International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

Coupling Factor and Cost Based Task Clustering Method to Optimize Task Clustering For Scientific Workflows in Cloud Environment

J. Jabanjalin Hilda, C.Srimathi

Abstract - Scientific workflows are large scale loosely coupled submissions that are used by Computational Scientists. They are composed of multiple tasks with dependencies between them and are composed of many fine granular tasks. Task clustering is an optimization method that combines multiple tasks into a single job such that task execution time and system overhead is reduced and thus the whole performance is improved in a cloud environment. Though existing task clustering algorithms has significantly reduced the System overhead, yet dependencies among the tasks are not well-thought-out. This work examines the features of task by which the tasks can be clustered and developed proficient task clustering algorithm. In this work two task clustering ideas were proposed namely Horizontal Coupling Factor (HCF) based clustering and Horizontal Processing Cost (HPC) based Task Clustering. Next, the proposed algorithm have been evaluated and tested for various real world applications and the experiment results shows that the proposed approach suits best for data intensive and Compute intensive applications. The obtained results showed that the HCF and HPC task clustering strategies can significantly improve the performance by reducing the task execution time and inter task Communication delay.

Keywords - Coupling Factor, Clustering, Execution Time.Processing Cost, Scientific Workflows,

I. INTRODUCTION

Huge scale loosely coupled applications are referred as scientific workflows. Several computational scientists progress and practise these large scale applications. Some of the specialized scientific workflows are Cybershake, Epigenomics, LIGO, Montage, Broadband, etc. Execution of data intensive workflows in an efficient manner is one of the advantages that the cloud environment has proposed. Researchers often need to automate data processing steps, replicate past outcomes, share their contemplate steps with other researchers, track those Inception for data products, and perform analyses in parallel and distributed resources and so on. [1]. Scientific workflows offer solution to these problems. These scientific workflows can be expressed as Directed Acyclic Graphs (DAGS) and can be parallelized onto distributed resources. For instance, Montage has 18 million input images (approx... 2.5 TB), 900 output images (approx... 2.4 TB), 10.5 million tasks (34,000 CPU hours).

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
J. **Jabanjalin Hilda*,** School of Computer Science and Engineering, Vellore Institute of Technology, Vellore – 632014, India
C.Srimathi, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore – 632014, India

Scientific workflow management framework is designed to routinely allocate data and computations for these huge applications. More computational power is required to run workflows, since the workflows contains of huge number of computational tasks.

The Scientific Workflow Management Systems (SWFMS) automatically allocate data and computations for these huge applications. They compile abstract workflows to executable workflows. Scientific workflows are collection of dependent and inter-dependent tasks and these scientific workflows consists of many computational tasks with dependencies among them and they are of short running and long running tasks. [2]. these applications are executed on multi-machine distributed environment, such as cloud .On execution, substantial overheads may exist and may badly back off the application performance. To minimize this effect, Task clustering techniques come into picture. This increases the performance and scalability. Task clustering group's fine grained tasks into coarse grained tasks, so that the no of computations are reduced and their compute granularity is increased thus reducing the system overheads.

Data dependencies show a significant role when grouping the tasks within the level or pipeline. Task clustering which is addressed in the existing work, suffer from data locality problems and due to which it is poorly distributed. The data dependency is data transfer between tasks .i.e. output data for one is given as input data for the other. This caused an increase in Failure probability and the data transfer rate. In order to address these issues, dependency based clustering methods are introduced in this work. There is a trade-off between runtime imbalance and dependency imbalance. A performance assessment study shows that the proposed method significantly reduces the dependency problem. Customarily there are two methodologies to improve the performance of task grouping. One is the top down approach and the other is the bottom up approach, which optimizes globally and locally. Our work outspreads these approaches by even considering the strong connections between the tasks. Granularity size is the period within which a job is handled at the resources. It is used to measure the entire amount of jobs that can be finished within a quantified time in a specific resource.

Also, heavily communicating tasks plays a major role in the workflows. Hence, heavily communicating tasks are grouped into a cluster and then assign these tasks to the similar resource. Workflows consists of various tasks and are assigned to different resource types and it is stated as heterogeneity in workflow. Tasks can be grouped based on set of tasks that run concurrently and distribute them to various sub workflows.

This can reduce the inter-communication cost and attain high parallelism .Thus task to resource map can be done efficiently.

The proposed work focusses on reducing the number of resources. In case of high contention scenarios the number of resources are limited. In such scenarios, if the task clustering is good, the task execution time and the system overhead can be reduced. Also, in case of large number of tasks, the task clustering technique should be such that the resource wastage is reduced.

CloudSim is a framework for simulating and modelling cloud computing services. WorkflowSim[3] is a toolkit for simulating scientific workflows (SWFs) in distributed environments. In this work, WorkflowSim is used to test the algorithm in the distributed environment. The rest of the article is structured as follows. Section II gives an overview of related work. Section III shows the Architecture of workflow management System, Section IV shows the proposed task clustering strategies, Section V shows the illustrations on the Experiments and Results and Section VI ends with the Conclusion.

