International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:4, 2015

Scheduling Multiple Workflow Using De-De Dodging Algorithm and PBD Algorithm in Cloud: Detailed Study

B. Arun Kumar, T. Ravichandran

***Abstract*—**Workflow scheduling is an important part of cloud computing and based on different criteria it decides cost, execution time, and performances. A cloud workflow system is a platform service facilitating automation of distributed applications based on new cloud infrastructure. An aspect which differentiates cloud workflow system from others is market-oriented business model, an innovation which challenges conventional workflow scheduling strategies. Time and Cost optimization algorithm for scheduling Hybrid Clouds (TCHC) algorithm decides which resource should be chartered from public providers is combined with a new De-De algorithm considering that every instance of single and multiple workflows work without deadlocks. To offset this, two new concepts - De-De Dodging Algorithm and Priority Based Decisive Algorithm - combine with conventional deadlock avoidance issues by proposing one algorithm that maximizes active (not just allocated) resource use and reduces Makespan.

***Keywords*—**Workflow Scheduling, cloud workflow, TCHC algorithm, De-De Dodging Algorithm, Priority Based Decisive Algorithm (PBD), Makespan.

I. Introduction

THE user accesses computing resources, in a cloud, as general utilities that are leased and re-leased [1]. The benefits to cloud users are avoidance of up-front investment, lower operating cost, reduced maintenance cost, and scalability on demand. These features ensure elasticity to a user's computing environment, adapting the computer system to user needs.

Virtualization [2] presents a logical grouping/subset of computing resources accessed in abstract ways with benefits over original configuration. Virtualization software abstracts hardware by creating an interface to Virtual Machines (VMs), representing virtualized resources like CPUs, network connections, physical memory, and peripherals. A VM is an isolated execution environment independent of others. A VM has its own operating system, applications, and network services. Virtualization allows server consolidation, hardware normalization, and application isolation in same machine.

Workflow systems are a vehicle for efficient scientific applications development. Such systems benefit from resource provisioning technology from cloud computing. Workflows are usually organized as Directed Acyclic Graph (DAG), where constituent jobs (nodes) are either controlled or are data dependent (edges). Control-flow dependency specifies that a job should be completed before others start their process. In contrast, dataflow dependency specifies a job cannot start till all input data (created by earlier completed jobs) is available [3]. Control-flow is a common abstraction to reason about relationship between different jobs but shows how dataflow information is valuable to effectively use storage. A workflow-based workload has many workflow instances. A workflow instance is data-independent as they compute with differing inputs/parameters [4]. Also, workflows are designed, assembled, validated, and analyzed collaboratively. Workflows are shared similar to data collections, and compute resources are shared today by communities. The analysis of workflows necessitates substantial computational and data resources, which generate required results [5]. So, to offset this, Cloud computing is designed to ensure on-demand resources to users, to provide locally available computational power, delivering new computing resources when needed.

The remainder of this paper is organized as follows. Section II reviews several related works. Section III presents the De-De-Dodging Algorithm and PBD based workflow scheduling. Section IV shows experimental evaluation of heuristic and discusses the result. Section V concludes the paper.

II. Literature Survey

Important workflow scheduling strategies which bring out a survey of such strategies in cloud computing including their detailed classification was focused on by [6]. Then a comparative analysis of studied approaches was made.

Priority based Genetic Algorithm BCHGA to schedule workflow applications to cloud resources optimizing total workflow cost within a user`s specified budget was presented by [7]. A workflow`s task is assigned priority using bottom level (b-level) and top level (t-level). To increase population diversity, priorities create initial BCHGA population. The new algorithm is simulated in Java and evaluated with synthetic workflows based on realistic workflows from various areas considering cloud service provider's pricing model like that of Amazon. Simulation shows that the new algorithm promises performance compared to Standard Genetic Algorithm (SGA).

