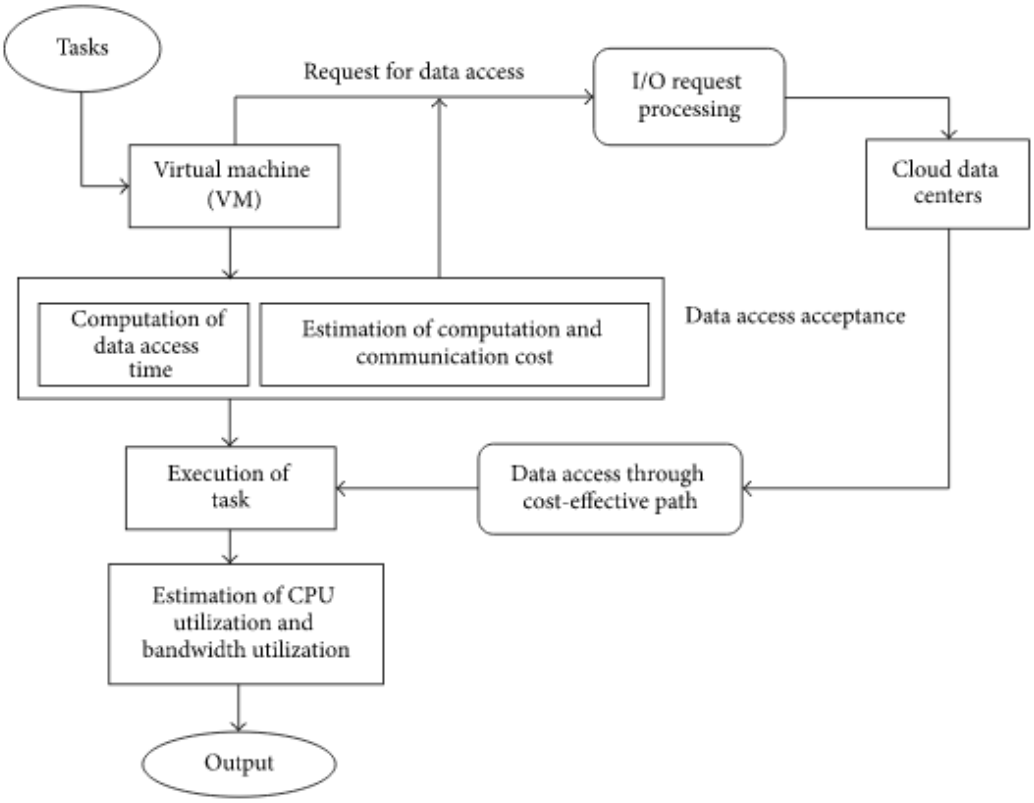


Figure 1. MRS-OSIF Scheduling Strategy

Schedules a resource to the windows w_k and w_l expected to be arrived at different times, if available with scheduling interval $i_{w_k \to w_l}$, such that

- $b(i_{w_k \to w_l}) < \tau_{w_i}$ // begin time $b(i_{w_k \to w_l})$ of the scheduling interval $i_{w_k \to w_l}$ is less than the arrival time τ_{w_i} of the window w_i.

ISSN(Online):22031731
ISSN (Print): 2204-0595

- $e(\acute{\imath}_{w_{\acute{k}} \to w_{\acute{\imath}}}) > (c_{w_i} + \beta)$ // end time $e(\acute{\imath}_{w_{\acute{k}} \to w_{\acute{\imath}}})$ of the interval $\acute{\imath}_{w_{\acute{k}} \to w_{\acute{\imath}}}$ is greater than the completiontime $(c_{w_i} + \beta)$ of the tasks in window w_i, here β is the elapsed completion time offset defined.

If failed to meet the above criteria, then selects minimalset of resources, which are already scheduled andhaving scheduling intervals such that,

- Scheduling Interval begin time of all the selected compatible resources are identical and less than the arrival time of the window, and sum of the scheduling intervals is greater than the completion time of the tasks found in given window. If found pools all the selected resources and schedules to the target window.