II. RELATED WORK

In cloud systems, the low performance of fine-grained tasks is one of the common issues in extensively distributed platforms. This causes system overhead and high queuing times at resources. The decrease in system overhead is a subject of great attention in distributed platforms. Numerous works have addressed the task clustering strategies by controlling the task granularity. [4] Suggested task grouping strategies according to the processing capabilities (MIPS) of available resources and the granularity size. This need has diminished the communication overhead time and processing overhead time of every user job. [5] Examined the reason that cause Runtime Imbalance and Dependency Imbalance and proposed task balancing methods to solve imbalance problems. The Horizontal Impact Factor Balancing and Horizontal Distance Balancing (HIFB and HDB) methods performs better when data dependency is more important. This approach has improved the runtime performance and data dependencies, but failed to address fault tolerance strategies. [6] Proposed a technique and tested with Dynamic Clustering workflows (DCW) algorithm that can reduce inter-task communication cost, improve makespan and achieve high parallelism. Here clustering is based on the heavily communicating tasks. Though the makespan is increased the task execution time is not reduced for data intensive applications. The [7] fault tolerant clustering algorithms such as dynamic re-clustering, Selective re-clustering were discussed. But the model is not tested in real world applications. [8] The paper has incorporated dynamic re-clustering and selective re-clustering fault tolerant strategies which can be applied for homogeneous environments but not for heterogeneous environments. The explanations for the cause of Runtime and Dependency Imbalance in Task Clustering are discussed in [9]. Their balancing Methods address the load balancing problem by merging the task based on Runtime, Impact Factor and Distance. Runtime imbalance are Dependency imbalance and is calculated for task clustering. Horizontal Runtime Balancing algorithm, Horizontal Impact factor Balancing algorithm and Horizontal distance Balancing Algorithm(HIFB and HDB) are discussed in the paper to reduce the execution time. Graph-skeleton based clustering has been proposed for task clustering. Barycentre's were used for cluster identification, workflow ranking and recommendation. [9]. [10] has developed Parallel Task Execution (PTE) algorithm to schedule the tasks in reduced time and cost and detect transient failures. [11] has compared various Fault Tolerant strategies and concluded saying by integrating Fault Tolerant strategies with clustering method can yield better scheduling. [12] has developed fault-tolerant aware dynamic clustering league championship algorithm using dynamic clustering strategy. [13] has designed an intelligent task failure prediction model to predict tasks failures. Nevertheless, the QoS parameters like scalability and availability are not considered. Graph skeleton based clustering has been proposed for task clustering. Barycentre's were used for cluster identification, workflow ranking, and recommendation.[14]. [15] has proposed a task clustering technique named Workflow Platform Aware task clustering, which identifies no. of clusters at each level, to achieve parallelism. This also considers workflow structure and resource set size for task clustering which minimizes system overheads, reduces execution time and resource wastage. They groups tasks that can run in parallel into different clusters and create equal sized clusters.

In this work the information obtained from previous works on different parameters like impact factor and distance is taken as a foundation to develop Horizontal Coupling factor (HCF) based task Clustering and Horizontal Processing Cost (HPC) based clustering technique. The algorithm was proposed mainly to reduce the task execution time for data intensive and IO intensive applications. Summary of different clustering methods are discussed in Table 1.

TABLE 1
SUMMARY OF TASK CLUSTERING TECHNIQUES

Paper	Clustering Technique	Methodology
[5]	Balanced Clustering	-Sort by Clustering Factor and Available Resources. -Size of the list of Clustered Job (CL) =Available resources X Clustering Factor.
[8]	Horizontal Clustering	-Cluster by horizontal level
[8]	Vertical Clustering	-Cluster by Workflow pipelines
[9]	Horizontal Runtime Balancing(HRB)	-Cluster based on Longest Runtime +Job with Shorted Runtime
[9]	Horizontal Impact Factor Balancing(HIFB)	-Sort by Impact Factor, then on Runtime
[9]	Horizontal Distance Balancing(HDB)	-Sort by Distance , then on Runtime
[9]	VC Prior	-Cluster by Vertical clustering first , then Horizontal Clustering
[9]	VC Posterior	-Cluster by Horizontal Clustering first , then Vertical Clustering
[10]	Workflow and Platform Aware Clustering	-Cluster by longest parent and indegree

III. ARCHITECTURE OF WORKFLOW MANAGEMENT SYSTEM(WMS)

A. Architecture of Workflow Management System

The architecture of the Workflow Management System is shown in Fig. 1 .The Workflow is exhibited as a Directed Acyclic Graph (DAG).The Workflow consists of Submit host and Execution Site. The components in the Submit host are WF Mapper, WF Engine, Local Queue and Job Scheduler. The components in the execution Site are Job wrappers and Remote Queue.