New task schedulers for clouds, achieving energy savings was presented by [8]. Task dependency leads to low cloud utilization. The above mentioned task is presented to address this. Results comparing the new schedulers with current ones show it is possible to get energy savings up to 22.7%, with no

B. Arun Kumar is Research Scholar, Karpagam University, Tamilnadu, India (e-mail: arunkumar.b.karpagam@gmail.com).
T. Ravichandran is Principal, Hindusthan Institute of Technology, Tamilnadu India.

makespan penalty. It is also seen that scheduling efficiency depends on how tasks are interconnected in workflows.

A workflow scheduling framework that efficiently schedules series workflows with many objectives in a cloud system was proposed by [9]. A meta-heuristic method called Artificial Bee Colony (ABC) creates an optimized scheduling plan. The framework allows setting multiple objectives. Conflicts among objectives are resolved using Pareto technique. Experiments investigate performance comparing algorithms used often in cloud scheduling. Results show that the new method reduces 57% cost and 50% scheduling time in a similar Makespan of HEFT/LOSS for a scientific workflow like Chimera-2.

Cloud computing introduction, workflow basics, and scheduling were described by [10] where scheduling algorithms used in workflow management considered the algorithms, types and tools used.

An Adaptive Hybrid Heuristic for user constrained data-analytics workflow scheduling in hybrid Cloud environment integrating the dynamic nature of heuristic based approaches and workflow-level optimization ability of meta-heuristic based approaches was proposed by [11]. The new approach's effectiveness is illustrated through a comprehensive case study compared to other techniques.

A strategy to schedule service workflows in a hybrid cloud proposed by [12] determines which services use paid resources and what resources should be requested to the cloud to minimize cost and meet deadlines. Experiments suggest that strategy decreases execution costs and ensures reasonable execution times.

Implementation of workflow scheduling to reduce overall jobs execution time in a workflow was focused on by [13]. The new scheme was evaluated using simulation based analysis on WorkflowSim.

A delay-constrained optimization problem to increase resource use and a two-step workflow scheduling algorithm to reduce cloud overhead in a user-specified execution time bound was formulated by [14]. Simulation shows that the new approach consistently achieved lower computing overhead and higher resource use than current methods in execution time bound. It reduced total execution time greatly by strategically choosing appropriate mapping nodes for prioritized modules.

A Multiple QoS constrained scheduling strategy of Multi-Workflows (MQMW) was introduced by [15]. The strategy schedules multiple workflows started at any time and QoS requirements are considered. Experiments show the strategy increasing scheduling success rate significantly.

III. Methodology

Users currently don't want to be stuck to own cloud providers to execute and schedule multiple workflows. Many organizations have their private cloud, but when extra resources are needed they opt for public cloud where they outline their use. In dependent workflow scheduling, switching between private and public cloud resources increases execution time and cost. Multiple requests for resources result in increased bandwidth. In this section two different Scheduling algorithms, De-De Dodging and Priority Based Decisive (PBD) algorithm, are proposed.

A. Time and Cost optimization for Hybrid Clouds (TCHC) Algorithm

TCHC algorithm decides task scheduling to public cloud, by determining dependencies among workflows. The algorithm decides best split between private and public cloud determining a task with least cost and execution time to be scheduled in public clouds. Cost is reduced by choosing minimum bandwidth in a public cloud.

The algorithm's 2 steps are task selection to reschedule and resources selection from a public cloud to create a hybrid cloud. The former decides which tasks can have reduced execution time using powerful resources from a public cloud; the latter determines execution time and costs involved in a new schedule. When the task is selected, initial scheduling involves available resources verification in a private cloud. When resources are available for a task, scheduler schedules tasks inside a private cloud. The algorithm checks whether a private resource pool (J) is less than deadline (Z) the loop continues till all tasks (T) are completed. Inside a loop, a node is selected from task set with highest priority and its Predetermined Start Time (PST) and Predetermined Finish time (PFT) are calculated with dependency ratio. Next, priority is set to next highest PST. Finally, all nodes are added, and resources allocated to each set of task. But, if resources are not enough, rescheduling requests the public resource pool. The algorithm now checks whether PFT is greater than Application Time Remaining (ATR). If the answer is yes then it calculates execution time and cost for extra resources, based on Path Clustering Heuristic (PCH) algorithm, or else it goes to private resource pool itself. Next if Pending Task (PT) is more than available Public Resource Pool, then tasks are queued for execution. Each task is selected and cost and execution time is premeditated for new resource. When the CT value is less than the resource in private pool, it is continued, or else algorithm requests from the public pool again. Finally, the algorithm schedules resource with least PFT and task continues.

