Improved Context Aware PSO Task Scheduling in Cloud Computing

B. Siva Rama Krishna
Research Scholar, Department of CSE, Dr. Y.S.R. ANU College of Engineering & Technology, Acharya Nagarjuna University, Nagarjuna Nagar, AP, India.
E-mail: sivaram6115@gmail.com

E. Sreenivasa Reddy
Professor, Department of CSE, Dr. Y.S.R. ANU College of Engineering & Technology, Acharya Nagarjuna University, Nagarjuna Nagar, AP, India.
E-mail: esreddy67@gmail.com

Received September 20, 2021; Accepted December 17, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V19I1/WEB19244

Abstract

One of the major advantages of switching to the clouds is the scalability ability of the applications. Contrary to the grids, the ability to scale the cloud resources allows their real-time provisioning so as to meet the application constraints Generation of optimal schedule for given set of tasks and machines. Different experiments show that although having an optimum solution is almost impossible but having a sub-optimal solution using heuristic algorithms seems possible. In this paper, we propose a Context Aware PSO Task Scheduling scheme to analyze various scenarios with different parameters in cloud computing system corresponding to APSO In each scenario, we change one parameter and keep other parameters constant.

Keywords

Cloud Computing, Task Scheduling, Particle Swarm Optimization, Nearest Neighbours.

Introduction

Resource Management comprises of various phases of workload and resources from submission to execution Resource management has 2 steps: i) Resource provisioning ii) Resource scheduling. Resource provisioning is the analysis where requirements by consumers based on QoS and check proper resources are given to workload where the resource scheduling continues the work of resource provisioning according to the resources selected by the consumer the workload are mapped accordingly (K. Bratanis et.

al, 2016). Key player in cloud is consumers and Provider, Providers allocates the resources according to the resources demanded by cloud consumer both players have different agenda, providers want more as earn much profit as possible with minimum investment many requests handled on one resource will lead to performance downgrade while the user wants minimum cost and minimum execution time service quality is maintained by rejecting the request which has indefinite result. Scheduling becomes hectic and information trading between is mostly not followed. Challenges in resource scheduling include dispersion, uncertainty which is not solved by RSA. Altering cloud environment properties is not enough. Consumers submits the workload is queued. Resources are assigned to the workflow according to the details provided. Workload is provided with demanded resources by resource provisioned, resource pool contains all the resources. If occurred shortage of resources on basis of QoS requirements the workload management system send a new request by informing SLA with new QoS requirements. Resources Scheduler is provided with workload just after provisioning of resources is completed successfully. In the next phase result are provided to the Workload Management System. Request provided by the cloud consumer on the basis of request scheduling policy is picked by the policy selector J. Wilkes and C. Reiss (2015). Cloud Environment also a scheduler that executes diverse planning strategies dependent upon the choice taken toward arrangement selector. In view of the planning policy, the resources need aid allocated of the cloud workloads.

1. Need of Resource Scheduling

A lot of factors like user needs, type of system etc have to be considered while designing a scheduling algorithm. Avoiding indefinite starvation i.e. a process must not wait indefinitely during the process service. Minimizing overhead as overhead causes wastage of the resources and if overhead is minimized, the overall performance of the system improves a lot. Enforcing priorities i.e. if a system assigns any priorities to the process, the algorithm must process the process with highest priority first. Achieving balance between response and utilization so that all the resources of the system are busy. Depending on the type of system, a user may expect the following things from the scheduler.

- Enhance the time and resource utilization parameter with reference to workloads.
- The amount of Resources of should be minimum for the workload to satisfy the Quality Level.
- To Minimize the Completion time for better Resource Scheduling.
- Allocate Suitable Workload to the Virtual Machines.

Related Work

1. Context Aware PSO (CA_PSO)

In CA_PSO, population is initialized by random selection from execution times of given tasks on available resources. The basics of the CA_PSO are discussed in this section.