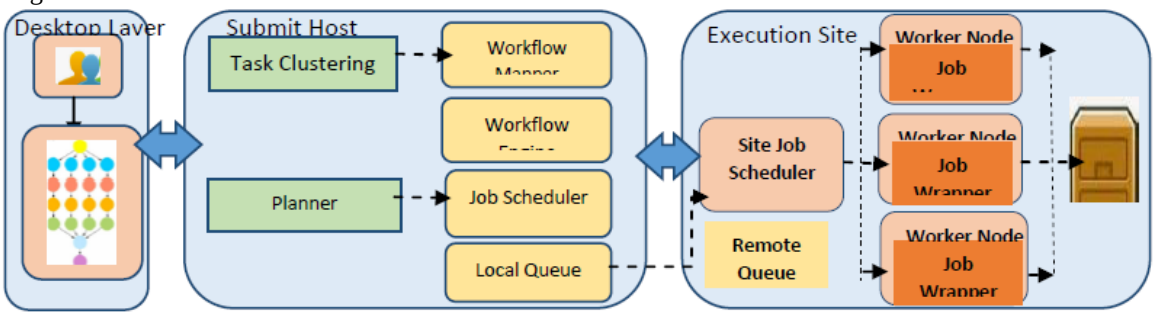


Fig. 1 Workflow System Model

The Components in the Submit host is explained below:

The Submit host performs clustering and mapping and the Worker Nodes in the execution site executes the jobs.

Workflow Mapper: It produces an executable workflow from an abstract workflow. Also it combines the smaller tasks into clustered job.

Workflow Engine (Local Execution Engine): Based on the dependencies the tasks are executed. The jobs are submitted to the Job Scheduler only when its parent jobs are completed. Workflow engine delay is the elapsed time when a job is released to the time when the job is submitted to the job scheduler.

Local Job Scheduler and local Queue: It manages and controls the execution of the jobs on local and remote resources. The scheduler relies on the compute, storage and network resources defined in the executable site. Queue delay is the time of submission of the job by the workflow engine to the local queue to the time of start its execution in the worker node.

Job wrapper: Executes the tasks in the worker node by extracting the tasks from the clustered jobs. The time of extraction process is the clustering delay.

Execution Site: It executes the tasks on the remote worker nodes.

B. Scientific Workflow Applications

In this experiment five scientific workflow applications are tested with the proposed algorithm. They are Montage, Cybershake, Epigenomes, SIPHT and LIGO. Here the structure and characteristics of the workflow were studied.

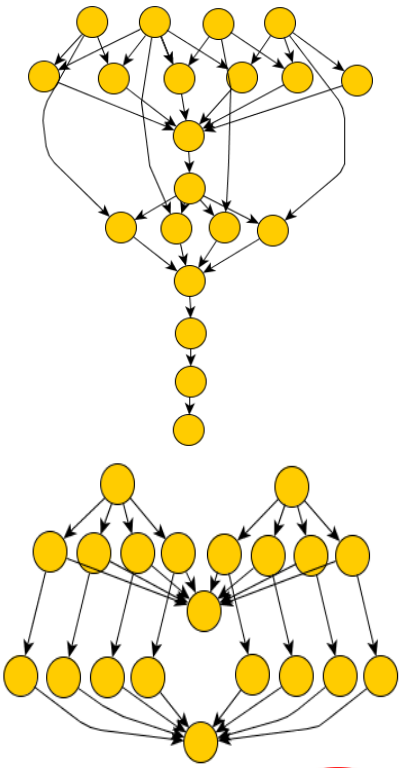
Montage[16]: This is an astronomical submission which is used to generate image montages of the sky. The mosaic image deliver a meticulous thoughtful of the slice of the sky. Fig.2(a) shows the structure of the Montage workflow. Most of its tasks do not involve much processing capacity and stay characterized as IO intensive. The structure of the workflow changes when increase in the tasks.

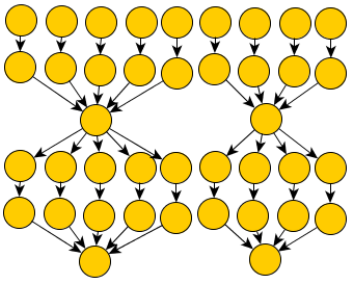
Cybershake[17]: This is a seismology application that calculates seismic Hazard curves. It recognises rupture within 200 kilometres. Fig .2(b) shows an illustration of the Cybershake workflow. Most of its tasks require large memory and CPU and stay characterized as data intensive.

LIGO [18]: This workflow is used to explore gravitational wave signatures produced by various events in the signatures. The workflow consists of compute intensive applications and consumes large memory. Fig .2(c) shows an illustration of the LIGO workflow.