If failed to meet the above criteria, then segments the target window in to two and executes MRS-ISOF on each window

Algorithm: Resource Allocation Controller- $MRS(w_i, cf)$
*Step1:*Let cf be the control frame of respective window w_i, *Step2:* $\bar{r} \leftarrow \phi$ // vector of optimal resources, which is empty initially. *Step3:* $\bar{r} = MRS - OSIF(cf, R)$ //invoking a method that tracks optimal resource under three levels of RS-OSIF that meets the criteria of requirements found in cf respective to the window w_i, here R is the set of resources available. *Step4:*If $(\bar{r} \neq \phi)$ Begin • Partition the w_i in to two windows $\{\vec{w}_i, \overleftarrow{w}_i\}$ and apply RS-OSIF on each such that control frame cf represents the both windows. • $RS(\overleftarrow{w}_i, cf)$ // invoking main method for first part of the window. • $RS(\vec{w}_i, cf)$ // invoking main method for second part of the window. *Step5:*End // Step4 *Step6:* Else Begin // Step4. • If the size of the $\bar{r}$ is one then schedules that resource. • Else pools the all resources as one unit and schedules to the window w_i represented by cf . • Exit *Step7:* End // of condition in Step6. *Step8:* End // of the function.

Representation of optimal resource selection algorithm

Algorithm: Representation of optimal resource selection ($MRS - OSIF(cf, R)$)
*Step1:*Begin *Step2:* $er \leftarrow \phi$ // an empty vector that contains eligible resources identified during the process *Step3:* $\bar{r} \leftarrow \phi$ // an empty vector contains optimal resources to schedule found in the process. *Step4:*For-each $\{r \exists r \in R\}$ Begin

ISSN(Online):22031731
ISSN (Print): 2204-0595

Step5: IF $\left\{\begin{matrix}(b(nit_r)+\alpha)<(\tau_{w_i})\wedge \\ (e(nit_r)-b(nit_r))>(c_{w_i}+\beta)\end{matrix}\right\}$ then Begin // idle time frame $b(nit_r)$ that summed up withelapsed threshold α defined is less than the arrivaltime τ_{w_i} . In addition, the total idle time of the resource(which is the absolute difference between end andbegin of the idle time) r is greater than the expectedcompletion time c_{w_i} of the given task window thatsummed up with completion elapsed offset β defined.

Step6: $\bar{r} \leftarrow r$

*Step7:*Break the loop // Step4.

Step8: End //of the condition in Step5

*Step9:*End of the loop in Step3

Step10: IF ($\bar{r}$ is not empty)return $\bar{r}$ // completion of the method at first level of the hierarchy.

*Step11:*For-each $r \exists r \in \mathcal{R}$ Begin

Step12: if $((b(nit_r)+\alpha)<(\tau_{w_i}))$ then $er \leftarrow r$

Step13:End // of Step12

*Step14:*End // of Step11

*Step15:*IF $(er$ is not empty) Begin

- Sort the er as $\overline{er}$ in descending order of their idle time.
- $snit$=0 // aggregate of the idle times observed for selected resources in $\bar{r}$

Step16: For-each $\{r \exists r \in \overline{er}\}$ Begin

- $\bar{r} \leftarrow r$
- $snit+=(e(nit_r)-b(nit_r))$
- IF$(snit>(c_{w_i}+\beta))$ Begin
- Return $\bar{r}$ // completion of the method at second level of the hierarchy

Step17: End //of Step16

*Step18:*End//of Step15

Step19: $er \leftarrow \phi$ // empty the vector er

Step20: For-each $\{r \exists r \in \mathcal{R}\}$ Begin

Step21: IF $(b(si_r)+\alpha)<(\tau_{w_i}))$ Begin // if the begin of the schedule interval si_r of the resource r is less than the arrival time τ_{w_i} of the window w_i

*Step22:*End // of Step21

Step23: End // of Step20

*Step24:*IF $(er$ is not empty) Begin

- Sort the er as $\overline{er}$ in descending order of their schedule intervals.

*Step25:*For-each $\{r \exists r \in \overline{er}\}$ Begin

- $\bar{r} \leftarrow r$
- $snit+=(e(si_r)-b(si_r))$

ISSN(Online):22031731
ISSN (Print): 2204-0595

- IF($snit > (c_{w_i} + \beta)$) Begin
- Return $\bar{r}$ // completion of the method at third level of the hierarchy

Step26: End // of Step25
Step27: End // of Step24
Step27: Return $\bar{r}$
*Step28:*End //of the Method

3. Results and analysis

The performance of MRS-OSIF is assessed through simulation study performed on Planet Lab is used to simulate the distributed cloud computing environment with stream of tasks and rationally virtualized multiple resources. The performance of the MRS-OSIF is assessed by the metrics task load versus resource allocation failure, task load versus task completion optimality schedu ling process overhead. The parameters used in simulation environment are as follows

Table 1. Parameters used in Simulation Environment

Number of users	125
No of Resources and their virtualizations	155
The range of tasks involved to form a Request window	11 to 25 Similar Tasks
The range of million instructions per request window	0.1 to 1
Range of task priorities	5 to 15
Elapsed threshold values used	0.05% of actual