$$\mathcal{P}_{best_i}(t+1) \left\{ \begin{array}{ll} \mathcal{X}_i(t) & \text{:if } f_t(\mathcal{X}_i(t+1)) < f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & \text{:if } f_t(\mathcal{X}_i(t+1)) > f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & \text{:if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \textit{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) > f_{\text{cost}}(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t) & \text{:if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \textit{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) < f_{\text{cost}}(\mathcal{X}_i(t)) \end{array} \right\} \quad (5)$$

$$\mathcal{G}_{best}(t+1) = \{\text{Max}\{\mathcal{P}_{best_1}(t+1), \mathcal{P}_{best_2}(t+1), \dots, \mathcal{P}_{best_i}(t+1)\}\} \quad (6)$$

$$f_t(\mathcal{X}_i) = \frac{1}{\mathcal{T}_i} \text{ where } \mathcal{T}_i = \text{Makespan}(\text{X}_\text{i}) \quad (7)$$

$$f_{\text{cost}}(\mathcal{X}_i) = \frac{1}{\text{Cost}_{(\mathcal{X}_i)}} \text{ where Cost}_{(\mathcal{X}_i)} = \text{Makecost}(\mathcal{X}_i) \quad (8)$$

$\mathcal{P}_{best_i}(t+1)$ and $\mathcal{G}_{best_i}(t+1)$ for i^{th} particle after $(t+1)$ iterations is computed using {Eqn. 5 and 6}. CA_PSO uses two fitness functions, which are respectively based on Makespan and Makecost. Fitness function f_t computes the maximum completion time taken by tasks (Makespan) and another fitness function f_{cost} computes the cost liability corresponding to a schedule or any particles current position (Makecost).If $\mathcal{X}_i$ is a matrix with dimensions $m \times t$ then $\mathcal{X}_i$ can be defined as in {Eqn.9}.

$$\left\{ \begin{array}{ll} \mathcal{X}_i(\ell, p) & \\ \frac{\text{LEN}_{\mathcal{T}_p}}{\text{MIPS}_{\mathcal{R}_\ell}} & \text{: Where MIPS}_{\mathcal{R}_\ell} \text{ is capacity of } \ell^{th} \text{Resource and LEN}_{\mathcal{T}_p} \text{ is length of } p^{th} \text{ tasks in MIPS} \\ 0 & \text{: Otherwise} \end{array} \right\} \quad (9)$$

2. Improved Context Aware PSO (CA_PSO)

In an effort to improve the performance credentials of CA_PSO, a VMs capacity and heterogeneity oriented variant of CA_PSO with Nearest Neighbor (NN) has been proposed.

3. Nearest Neighbor (NN)

A novel heuristic is proposed in this section which is based on the concept of execution time characteristics of heterogeneous VMs. New task is assigned to a free VM only if next allocation reduces the collective variance of execution time of all tasks allocated (including completed tasks) to this machine. Below table illustrates the working of proposed Nearest Neighbor (NN) heuristic. The key concept is to ensure that the selection of next task for scheduling reduces the variance by maximum value or increases the variance by minimum value.

```
Algorithm: Nearest_Neighbor ($\mathcal{M},\mathcal{T}$)
$\mathcal{M}$ : Set of VMs available for computation and execution; Number of VMs are $m$ .
$\mathcal{T}$ : Set of tasks for execution; Number of tasks are $t$ .
Step 1: Start
Step 2: Compute Load Matrix $\mathcal{L}(m*t)$ from $\mathcal{M}$ and $\mathcal{T}$ .
Step 3: Compute mean execution times MEM($m*1$) for each machine using $\mathcal{L}(m*t)$.
Step 4: Initialize $S = \text{Zeros}(m,t)$ // Zero Matrix
Step 5: For $\ell$=1 to $t$ do
    • Identify the Machine id $\mathcal{M}_i$ which is free and can be considered for next task.
    • Identify the Tasks id $\mathcal{T}_j$ which if allotted to $\mathcal{M}_i$ results in minimum increase in
      variance of execution times of completed and newly submitted task $\mathcal{T}_j$ and
      name this task as Nearest Neighbor Task.
    • Set $S(i, j)$=$\mathcal{L}(i, j)$
Step 6: Print values of S as output schedule.
Step 7: Stop
```