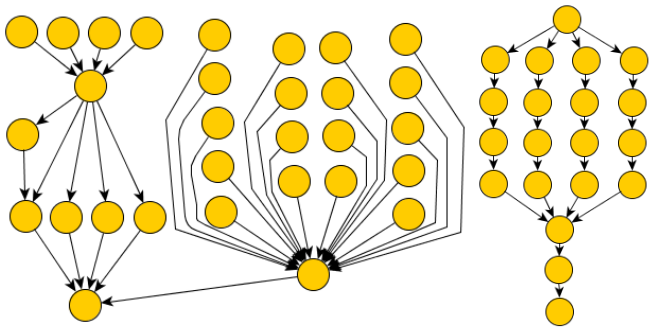
SIPHT[19]: This workflow conducts an exploration for sRNA that adjusts the processes such as secretion or virulence in bacteria. Using Pegasus sRNA is predicted. The task execution time of SIPHT is high and utmost of the tasks have high CPU utilization and low IO utilization. . Fig .2(d) shows an illustration of the SIPHT workflow.

Epigenomics[20]: The DNA sequences generated by Illumina-Solexa Generator are converted into reference genome or assemble an entire genome.. The scientific workflow maps DNA sequences to right locations in a reference genome. The epigenomics structure is symmetric and the application is CPU intensive. Fig .2(e) shows an illustration of the Epigenomics workflow.





(a) Montage (b) Cybershake (c) LIGO



(d) SIPHT (e) pigenomics

Fig. 2 Scientific Workflows

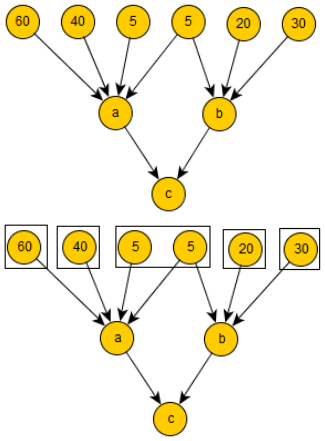
IV. THE PROPOSED TASK CLUSTERING STRATEGIES

A. Horizontal coupling factor Algorithm (HCF)

This section explains the proposed Horizontal Coupling factor based Task clustering technique (HCF) which suits for symmetric workflow structure .Unsuitable task clustering may lead to negative impact on the workflow execution time. This reduces the system overhead and at the same time also reduces the execution time. HCF technique merges multiple tasks within the same level in the workflow with decreasing order of task run time and coupling factor .Fig. 3 shows how the coupling factor can be calculated. The coupling factor for the task 'a' is 4 and the coupling factor for the task 'b' is 3. The tasks are first sorted in decreasing order. The tasks are clustered based on the highest parent. The highest parent is 60. The remaining tasks are clustered such that the clusters are less than or equal to the highest parent. 40, 5 and 5 are the remaining parent tasks for the task 'a'. Task with the runtime 5 can be combined together and task with the run time 40 can be clustered separately. As the tasks with the highest coupling factor is clustered first, the remaining tasks can be completed soon. Similarly for the task 'b', the highest parent is 30.So 20 can be clustered as another cluster. So this method takes care of both runtime variance and also dependency variance [9].

Algorithm 1: Horizontal coupling factor Algorithm:

```
Input: W:Workflow; JL:no. of jobs per Horizontal Level
Extract all the Level (L) from the source to destination
Begin:
1.    for every level<Depth(D) do
2.        Task_List= GetTaskatlevel(W,Level)
3.        Cl_Size=Task_List/JL
4.        for(i<R)
5.            Cl_list_i={}
6.        end for
7.        Sort task based on coupling factor
8.        for all tasks in Task list
9.            T=Sort the tasks in decreasing order of task runtime
10.           J=Group the job with the shorted runtime
11.           J.append(t)
12.       end for
13.       for(i<R)
14.           Cl_list_i.append(J)
15.       end for
16.       W = W - Task_List + Cl_list
17.   end for
End
```



(a) Example Workflow (b) Clustering based on Coupling Factor

Fig. 3 Horizontal Coupling Factor based clustering

Algorithm 1, aims to distribute the tasks based on coupling factor. The workflow and the number of jobs per level is given as input. For each level in the workflow, line number 1-3 identify the task list at level and calculate the cluster size. Line number 7 to 12, sorts the tasks based on coupling factor and then sorts based on the runtime. Finally cluster is formed by merging of tasks with the maximum coupling factor. Combination of tasks are called as a Job. The coupling factor captures the structure of the Workflow and also they reduces the overhead.