B. DE-DE Algorithm Description

A cloud system receives many requests for a set of resource to complete a job. Jobs are termed workflows. Each workflow consists of a set of tasks which are dependent on others by some means. This study uses the De-De algorithm which considers a set of workflows and detects whether deadlocks occur between them using the banker's algorithm. The detailed logical flow is shown:

```
Workflows F: {f1, f2, f3…fn}
Deadline E
Resource H
Predestined Start Value PSV
Predestined Finish Value PFV
Public resource pool FB
Private Resource Pool G
Rescheduling group N
Priority Pr
```

Pending task PT
Application Remaining Time ART
Node set NS
Time & Cost value TCV
Job J with the instance i
Instance of workflows to be scheduled, Ii
Time taken for completion of a job, time ( )
Temporary variables Wi and Ri
Storage request for the job getWriteSet ()
Storage allocation of the job getReadSet ( )
Need of i resources in time t alloc (i, t) / need (i, t)
System safety check safetycheck ( )
Deadlock Dependency Detection Algorithm (De-De)

The detailed flow of the De-De Algorithm is as:

```
1) F= Set of Workflows{ F=Workflows==set of tasks TS==single task T}
2) function De-De ( Ii , F)
3) R i← getReadSet ();
4) J ← J − (|Wi| − |Ri|);
5) alloc (i, t) ← alloc (i, t) + (|Wi| − |Ri |);
6) need (i, t) ← need (i, t) − |Wi|
7) if (safetycheck (Ii))
8) J ← J − |Ri |;
9) alloc (i, t) ← alloc(i, t) + | Ri |;
10) return true;
11) goto line 19;
12) else
13) J ← J + (|Wi | − | Ri |);
14) alloc (i, t) ← alloc(i, t) − (|Wi | − |Ri |);
15) need (i, t) ← need(i, t) + | Wi |;
16) return false;
17) goto line 54
18) End function
19) Perform initial schedule
20) Dependency De=0-5
21) For each W in TW
22) For each T in TS do
23) If T < De Do
24) If (H Є G) then
25) Schedule F in G
26) While (time(F) > E && iteration =F) do
27) Select node from NS with ↑Pr
28) If ni Э NS then
29) Add ni to NS
30) Iteration=iteration+1
31) End while
32) Schedule the H with ↓ PFV
33) De-De ( Ii, H);
34) else select next task from TS
35) else select next workflow from WT
36) Else
37) Wi← getWriteSet ();
38) While (| Wi | > G && iteration =F ) do
39) Request for H in FB
40) If PFV > ART then
41) Queue PT to execute
42) For each W in TW
43) For each T in TS do
44) If T < De Do
45) Select H Є FB then
46) Calculate TCV for new H
47) If TCV < ( H Є G ) then
48) Add H to FB
49) else select next task from TS
50) else select next workflow from WT
51) Schedule H with ↓ PFV
52) De-De (Ii,H);
53) End while
54) End else
```

The algorithm's first line initializes a set of workflows consisting of set of tasks T to a variable F. The Function De-De algorithm is defined clearly including some parameters associated with instance Ii i.e., r (t), alloc (i, t) and need (i, t)) are updated accordingly. In the third line function, De-De is given where Ri is assigned with allocated workflow resources. In variable G, remaining resource is calculated by subtracting available resource in a private pool with already allocated and requested resources. De-De algorithm checks if current available storage is sufficient to satisfy job request (obtained via getWriteSet ()). If not, job requests from a public resource pool. In line seven, safety check algorithm verifies whether the system is in a safe state for each workflow. Once verified, line 19 is called if it returns true. In 19th line, initial scheduling is done which considers only Private resource pool and schedules the workflows in a Private resource pool based on attributes like communication cost, priority, and time, resource allocation. A range is assigned for dependency; for instance: dependency De value is between 0 - 5. The 23rd line checks range and if dependency value is less than range, allocation/request to resource is done. Next the algorithm checks whether available resources are enough. If sufficient to finish the job, workflow is requested in a private cloud or a public cloud. Once scheduled, workflows in private cloud run the task inside the private cloud till the deadline is met. The iteration is repeated till deadline E is met, where the algorithm continues by choosing a node Ni from node set NS with highest priority. Then safety check is algorithm is sought.