4. Nearest Neighbour Context Aware PSO (NNCA_PSO)

In CA_PSO, a modified Cost Aware variant of PSO is considered for generation of optimal schedule for given set of tasks and machines. A Hybrid variant of CA_PSO has been proposed in this Section. CA_PSO assumed to have two objective functions for optimizing time and cost. If global and local best values could not be improved by using time based optimization then cost based optimization is applied. The sequence of application of objective functions in NNCA_PSO has been maintained in the same way as proposed in CA_PSO. In NNCA_PSO, the initial population and population after iterations is treated with NN heuristic resulting in better selection of global and local best.

```
**Algorithm:** NNCA_PSO($\mathcal{M},\mathcal{T}$)
$\mathcal{M}$ : Set of VMs available for computation and execution; Number of VMs are $m$.
$\mathcal{T}$ : Set of tasks for execution; Number of tasks are $t$.

Step 1: Start
Step 2: Obtain schedule(s) using Nearest_Neighbor ($\mathcal{M},\mathcal{T}$)
Step 3: Initialize Population using schedules obtained from Nearest_Neighbor ($\mathcal{M},\mathcal{T}$)
Step 4: Fix number of Iteration (niter)
Step 5: Define two objective functions based on Makespan $f_t$ and Makecost $f_{\text{cost}}$.
Step 6: Apply Objective Function on Particles and obtain Fitness Value of each particle.
Step 7: Set $\mathcal{P}_{best_i}(0)$ for each particle $i$ by using
        $\mathcal{P}_{best_i}(t+1)$
        $\begin{Bmatrix} \mathcal{X}_i(t) & :\text{if } f_t(\mathcal{X}_i(t+1)) < f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & :\text{if } f_t(\mathcal{X}_i(t+1)) > f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & :\text{if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \text{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) > f_{\text{cost}}(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t) & :\text{if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \text{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) < f_{\text{cost}}(\mathcal{X}_i(t)) \end{Bmatrix}$ and
    Set $\mathcal{G}_{best}(0) = \mathcal{G}_{best}(t+1) = \{\text{Max}\{\mathcal{P}_{best_1}(t+1), \mathcal{P}_{best_2}(t+1), ..., \mathcal{P}_{best_i}(t+1)\}\}$
Step 8: For $i = 1$ to niter do
    Update location of each particle using
        $\mathcal{V}_i(t+1) = \mathcal{V}_i(t)w + C_1 \times \text{rand}() \times (\mathcal{P}_{best_i}(t) - \mathcal{X}_i(t)) + C_2 \times \text{rand}() \times (\mathcal{G}_{best}(t) - \mathcal{X}_i(t))$
        $\mathcal{X}_i(t+1) = \mathcal{X}_i(t) + \mathcal{V}_i(t+1)$
    Evaluate fitness values of each particle using
        $f_t(\mathcal{X}_i) = \frac{1}{\mathcal{T}_i}$ where $\mathcal{T}_i = \text{Makespan}(X_i)$
        $f_{\text{cost}}(\mathcal{X}_i) = \frac{1}{\text{Cost}_{(\mathcal{X}_i)}}$ where $\text{Cost}_{(\mathcal{X}_i)} = \text{Makecost}(\mathcal{X}_i)$
    Update $\mathcal{P}_{best_i}(t)$ for each particle by using
        $\mathcal{P}_{best_i}(t+1)$
        $\begin{Bmatrix} \mathcal{X}_i(t) & :\text{if } f_t(\mathcal{X}_i(t+1)) < f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & :\text{if } f_t(\mathcal{X}_i(t+1)) > f_t(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t+1) & :\text{if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \text{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) > f_{\text{cost}}(\mathcal{X}_i(t)) \\ \mathcal{X}_i(t) & :\text{if } f_t(\mathcal{X}_i(t+1)) = f_t(\mathcal{X}_i(t)) \text{ and } f_{\text{cost}}(\mathcal{X}_i(t+1)) < f_{\text{cost}}(\mathcal{X}_i(t)) \end{Bmatrix}$
    Update $\mathcal{G}_{best}(t)$ for current iteration by using
        $\mathcal{G}_{best}(t+1) = \{\text{Max}\{\mathcal{P}_{best_1}(t+1), \mathcal{P}_{best_2}(t+1), ..., \mathcal{P}_{best_i}(t+1)\}\}$
Step9: Print values of $\mathcal{G}_{best}$ and the corresponding particle.
Step10: Stop
```

Results and Analysis

Here this work analyzes various scenarios with different parameters in cloud computing system corresponding to APSO algorithm as described above. In each scenario, we change one parameter and keep other parameters constant. On the basis of behavior of the system, we analyze it's performance. The virtual machine configuration, task and host are initially taken as given below:

VM Parameters	Task (cloudlet) Parameters	Host Parameters

long size = 10000; //image size (MB)
int RAM = 512; //VM memory (MB)
int MIPS = 1000;
long BW = 1000;
int pes_Number = 1; //number of CPUs
	long length = 1000;
	long file Size = 300;
	long output Size = 300;
	int pes_Number = 1;
		int RAM = 4096; //host memory (MB)
		long storage = 1000000; //host storage
		int BW = 10000;
		CPUs/ Cores=Quad core and dual core

Datacenter: Contain 2–Host with One Quad Core and One Dual Core (Total P.E. = 6).
Virtual Machine (VM) =20
Cloudlet (Task) = 40(not divisible)
Scheduling: Space Shared and Time Shared.