B. Horizontal Processing Cost Algorithm (HPC)

Horizontal Processing Cost (HPC) based clustering aims at clustering the tasks based on the Impact factor, runtime and processing cost. Impact factor variance [9] of tasks captures the similarity of

tasks. It is calculated using the formula $IF(t_a) = \sum_{t_b \in Child(t_a)} \frac{IF(t_b)}{\|Parent(t_b)\|}$

Where, $Child(t_a)$ is the set of child tasks of t_a, $IF(t_b)$ is the impact factor of t_b, $Parent(t_b)$ is the number of parents tasks of t_b. The impact factor for the Fig. 4 is calculated as follows:

IF(c) =1

IF (a) =1/4=0.25

IF (b) =1/3=0.33

IF (d)=1/16+1/4=0.31
IF (e) =IF (f) =1/16=0.06
IF (g)=1/16+1/9=0.17
IF (h) =1/9=0.11
IF (i)=1/9+1/4=0.36

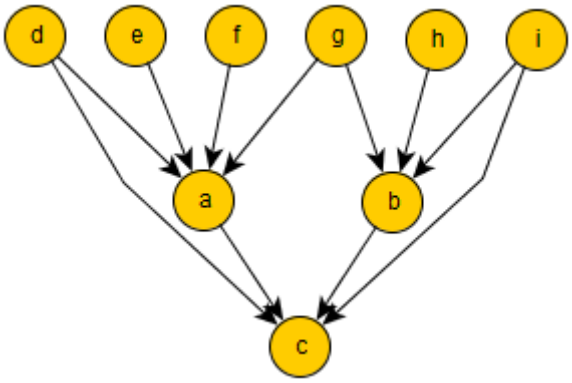


Fig. 4: Workflow with data dependencies

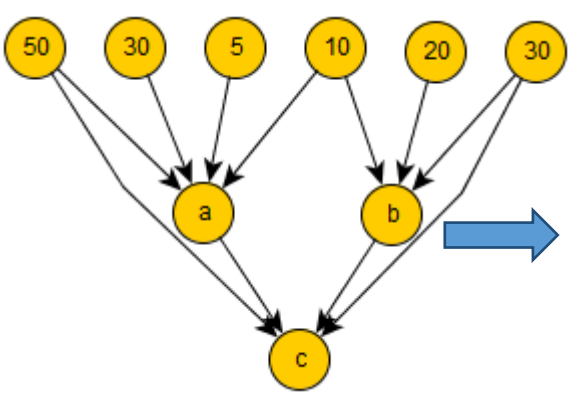


Fig. 5 Example Workflow

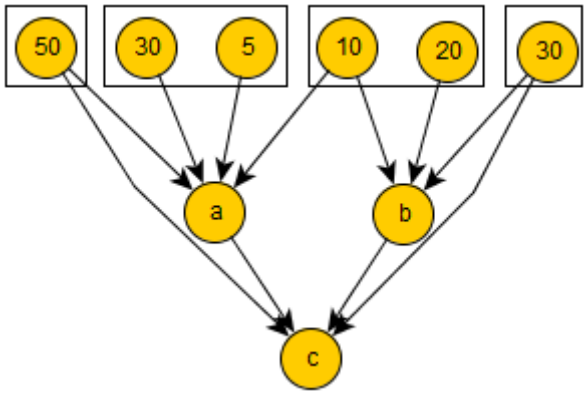


Fig. 6 Clustering based on Processing Cost

Horizontal Processing Cost (HPC) based algorithm, initially sort the task based on Impact factor and runtime, then with respect to task runtime and finally group the tasks based on the processing cost. For the example workflow which is shown in Fig. 5, the clustering outcomes are depicted in Fig. 6. The processing cost is calculated using the formula Processing cost=Input data transfer + Processing cost + Output transfer cost

Algorithm 2: Horizontal Processing Cost Algorithm

```
Input: W:Workflow; JL: no of jobs per Horizontal Level
Extract all the Level (L) from the source to destination
Begin:
1.      for every level<Depth(D) do
2.              Task_List= GetTaskatlevel(W,Level)
3.              Cl_Size=Task_List/JL
4.              for(i<R)
5.                      Cl_list_i={}
6.              end for
7.              Sort the tasks based on Impact factor and Runtime
8.              for all tasks in Task list
9.                      T=Sort the tasks based Processing Cost
10.                     J=Merge with Shortest task runtime and less than 't' tasks in T
11.                     J.append(t)
12.             end for
13.             for(i<R)
14.                     Cl_list_i.append(J)
15.             end for
16.             W = W - Task_List + Cl_list
17.     end for
End
```

In Algorithm II , line no 7 to 11, calculates the Impact factor and runtime and group the task based on these parameters and then cluster based on the processing cost.

V. EXPERIMENT AND EVALUATION

In this section, simulation of the proposed algorithm is presented. The experiments show the evaluation of the performance of our algorithm in comparison with the existing task clustering algorithms.