If it returns true then system is safe state, or it is said to wait, and next workflow considered. Simultaneously if resource is not enough in a private pool, it is requested in a public pool as in line 39. Line 40 in algorithm verifies whether Predestined Finish value (PFV) is greater than ART, then queues tasks to execute. Again dependency range is checked for new, and if dependency value is less than range, allocation/request to resource is done. Line 46 evaluates the new TCV for new resource allocation. When value of TCV is less than available resource in a private cloud, then public cloud is requested. As TCV is considered less than old TCV resource is added to set NS. Now schedule resource with lowest PFV, suppose TCV value is larger, then verify inside private cloud. De-De algorithm is invoked in line 41 to check safety and if it returns true allocate resource with lowest PFV. Finally, the new algorithm is well furnished to bind between selecting public and private cloud and allocating requested resources to a specific workflow with low cost/time and without any deadlocks and dependencies between them.

C. Priority Based Decisive Algorithm (PBD)

The current algorithmic program's drawback is that computation time is incredibly low for truthful policies, and deadlock avoidance is not considered. To offset this in the new work, there is a tendency to square measure enhancing 3 proposals.

As primary proposal, dependency is checked among workflow tasks to avoid deadlocks. Tasks are in queue with their priority basis. PBD finds a critical path in a workflow and also performs initial scheduling. The dependency of obtained path and execution time is checked, and if a schedule is feasible it is scheduled in a passive state. If not feasible then dependency and execution time are resolved where multiple requests are verified. The scheduled task is taken, and execution cost is discussed to find whether they are completed within estimated cost. If yes, then they continue recursively till all tasks in workflow are scheduled in active state.

Quantifying planning performance and allocation policy on a Cloud infrastructure (hardware, software, services) for various application and repair models below variable load, energy performance, and system size is tough to tackle. To change this, a CloudSim is proposed as a final proposal: a new generalized/protractible simulation framework permitting seamless modeling, simulation and experimentation of rising Cloud computing infrastructures and management services. Fig. 1 shows the algorithm description of PBD. The variables used are

W- Set of Workflows
T- Set of Task
Pr- Priority
D- Deadline
De- Dependency
TQ- Task Queue
Pj- Time Duration
ES- Early Start
EC- Early Complete

IV. Experimental Results

Cloudsim simulator was used to measure the performance of the proposed algorithms. Two data centers were used. Two experiments were conducted with 3 VM chosen randomly from the two data centers and in the second scenario 6 VM chosen form these two data centers. Makespan, cost, CPU Time in ms and resource utilization using TCHC, De-De, and PBD were computed. Tables I-IV show the result of performance metrics mentioned above sentence.

TABLE I
Makespan When Three VM Were Used

Number of tasks	TCHC	De-De	PBD
200	40108	38592	37893
400	81932	79835	77910
600	164549	160978	157803
800	333382	323722	317745
1000	675256	653804	637831

Table I shows the Makespan of PBD performs better than TCHC in the range of 4.18% to 5.7% and better than De-De in the range of 1.82% to 2.47%.

Table II shows the Cost of PBD performs better than TCHC in the range of 4.2% to 6.18% and better than De-De in the range of 2.44% to 3.84%.

TABLE II
Cost When Three VM Were Used

Number of tasks	TCHC	De-De	PBD
200	13676	13381	13058
400	27688	27247	26549
600	55996	54139	52740
800	114028	111525	107575
1000	229933	224604	216139

TABLE III
CPU Time in MS for Number of Resources Used Is Three

Number of tasks	TCHC	De-De	PBD
200	30214	29272	28656
400	61333	59084	56883
600	124233	119672	117269
800	249369	241404	234500
1000	505645	489819	473267

Table III shows the CPU Time of PBD performs better than TCHC in the range of 5.2% to 7.5% and better than De-De in the range of 2.02% to 3.79%. Fig. 2 shows the resource utilization of PBD performs better than TCHC in the range of 4.39% to 5.6% and better than De-De in the range of 2.17% to 3.1%.