VM Allocation Policy: this policy chooses a host for a VM with fewer processing elements (PEs) in use. This allocation policy does not perform any optimization of the VM allocation.

CloudSim is a framework for modeling and simulation of cloud computing infrastructure and services. In CloudSim scheduling perform at two levels. First it is implemented between Hosts and Virtual Machines (VM) and then implemented between Virtual Machines and cloudlets. In Space-Shared scheduling a processing element can be allotted to new virtual machine. While in Time-Shared scheduling, task can be shared to the processing elements for execution.

Scenario–1: Virtual Machine Aware

In this scenario the parameters used are Datacenter= 2 and each Datacenter contain two hosts in which one host is quad-core and other is dual-core. Number of tasks is 40, VM's Allocation Policy is space shared and changing the number of virtual machines (VM) metric of result is Execution Time (ET) value of tasks (in milliseconds) taken by three different methods is shown in table-6.

Table 1 Virtual Machine Aware Methods

Algorithm	No. of Virtual Machines (VMs)						
	2	5	6	8	10	15	20
PSO	20.4	8.2	7.0	5.0	4.2	3.8	3.8
APSO	20.0	7.8	6.8	4.9	3.6	3.2	3.2
NNCA_PSO	20.1	7.8	6.8	4.7	3.5	3.2	3.1

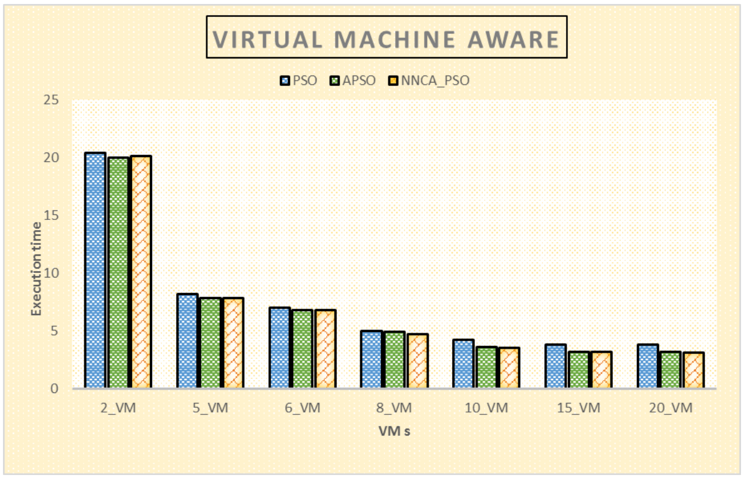

Fig. 1 Execution time as per no. of virtual machines

Table-1 and Fig-1 shows that, as the number of virtual machines increase the execution time decrease and it become constant after fixed number of virtual machines. Each virtual machines use only one PE from each host and only 16 VMs is created as per parameter used. Hence, when number of VMs becomes greater than 16, there is no change in execution time. The APSO and NNCA_PSO approaches exhibit the better result as compare to PSO.

Scenario-2: Virtual Machine with Number of PEs Aware

In this the number of VM's kept constant (i.e. VMs=20) and rest data is as in scenario–1. Now changing the number of PEs per VM, the resultant ET value of the tasks taken by three different methods is shown in Table–2. Processing elements (PEs).

Table 2 Virtual machine with number of PE's aware

Algorithm	No. of Processing Elements (PEs)					
	1	2	3	4	5	6
PSO	4.4	4.6	6.6	6.6	N.E	N.E
APSO	4.0	4.4	6.2	6.2	N.E	N.E
NNCA_PSO	4.0	4.4	6.2	6.2	N.E	N.E