A. Experimental Conditions

The WorkflowSim simulator tool is extended with the proposed task clustering techniques, where we could evaluate the performance by varying the virtual machines and data size. The workflowsim platform comprises of 20 homogeneous core virtual machines where most of the cloud platforms such as Amazon EC2 and Future Grid uses. The machine has 512MB of RAM and the processing capacity is 1000 million instruction per second.

The workflow applications such as Montage, Cybershake ,and LIGO, Epigenomics and SIPHT are used for analysis. The workflow structure of these real world applications are presented in Fig. 2. These workflow applications consists of 100 tasks. The task runtime, Impact Factor, Coupling Factor and Processing Cost information were collected from the simulation tool and these parameters are used for Clustering.

Two sets of Experiments were conducted. Experiment 1 targets at determining performance gain (µ) of the proposed HCF and HPC algorithms and the existing algorithms over the execution without task clustering. This experiment assist in finding the degree to which a given task clustering approach can improve the whole performance of the workflow execution. The performance gain (µ) is calculated using this formula.

P

erformance Gain (µ) =

$$\left(\frac{\text{Total Exec. Time without clustering} - \text{Total Exec. time with Clustering}}{\text{Total Exec. Time without clustering}}\right) * 100$$

B. Results and Discussion

Experiment -1:

The performance gain obtained by the two proposed algorithms and the baseline algorithms is shown in Fig. 7 .It is experiential that the proposed algorithm exhibit a positive performance gain and thus improves the overall execution time of the 4 workflow applications. The proposed HCF and HPC clustering method gives better performance for Montage and Cybershake workflow application than HIFB and HDB. Epigenomics and SIPHT has higher runtime variance and due to this reason Horizontal Clustering(HC) technique gives lower performance and Montage and LIGO gives better gain for HC. Vertical Clustering(VC) have higher performance only for Montage Workflow and for the other workflows they have lesser gain. This is due to the fact that Montage has lesser runtime variance when compared to other workflows. HRB gives better gain for Epigenomics and LIGO due to higher runtime variance. HCF performs better for SIPHT when compared to other methods except HIFB. SIPHT has higher CPU utilization and lower I/O utilization. Also, Montage and Cybershake have fine granularity tasks and the proposed algorithms gives better results for data intensive and IO applications.

Experiment II:

Fig. 8 and Fig. 9 shows the performance gain for Montage and Cybershake workflows when varying the number of Virtual Machines(VMs). In Montage, with the increasing in the number of VMs, there is a gradual increase in the performance gain. When the number of resources are less, i.e during high contentions, Montage gives lesser gain. But when the number of VMs are increased, there is increase in gain. With respect to Cybershake , there is no change in the gain . All the balancing methods gives better gain except Horizontal and Vertical Clustering. For Symmetric structures and lesser runtime variance, there is no much difference in the number of virtual machines.

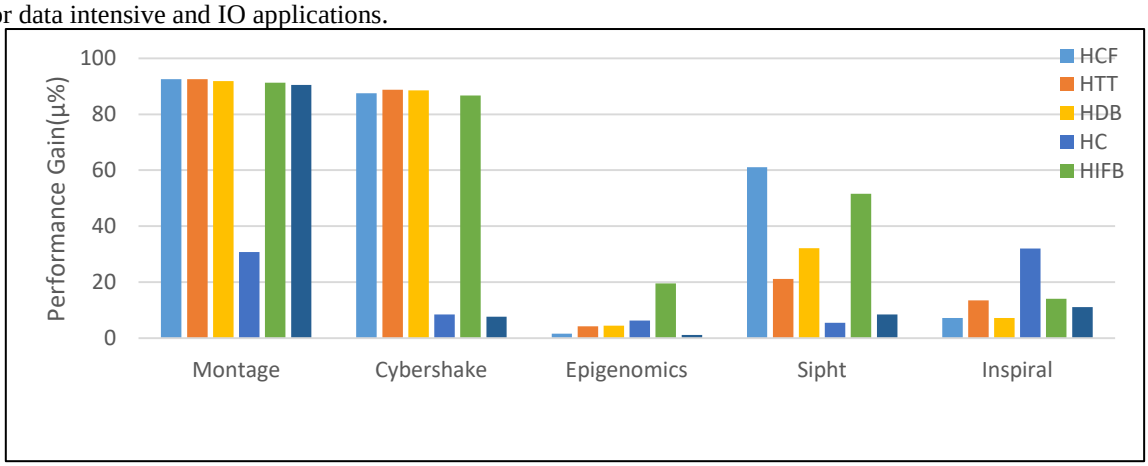


Fig. 7 Performance Gain(µ%) of HCF and HPC over existing four algorithms with number of tasks =100