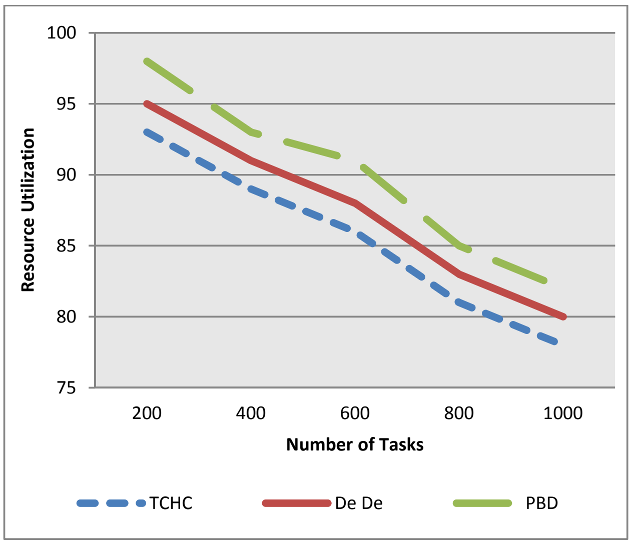

Fig. 2 Resource Utilization when number of resources is three

Table IV shows the makespan obtained when the number of resources used is six.

TABLE IV
Makespan for Number of VM=6

Number of tasks	TCHC	De-De	PBD
200	21118	20396	19759
400	42011	41314	40406
600	85755	83807	81258
800	173950	170157	165978
1000	351642	343392	335858