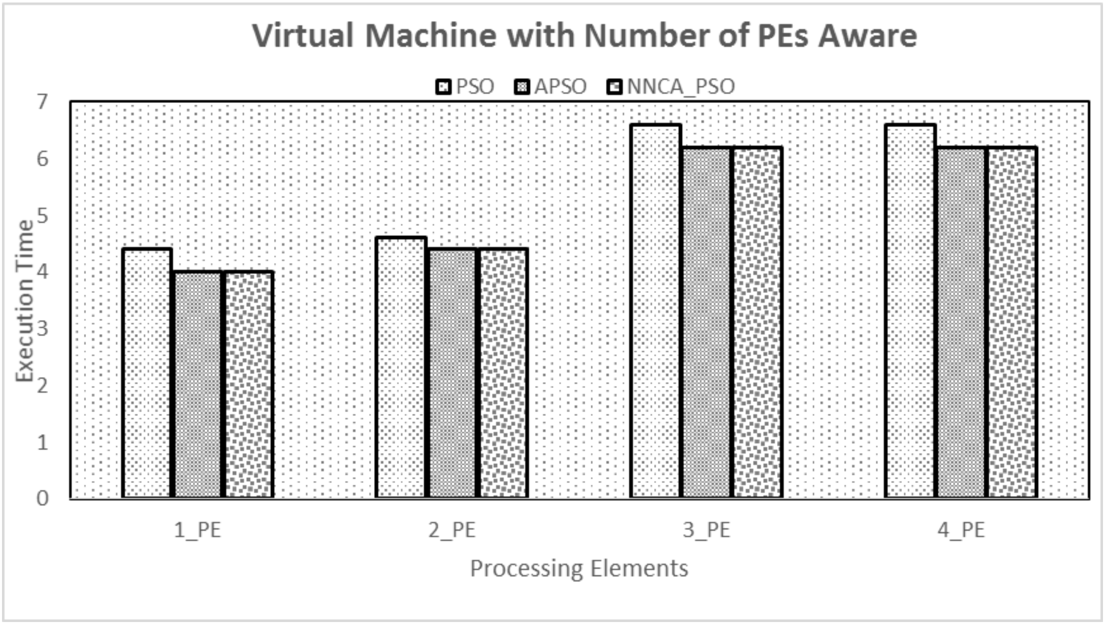

Fig. 2 Execution time as per no. of P.E per host

Table-2 and Fig.2 shows that as we increase the number of PEs per VMs the execution time also increases and after a fixed number of PEs the task is not executed. Since the task is dependent so the task will execute one by one. The result obtained in APSO-1 and NNCA_PSO-2 exhibit the better as compare to PSO.

Scenario–3: Host Bandwidth Aware

In this scenario parameter used are #Datacenter= 2, #VM=20, #Task =4000, Space Shared/time Shared, changing bandwidth (BW) of host by keeping the VM bandwidth constant the resultant ET value of the tasks taken by three different methods is shown in Table 3.

Table 3 Host bandwidth aware

Algorithm	**Host Bandwidth (BW)**						
	800	**1000**	**2000**	**3000**	**4000**	**5000**	**10000**
PSO	NE	488.2	480.2	446.2	348.2	344.8	356.6
APSO	NE	400.5	360.8	360.8	330.8	330.8	332.6
NNCA_PSO	NE	380.5	366.8	360.6	350.8	350.8	334.2