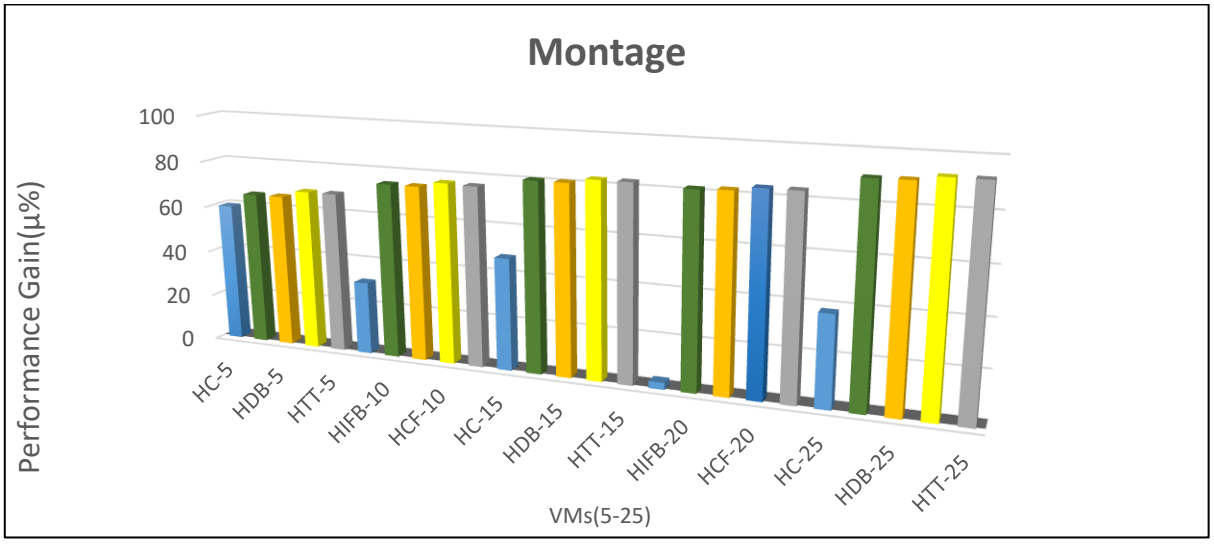


Fig. 8. Performance Gain (μ %) of HCF and HPC over existing algorithms with varying VMs (5-25) for Montage

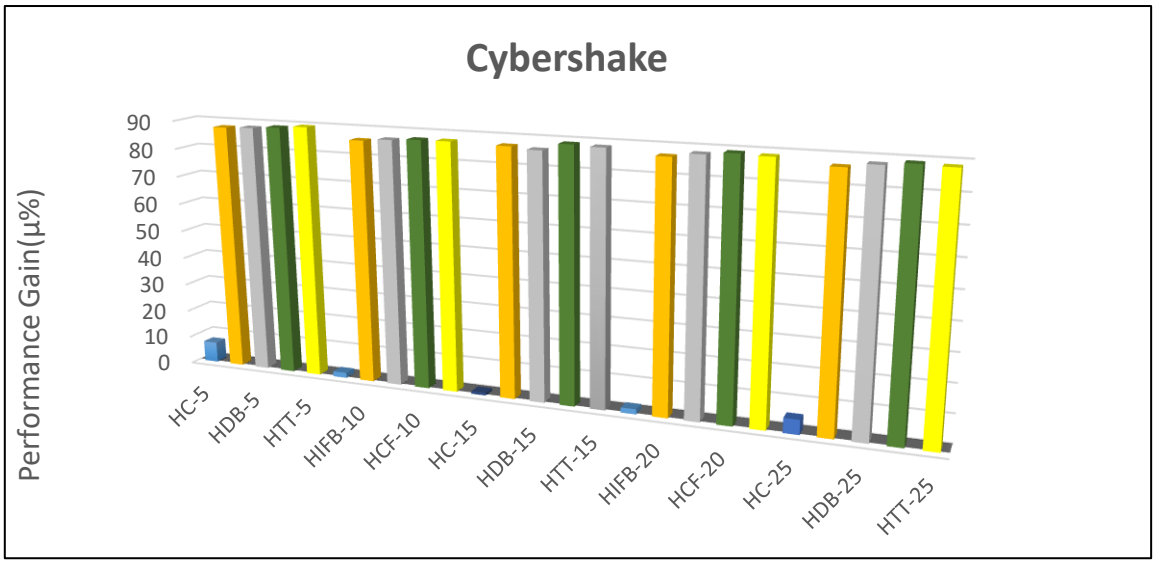


Fig. 9 Performance Gain(μ%) of HCF and HPC over existing algorithms with varying VMs(5-25) for Cybershake

VI. CONCLUSION AND FUTURE WORK

The proposed task clustering techniques (HCF and HPC) allows task clustering for data-intensive and I/O intensive scientific workflow applications while reducing the system overhead, data transfer time and task execution time. The algorithms were implemented and tested using workflowsim simulation tool. The two algorithms are compared with the existing HIFB and HDB task clustering approaches. The proposed approach considers the workflow structure for gaining preferred clustering. Firstly, the HCF algorithm uses a coupling factor that captures the structure of the workflow and the experiments show that this algorithm can be appropriate for data-intensive and IO intensive applications. The experiments illustrates the performance gain obtained for HCF and the results reveal that the runtime and dependency imbalance is reduced. Secondly, the proposed HPC algorithm uses Impact factor, runtime and processing cost, because of which it reduces the data transfer time. The experiment aimed at measuring the performance gain for various number of virtual machines (VMs). It resulted in an increase in the gain for Montage and Cybershake workflow applications.