The similar tasks were pooled as window in the range of 11 to 25 tasks in each window. The proposed MRS-OSIF is implemented in java and deployed as frontend of the simulation. The resources scheduled and the tasks completion status was logged along with MRS-OSIF execution flow. The execution flow logs were used to estimate the process overhead and the logs of scheduled resources and tasks completion state were used to assess the tasks load versus resource allocation failures and tasks completion optimality. The evinced results for these metrics at divergent load of tasks were compared with the results obtained from other contemporary models FRAS and AHP. The comparison of Load versus resource scheduling failures observed for MRS-OSIF, FRAS and AHP were analyzed and represented in Figure 2 as line chart, which is concluding that the proposed model is 39%, 28% of scheduling failures were reduced that compared to FRAS and AHP respectively.

Table:2. Resource Scheduling Failure ratio against window load as set of instructions in millions.

Set of Instructions (in millions)	Models		
	FRAS	AHP	MRS-OSIF
0.1	0.48	0.47	0.295
0.2	0.51	0.48	0.331
0.3	0.54	0.52	0.333
0.4	0.76	0.57	0.344
0.5	0.79	0.59	0.352
0.6	0.81	0.67	0.381

ISSN(Online):22031731
ISSN (Print): 2204-0595

PublishedOnline12April 2021

0.7	0.88	0.71	0.395
0.8	0.92	0.79	0.402
0.9	0.97	0.83	0.438
1	0.99	0.87	0.44

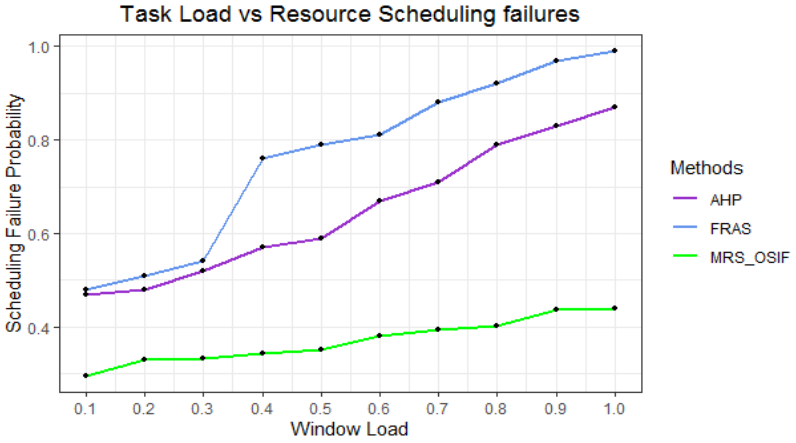
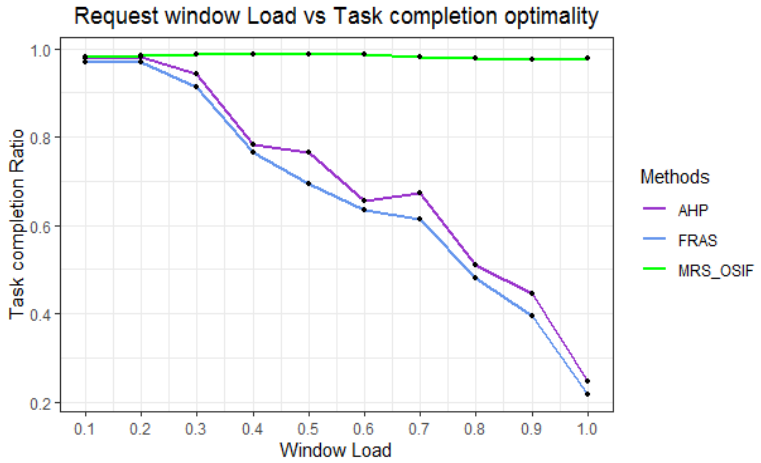


Figure.2: Request window Load versus resource allocation failures.

The task completion optimality observed for MRS-OSIF and other two models were analyzed and compared in Figure 3. The comparison of task completion optimality observed for all of these three models evincing that the MRS-OSIF is maximizing the task completion optimality by 31%, 28% in respective of FRAS and AHP.Process overhead observed against request window load (see Figure 3) is evinced as linear in the case of MRS-OSIF, where in other two cases the process overhead is nonlinear (NP-Hard).