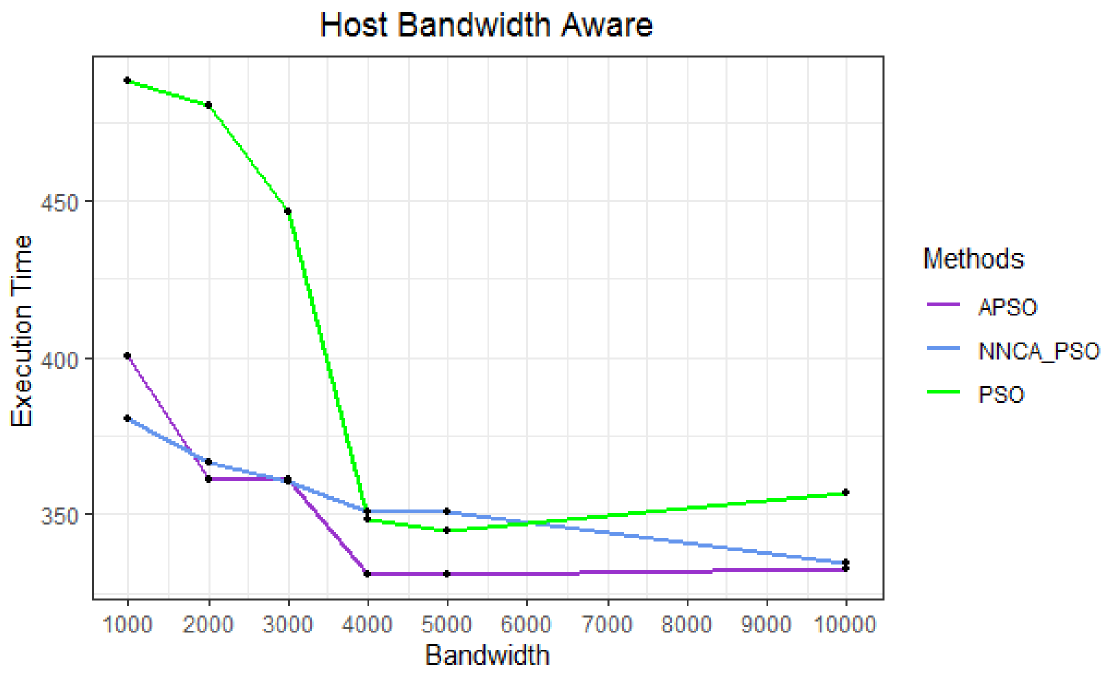

Fig. 3 Host bandwidth aware

Table 3 and Fig.3 shows that as we increase the bandwidth the execution time decrease and become constant after some fixed amount of bandwidth (when its bandwidth match with computational frequency). The Task is not executed when required bandwidth is not full–filled by data-center. APSO and NNCA_PSO exhibit the better result as compare to PSO.

Scenario–4: VM’s Bandwidth Aware

In this scenario the parameters are #Datacenter=2, #VM=20, #Task=4000, Space Shared/Time Shared, changing the bandwidth of VM by keeping the host bandwidth constant. The resultant ET value the tasks taken by three different methods is shown in Table 4.

Table 4 VM’s bandwidth aware

Algorithm	**VM’s Bandwidth (BW)**						
	800	**1000**	**2000**	**3000**	**5000**	**10000**	**12000**
PSO	348.8	343.2	333.0	330.2	330.2	330.2	330.2
APSO	348.8	343.2	333.0	330.2	330.2	330.2	330.2
NNCA_PSO	348.8	343.2	333.0	330.2	330.2	330.2	330.2

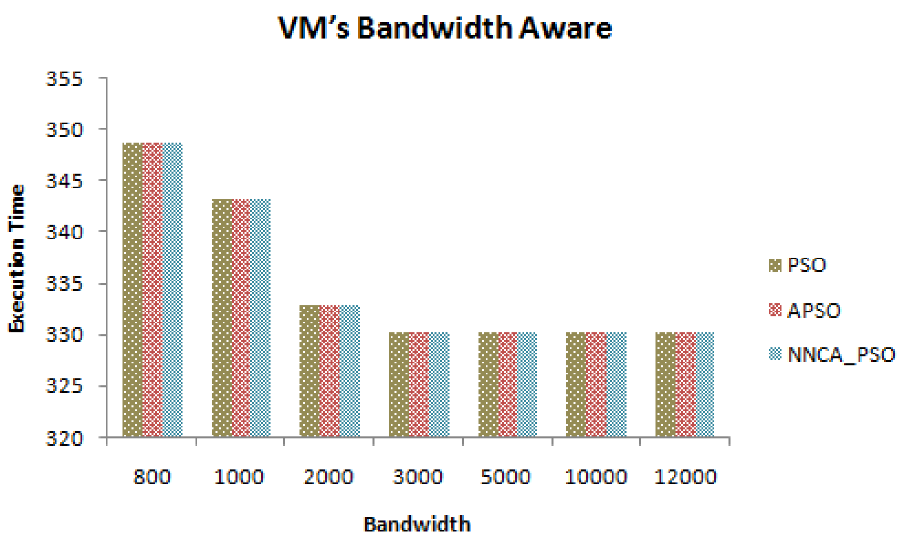

Fig. 4 Execution time as per VM bandwidth

Table 4 and Fig.4 shows that as we increase the bandwidth up to a fixed amount as per computational capability of VM's, number of VM's and bandwidth supply by data-center, the Execution time decrease and finally become constant after a fixed amount of bandwidth has shown above. Here APSO and NNCA_PSO exhibit the same result compare to PSO.

Scenario–5: Data Center Aware

In this there is 4 data-centers, each data-center contains 2 hosts with quad core system, #task=400). In second we considered 2 data-centers, each data-center contains 2 hosts with quad core, #task=400. In both cases VMs are variable and metric of result is Execution Time (ET) of task taken by three different methods is given in Table 5(a) and Table 5(b).