In the future, we will explore how to optimally select the resource size and cluster number and in what way HPC and HCF can be extended to work for all real world scientific workflow applications.

REFERENCE

1. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., ... & Wenger, K. (2015). Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, *46*, 17-35.
2. Atkinson, M., Gesing, S., Montagnat, J., & Taylor, I. (2017). Scientific workflows: Past, present and future.
3. Chen, W., & Deelman, E. (2012, October). Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th International Conference on E-Science* (pp. 1-8). IEEE.
4. Muthuvelu, N., Liu, J., Soe, N. L., Venugopal, S., Sulistio, A., & Buyya, R. (2005, January). A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids. In *Proceedings of the 2005 Australasian workshop on Grid computing and e-research-Volume 44* (pp. 41-48). Australian Computer Society, Inc...

5. Chen, W., Da Silva, R. F., Deelman, E., & Sakellariou, R. (2013, October). Balanced task clustering in scientific workflows. In *2013 IEEE 9th International Conference on e-Science* (pp. 188-195). IEEE.
6. Bagheri, R., & Haghighat, A. T. (2015) .Dynamic clustering for Scientific Workflows with Load Balancing in Resource. ***I***nternational ***J***ournal of ***C***omputer ***S***cience and ***T***elecommunications.6 (8), 24-28.
7. Chen, W., & Deelman, E. (2012, June). Fault tolerant clustering in scientific workflows. In *2012 IEEE Eighth World Congress on Services* (pp. 9-16). IEEE.
8. Chen, W., da Silva, R. F., Deelman, E., & Fahringer, T. (2015). Dynamic and fault-tolerant clustering for scientific workflows. *IEEE Transactions on Cloud Computing*, *4*(1), 49-62.
9. Chen, W., da Silva, R. F., Deelman, E., & Sakellariou, R. (2015). Using imbalance metrics to optimize task clustering in scientific workflow executions. *Future Generation Computer Systems*, *46*, 69-84.
10. Dharwadkar, N. V., Poojara, S. R., & Kadam, P. M. (2018). Fault Tolerant and Optimal Task Clustering for Scientific Workflow in Cloud. *International Journal of Cloud Applications and Computing (IJCAC)*, *8*(3), 1-19.
11. Prathiba, S., Sowvarnica, S., Latha, B., & Sumathi, G. (2017, January). A Comparative Study of Task and Fault Tolerance Clustering Techniques for Scientific Workflow Applications in Cloud Platform. In *International Conference on Data Science Analytics and Applications* (pp. 1-7). Springer, Singapore.
12. Latiff, M. S. A., Madni, S. H. H., & Abdullahi, M. (2018). Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. *Neural Computing and Applications*, *29*(1), 279-293.
13. Bala, A., & Chana, I. (2015). Intelligent failure prediction models for scientific workflows. *Expert Systems with Applications*, *42*(3), 980-989.
14. Zhou, Z., Cheng, Z., Zhang, L. J., Gaaloul, W., & Ning, K. (2016). Scientific workflow clustering and recommendation leveraging layer hierarchical analysis. *IEEE Transactions on Services Computing*, *11*(1), 169-183.
15. sahni, J., & Vidyarthi, D. P. (2016). Workflow-and-Platform Aware task clustering for scientific workflow execution in Cloud environment. *Future Generation Computer Systems*, *64*, 61-74.
16. Berriman, G. B., Deelman, E., Good, J. C., Jacob, J. C., Katz, D. S., Kesselman, C., ... & Su, M. H. (2004, September). Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *Optimizing Scientific Return for Astronomy through Information Technologies* (Vol. 5493, pp. 221-233). International Society for Optics and Photonics.[m]
17. Graves, R., Jordan, T. H., Callaghan, S., Deelman, E., Field, E., Juve, G., ... & Okaya, D. (2011). CyberShake: A physics-based seismic hazard model for southern California. *Pure and Applied Geophysics*, *168*(3-4), 367-381.
18. Brown, D. A., Brady, P. R., Dietz, A., Cao, J., Johnson, B., & McNabb, J. (2007). A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. In *Workflows for e-Science* (pp. 39-59). Springer, London.
19. SIPHT. http://pegasus.isi.edu/applications/sipht.
20. USC Epigenome Center. http://epigenome.usc.edu.