Table 3. Task completion ratio against window load as set of instructions in millions

Set of Instructions (in millions)	Models		
	FRAS	AHP	MRS-OSIF
0.1	0.971	0.979	0.981
0.2	0.969	0.981	0.984
0.3	0.913	0.943	0.986
0.4	0.764	0.784	0.987
0.5	0.695	0.765	0.987
0.6	0.634	0.654	0.986
0.7	0.614	0.674	0.981
0.8	0.481	0.511	0.979
0.9	0.394	0.444	0.976
1	0.217	0.247	0.979



Online):22031731
ISSN (Print): 2204-0595

PublishedOnline12April 2021

Figure.3:Request window Load versus task completion optimality.

The ratio of request window loss against the request window load is evinced in Figure 2. The request window load is normalized to the value between 0 and 1, which is actually the number of pool of tasks as window per unit of time. The experimental study indicating that the MRS-OSIF is significantly defused the window loss that compared to other two models (see Figure 2 and Table 3). Hence the high task accomplishment observed for MRS-OSIF (see Figure3 and Table 14). The conditional execution of the levels of hierarchical order followed by MRS-OSIF and allocation of resources to the pool of tasks also the context of pooling more than one resource to fulfill the need of a tasks window in the Table 3: Process overhead Ratio observed against window load as instructions per window in million second the process overhead as linear (see Figure.4).

Table 5: Process overhead Ratio observed against window load as instructions per window in millions

Set of Instructions (in millions)	Models		
	FRAS	AHP	MRS-OSIF
0.1	0.257	0.301	0.251
0.2	0.396	0.337	0.303
0.3	0.494	0.413	0.345
0.4	0.613	0.452	0.375
0.5	0.535	0.504	0.375
0.6	0.644	0.563	0.395
0.7	0.717	0.607	0.43
0.8	0.781	0.649	0.463
0.9	0.748	0.709	0.472
1	0.807	0.753	0.507

ISSN(Online):22031731
ISSN (Print): 2204-0595

Figure 4: Process overhead versus request window load

The resource utilization ratio is also being assessed (see Table 5 and Figure 4) for proposed model and the other two models as multiple instructions per second (MIPS), which is the benchmark standard considered for experiments. The utilization ratio is measured proposed by standard performance Evaluation Corporation .

Table 6: Resource Utilization Ratio (as Million Instructions per Seconds)

Set of Instructions (in millions)	Models		
	FRAS	AHP	MRS-OSIF
0.1	0.00036	0.00037	0.00454
0.2	0.00036	0.00038	0.00307
0.3	0.00062	0.00071	0.00986
0.4	0.00079	0.00089	0.00512
0.5	0.00032	0.00033	0.00213
0.6	0.00096	0.00099	0.00357
0.7	0.00056	0.00064	0.00404
0.8	0.00067	0.00072	0.00799
0.9	0.00017	0.00025	0.00499
1	0.0008	0.00083	0.00584

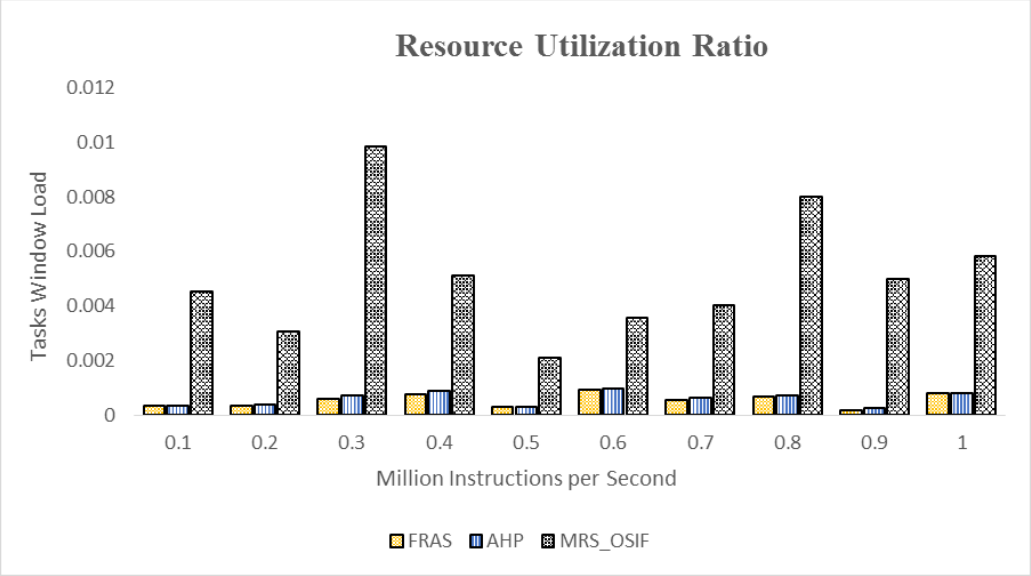


Figure 5:Resource utilization ratio in million instructions per second.