Table 5(a) Datacenter aware

Algorithm	**Virtual Machines(VMs)**						
	10	**15**	**20**	**30**	**40**	**50**	**70**
PSO	380.2	340.6	280.6	280.6	250.6	210.6	195.6
APSO	380.2	340.6	280.6	280.6	250.6	210.6	195.6
NNCA_PSO	380.2	340.6	280.6	280.6	250.6	210.6	195.6

Table 5(b) Datacenter Aware

Algorithm	**Virtual Machines(VMs)**						
	10	**15**	**20**	**30**	**40**	**50**	**70**
PSO	343.5	336.6	278.4	260.6	240.8	236.8	234.0
APSO	343.5	336.6	278.4	260.6	240.8	236.8	234.0
NNCA_PSO	343.5	336.6	278.4	260.6	240.8	236.8	234.0

Table–5(a) and Table–5(b) shows that as the VMs increase by increasing the number of data centres with the same capabilities, the execution time decrease. The APSO and NNCA_PSO exhibit the same result as the PSO in this scenario.

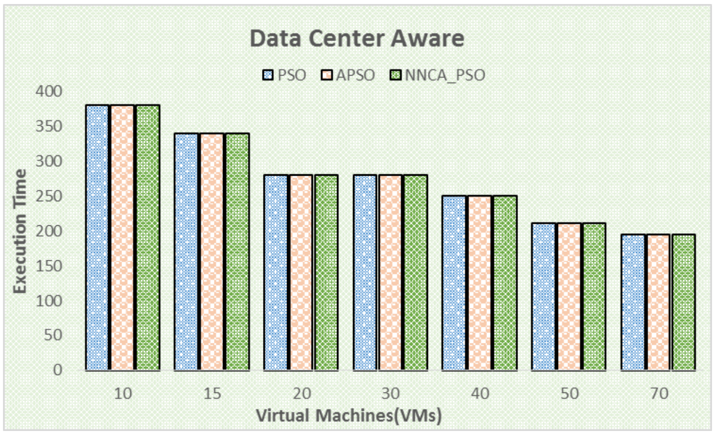

Figure 5 Execution time as per V.M bandwidth with 4 datacenter

Scenario–6: Host Storage Aware

In this there are two datacenters, each contains 2-hosts (one with quad core and other with dual core), task=40, host storage is variable and metric of result is Execution Time (ET) of the tasks by three different methods is shown in Table–6.

Table 6 Host Storage Aware

Algorithm	Virtual Machines(VMs)						
	1K	**10K**	**20K**	**30K**	**35K**	**40K**	**50K**
PSO	125.8	120.8	111.6	106.8	99.4	94.0	87.8
APSO	125.8	120.8	111.6	106.8	99.4	94.0	87.8
NNCA_PSO	125.8	120.8	111.6	106.8	99.4	94.0	87.8

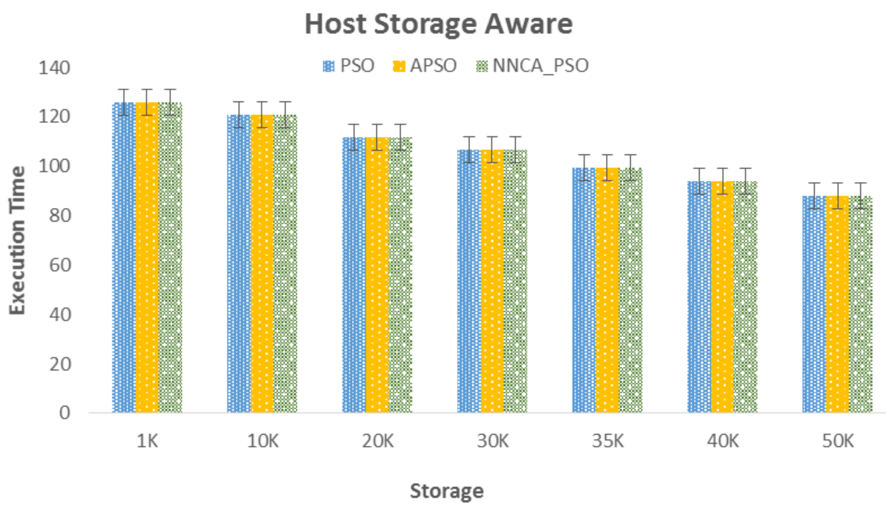

Figure 6 Execution time as per storage capacity

The Fig.6. shows that as host storage size increase, the execution time decrease. A storage requirement of VMs to store and communicate output of tasks increased. The APSO and NNCA_PSO exhibit the same result as the PSO.