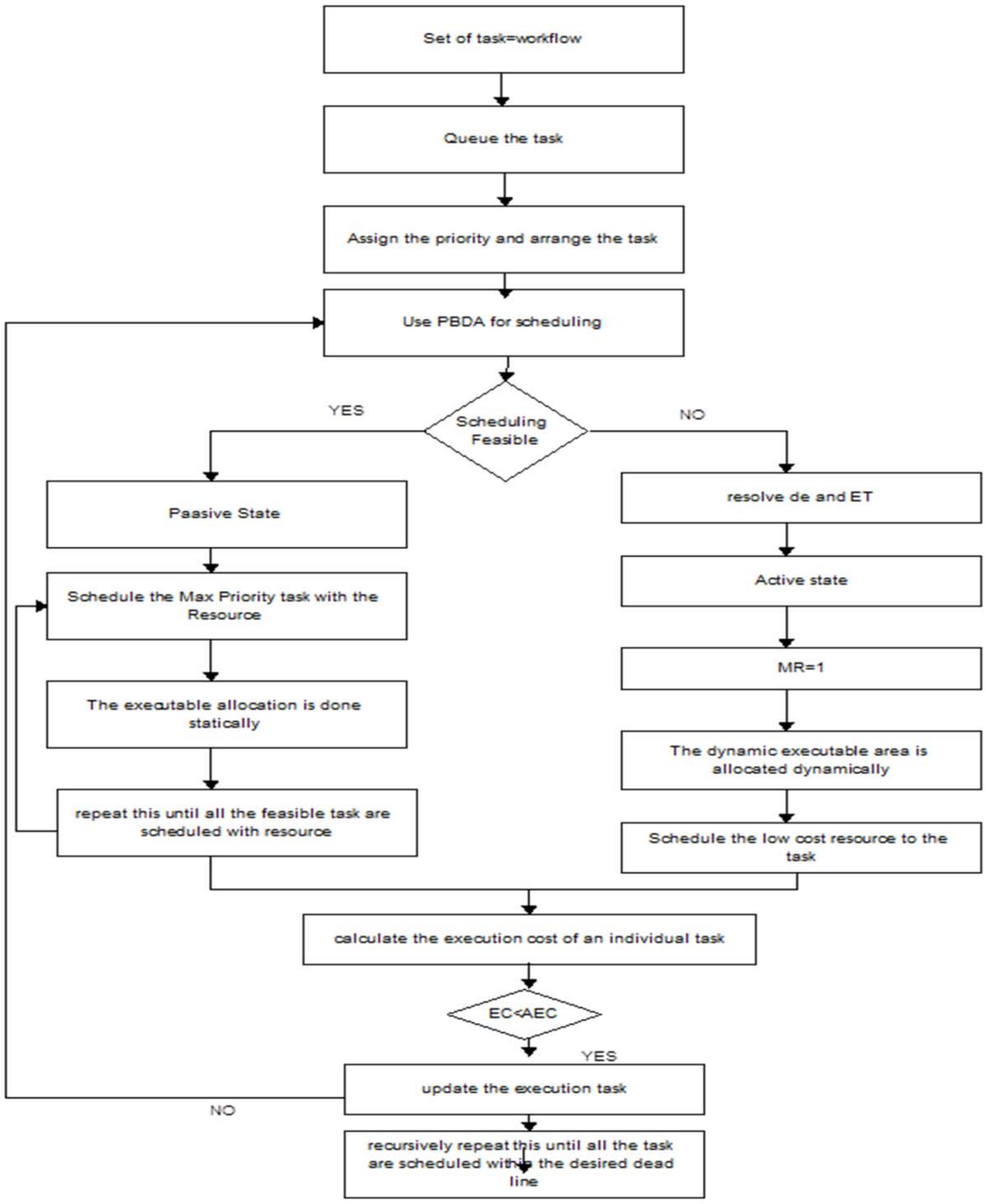
Set of task=workflow
Queue the task
Assign the priority and arrange the task
Use PBDA for scheduling
YES
Scheduling Feasible
NO
Passive State
Schedule the Max Priority task with the Resource
The executable allocation is done statically
repeat this until all the feasible task are scheduled with resource
resolve de and ET
Active state
MR=1
The dynamic executable area is allocated dynamically
Schedule the low cost resource to the task
calculate the execution cost of an individual task
EC<AEC
YES
update the execution task
NO
recursively repeat this until all the task are scheduled within the desired dead line

Fig. 1 Algorithm description for PBD

It is seen that Makespan of PBD performs better than TCHC in the range of 3.89% to 6.64% and better than De-De in the range of 2.21% to 3.17%. Table V shows the cost of resource utilization.

It is seen that cost of PBD performs better than TCHC in the range of 5.13% to 7.05% and better than De-De in the range of 1.89% to 3.7%. Table VI shows the CPU time in millisecond.

TABLE V
COST FOR NUMBER OF VM=6

Number of tasks	TCHC	De-De	PBD
200	7230	7104	6868
400	14353	13891	13375
600	28760	27995	27152
800	59817	57853	56765
1000	117468	114336	111102

TABLE VI
CPU TIME IN MS

Number of tasks	TCHC	De-De	PBD
200	15867	15556	14996
400	32177	31603	30536
600	64418	63395	61180
800	129384	125764	121553
1000	258350	249290	240651

The CPU Time of PBD performs better than TCHC in the range of 5.15% to 7.09% and better than De-De in the range of 3.4% to 3.66%. Fig. 3 shows the resource utilization.