4. Conclusion

The proposed model schedules the resources in hierarchical order. In first level of the hierarchy, tracks an idle resource that fulfils the priorities of the tasks window, if failed then tracks the set of idle resources and pools them to fulfill the need, if failed then tracks for one or more resources with compatible scheduling intervals and pools them and

ISSN(Online):22031731
ISSN (Print): 2204-0595

schedules to respective tasks window. If failed to meet the any of the above criteria of the hierarchy, then reforms the tasks window, such that the available resources in current context can fulfill the requirements of the tasks window. The experimental study evincing that the proposed model is robust in resource scheduling with optimal task completion time and minimal resource allocation failures. Since the allocation strategy is performed in hierarchical order and execution of each level in hierarchy is conditional, the computational overhead is found as linear. The maximal resource utilization with minimal virtual machines and less computational over head is observed since the resources are allocated to the pool of tasks, instead to an individual task.

References

[1]. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, 1. Brandic. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25(6), p. 599-616.

[2]. Esha Bansal, Nisha Bansal, 2016. “An Analysis of Cloud Computing”. International Journal on Computing and Corporate Research (IJCCR), Vol.1, Issue 3, Manuscript 7, November 2016

[3]. An Oracle White Paper in Enterprise Architecture, 2016. ”Architectural Strategies for Cloud Computing”, August 2013.

[4]. KwangSik Shin, MyongJin Cha, MunSuck Jang, JinHa Jung, WanOh Yoon, SangBang Choi. Task scheduling algorithm using minimized duplications in homogeneous systems. Journal ofParallel and Distributed Computing, 2016, 68(8) , p. 1146-1156.

[5]. Braun TD, Siegel HJ, Beck N, etc. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Joumal of Parallel and Distributed Computing, 2001, 61(6) , p. 810-837.

[6]. XiaoShan He, Xianhe Sun and Gregor von Laszewski. QoS guided Min-Min heuristic for grid task scheduling. Journal of Computer Science and Technology, 2003, 18(4) , p. 442-451.

[7]. Ching-Hsien Hsu, Zhan, J., Wai-Chi Fang, et al.Towards improving QoS-guided scheduling in grids. 2017 Third ChinaGrid Annual Conference (CHINAGRID). Dunhuang, Gansu, China, 2008, p. 89-95.

[8]. Y. C. Lee, A. Y. Zomaya, Rescheduling for reliable job completion with the support of clouds, Future Generation Computer Systems 26 (2017) 1192_1199

[9]. E. Vineka, P. P. Beranb, E. Schikutab, A dynamic multiobjective optimization framework for selecting distributed deployments in a heterogeneous environment , Procedia Computer Science 4 (2011) 166–175

[10]. V.M. Lo, "Task assignment in distributed systems", PhD dissertation, Dep. Comput. Sci., Univ. Illinois, Oct. 1983.

[11]. G. Gharooni-fard, F. Moein-darbari, H. Deldari and A. Morvaridi, Procedia Computer Science, Volume 1, Issue 1, May 2010,Pages1445-1454 ,ICCS 2017.

[12]. Baomin Xu, Chunyan Zhao, Enzhao Hua, Bin Hu. Job scheduling algorithm based on Berger model in cloud environment. Advances in Engineering Software, 2011, 42(7) , p. 419-425.

[13]. Xiaoyong Tang, Kenli Li, Renfa Li, Bharadwaj Veeravalli. Reliability-aware scheduling strategy for heterogeneous distributed computing systems. Joumal ofParallel and Distributed Computing, 2017, 70(9) , p. 941- 952.

[14] W. Zhang, Y. Wen, and D. O. Wu, “Energy-efficient scheduling policy for collaborative execution in mobilecloud computing,” *Proc. - IEEE INFOCOM*, pp. 190–194, 2016.

[15] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing

ISSN(Online):22031731
ISSN (Print): 2204-0595

PublishedOnline12April 2021

Environments," in *2010 24th IEEE InternationalConference on Advanced Information Networking and Applications*, 2010, pp. 400–407.
[16] Dr. R. Manikandan, Dr Senthilkumar A. Dr Lekashri S. Abhay Chaturvedi. "Data Traffic Trust Model for Clustered Wireless Sensor Network." INFORMATION TECHNOLOGY IN INDUSTRY 9.1 (2021): 1225–1229. Print.

ISSN(Online):22031731
ISSN (Print): 2204-0595