Conclusion

The implementation results of APSO and NNCA_PSO are presented in different tables considering different parameters as shown above. Since convergence rate of PSO is better than other heuristic algorithm for task scheduling. The APSO and NNCA_PSO gives better result than original PSO by keeping inertia weight between 0.4and 0.9. In some scenario PSO, APSO and NNCA_PSO exhibit the same result because variable parameter doesn't enhance the searching capabilities. It is clear from the implementation results of PSO, APSO and NNCA_PSO that inertia weight (w) plays an important role to find the best cost. APSO and NNCA_PSO both keep the value of w between 0.4 and 0.9, and enhance the searching capabilities for best cost. In implementation of PSO as a scheduling algorithm, various parameters of cloud setup like VMs, hosts, bandwidth, tasks, and numbers of PEs etc. have been considered. The implementation results of APSO on CloudSim show that on average it is capable to find best cost as compared to original PSO. Our results conclude that on average the proposed method is better than the existing method. In future to accept PSO as a scheduling algorithm some other parameters like acceleration coefficient and cognitive component must be taken into account. Virtualization and scheduling approach are the other parameters which affect PSO based scheduling algorithm in cloud computing.

References

Bratanis, K., Kourtesis, D., Paraskakis, I., & Braun, S. A Research Roadmap for Bringing Continuous Quality Assurance and Optimization to Cloud Service Brokers. *Proc. eChallenges*, 2016.

Wilkes, J., & Reiss, C. (2015). Details of the ClusterData-2011-1 trace, 2015. https://code.google.com/p/

Snaith, B., Hardy, M., & Walker, A. (2011). Emergency ultrasound in the prehospital setting: the impact of environment on examination outcomes. *Emergency Medicine Journal*, *28*(12), 1063-1065.

Buyya, R., Broberg, J., & Goscinski, A.M. (Eds.). (2010). *Cloud computing: Principles and paradigms* (Vol. 87). John Wiley & Sons.

Othman, M., Madani, S.A., & Khan, S.U. (2013). A survey of mobile cloud computing application models. *IEEE communications surveys & tutorials*, *16*(1), 393-413.

Proebster, M., Kaschub, M., Werthmann, T., & Valentin, S. (2012). Context-aware resource allocation for cellular wireless networks. *EURASIP Journal on Wireless Communications and Networking*, *2012*(1), 1-19.

Rahimi, M.R., Venkatasubramanian, N., Mehrotra, S., & Vasilakos, A.V. (2012). MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. *In IEEE Fifth International Conference on Utility and Cloud Computing*, 83-90.

Rahimi, M.R., Venkatasubramanian, N., & Vasilakos, A.V. (2013). MuSIC: Mobility-aware optimal service allocation in mobile cloud computing. *In IEEE sixth international conference on cloud computing*, 75-82.

Zhang, Q., Zhu, Q., Zhani, M. F., Boutaba, R., & Hellerstein, J.L. (2013). Dynamic service placement in geographically distributed clouds. *IEEE Journal on Selected Areas in Communications*, *31*(12), 762-772.

Miettinen, A.P., & Nurminen, J.K. (2010). Energy efficiency of mobile clients in cloud computing. *HotCloud*, *10*(4-4), 19.

Di, S., Kondo, D., & Cirne, W. (2012). Host load prediction in a Google compute cloud with a Bayesian model. *In SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 1-11.

Heo, J., Terada, K., Toyama, M., Kurumatani, S., & Chen, E.Y. (2010). User demand prediction from application usage pattern in virtual smartphone. *In IEEE Second International Conference on Cloud Computing Technology and Science*, 449-455.

Saripalli, P., Kiran, G.V.R., Shankar, R.R., Narware, H., & Bindal, N. (2011). Load prediction and hot spot detection models for autonomic cloud computing. *In fourth IEEE international conference on utility and cloud computing*, 397-402.

Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M., & Yimwadsana, T. (2005). Predictability of web-server traffic congestion. *In 10th International Workshop on Web Content Caching and Distribution (WCW'05)*, 97-103.

Andreolini, M., & Casolari, S. (2006). Load prediction models in web-based systems. *In 1st International Conference on Performance Evaluation Methodologies and Tools*, 2006.