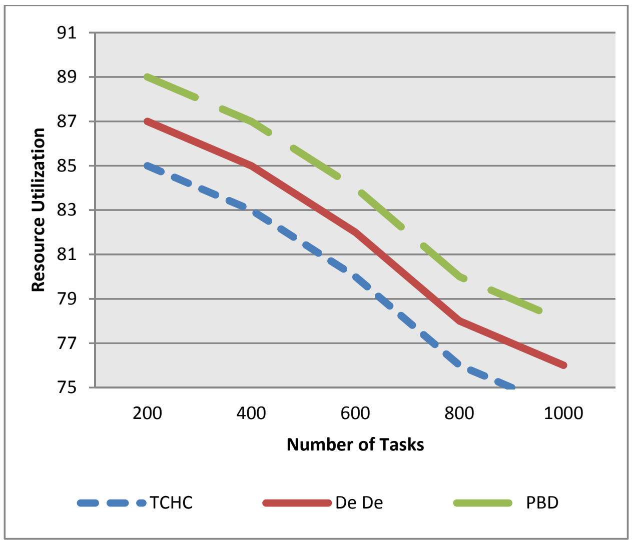

Fig. 3 Resource Utilization for number of VM=6

Resource Utilization of PBD performs better than TCHC in the range of 4.59% to 5.26% and better than De-De in the range of 2.27% to 2.59%.

V. CONCLUSION

A new algorithmic rule called Priority Based Decisive Algorithm (PBD) was proposed for SaaS Clouds that reduce execution price while meeting a user-defined execution time. Simulating the algorithmic rule with artificial workflows it was compared with other algorithms including TCHC and De-De. The algorithm's computation has shown good results in multiple workflows scheduling. Dependency among tasks pulls down optimization algorithms. PBD outperforms other algorithms when induced with inter-dependency among tasks in a workflow. PBD can enhance and schedule multiple workflows in hybrid cloud environments.

REFERENCES

[1] Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Services and Applications 1(1), 7–18 (2010).
[2] Smith, J.E., Nair, R.: The architecture of virtual machines. Computer 38(5), 32–38 (2005).
[3] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," Future Gener. Comput. Syst., vol. 25, no. 5, pp. 528– 540, May 2009.
[4] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Mayers, and M. Samidi, "Scheduling data-intensive workflows onto storage-constrained distributed resources," in Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, 2007, pp. 401–409.
[5] J. Bent, D. Thain, A. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Explicit control in a batch-aware distributed file system," in Proceedings of Networked Systems Design and Implementation (NSDI), San Francisco, California, USA, 2004, pp. 365–378.
[6] Fakhfakh, F., Kacem, H. H., & Kacem, A. H. (2014, September). Workflow Scheduling in Cloud Computing: A Survey. In Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International (pp. 372-378). IEEE.
[7] Verma, A., & Kaushal, S. (2013, September). Budget constrained priority based genetic algorithm for workflow scheduling in cloud. In Communication and Computing (ARTCom 2013), Fifth International Conference on Advances in Recent Technologies in (pp. 216-222). IET.
[8] Watanabe, E. N., Campos, P. P., Braghetto, K. R., & Batista, D. M. (2014, May). Energy Saving Algorithms for Workflow Scheduling in Cloud Computing. In Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on (pp. 9-16). IEEE.
[9] Udomkasemsub, O., Xiaorong, L., & Achalakul, T. (2012, May). A multiple-objective workflow scheduling framework for cloud data analytics. In Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on (pp. 391-398). IEEE.
[10] Arya, L. K., & Verma, A. (2014, March). Workflow scheduling algorithms in cloud environment-A survey. In Engineering and Computational Sciences (RAECS), 2014 Recent Advances in (pp. 1-4). IEEE.
[11] Rahman, M., Li, X., & Palit, H. (2011, May). Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment. In Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on (pp. 966-974). IEEE.
[12] Bittencourt, L. F., Senna, C. R., & Madeira, E. R. (2010, October). Scheduling service workflows for cost optimization in hybrid clouds. In Network and Service Management (CNSM), 2010 International Conference on (pp. 394-397). IEEE.
[13] Prakash, V., & Bala, A. (2014, July). A novel scheduling approach for workflow management in cloud computing. In Signal Propagation and Computer Technology (ICSPCT), 2014 International Conference on (pp. 610-615). IEEE.
[14] Zhu, M., Wu, Q., & Zhao, Y. (2012, December). A cost-effective scheduling algorithm for scientific workflows in clouds. In Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International (pp. 256-265). IEEE.
[15] Xu, M., Cui, L., Wang, H., & Bi, Y. (2009, August). A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on (pp. 629-634). IEEE.

B. Arun Kumar is Research Scholar, Karpagam University. He is currently pursuing his doctorate in India.

T. Ravichandran is Principal, Hindusthan Institute of Technology, India.