ISSN No. 0976-5697

Volume 7, No. 4, July-August 2016

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Hybrid Scheduling Algorithm to improve the Performance using Ant Colony Optimization and Cuckoo Search Algorithm with K-means++ in a Virtualized Environment

A.P. Nirmala
Sr. Asst. Prof., New Horizon College of Engg., Bangalore.
(Research Scholar, Karpagam University, Coimbatore, India)

Dr. S. Veni
HOD, Department of Computer Science,
Karpagam University, Coimbatore, India

Abstract: Virtualization plays a vital role in cloud computing. It provides better manageability, availability, optimistic provisioning, scalability and resource utilization in current cloud computing environments. However the performance isolation is a major concern in virtualization. The performance of the application running inside the virtual machine gets affected by the interference of the co-located virtual machines. This approach provides a novel task scheduling mechanism that provides the proper management of resource allocation among the virtual machines running simultaneously. An interference prediction scheme is proposed to utilize the application characteristics collected from the VMs to maintain a low system overhead. The Nelder-mead method is used to frame the relationship model from the observed response and control variables. The hybrid algorithm: Ant Colony Optimization and Cuckoo search algorithm with K-means++ is adopted for task scheduling process. This approach shows effective performance improvements in terms of throughput and execution time.

Keywords: Virtualization; Ant Colony optimization; Cuckoo search; K-means++ algorithm; Throughput; Performance Interference.

I. INTRODUCTION

Cloud computing is a novel technique in the field of Information Technology. It is considered as the extension of distributed computing, parallel computing and grid computing. The services delivered by the cloud computing are Software as a Service (SaaS), Infrastructure as a service (IaaS) and Platform as a Service (PaaS). The user can access these services by the Pay per-Use-On-Demand model that allows the usage of shared IT resources such as data storage, server, application, network through internet [1].

Virtualization has simplifies the growth of infrastructure cloud services with a single server sharing among multiple customers. In virtualization, a server is divided into multiple virtual machines (VMs) for providing the convenient management abstraction and resource boundaries between users. However, the performance isolation provided by virtualization software is not perfect and interference between guest VMs remains a challenge. If the hypervisor does not enforce proper priorities among guests, it is easy for one virtual machine's performance to suffer due to another guest.

Thus interference might be the hindrance in attracting the performance attractive customers. Also, interference occurs when the co-existing VMs are competing for hardware resources simultaneously. Cloud users may experience the performance degradation in terms of delay in tasks' completion time and this may affect the rate of completion jobs.

Hence it is essential to propose novel techniques for sharing of resources allocated to co-located VMs in the physical machine. In this regard, many scheduling algorithm are considered of minimizing the negative impacts of co-located applications or improving the overall system performance based on throughput.

This provides the motivation to propose scheduling algorithms to improve the performance based on throughput, runtime. PSO based k-means++ algorithm [14] was proposed and implemented on different types of cloud applications in our previous work. Our proposed work extends this phenomenon using Ant Colony Optimization (ACO) and Cuckoo search with k-means++ algorithm is proposed with the aim of improving the performance by scheduling the task to various VMs with minimized interference occurs from co-located applications.

The remainder of this paper discusses the related work in section 2, proposed methodology in section 3 and the results and discussion in section 4. Section 5 discusses the conclusion with future work.

II. RELATED WORK

Zhang Y et al. [2] proposed a scheduling algorithm to improve response time, throughput and utilization in large super computing environment. A scheduling algorithm proposed by Angelou et al. [3] considers workload interference in order to improve host efficiency and VM performance.

Rui Han et al. [6] proposed a dynamic and interference aware scheduler for large-scale, parallel cloud services. In this approach, the component latency and the overall performance are predicted by collecting the workload and resource data on running services at each scheduling interval. The interference-aware scheduler identifies the non uniform components and performs the near optimal component-node allocations based on the predicted performance. This approach is designed with three modules: on-line monitors, the performance predictor and the scheduler. However this system does not reduces the component latency variability and tail latency effectively.

Wei Zhang et al. [7] designed a Minimal Interference, Maximal Progress (MIMP) scheduling system that maintains both Virtual Machine CPU scheduling and Hadoop job scheduling for interference reduction and improves the overall efficiency. MIMP increases the cluster utilization by providing the priority of different VMs and the resources available on different servers. This approach allows web applications to achieve twice throughput and has response times nearly identical to running the web application alone. The execution time is reduced effectively but this approach is still meet more deadlines.

Chin-Fu Kuo et al. [8] studied the resource allocation problem for distributed real-time systems and developed a two-level resource allocation framework with off-line and on-line phase. The off-line phase applies the dynamic programming approach for sub-optimal resource configurations whereas the on-line phase uses the greedy approach. The author proposed this approach to trade the run-time overhead including time and memory space with the optimality of the resource configuration assignment.

Yu-Chon Kao and Ya-Shu Chen [9] introduced a data-locality-aware MapReduce real time scheduling framework for interactive MapReduce applications in order to improve the quality of service. A scheduler is used for scheduling two-phase Map Reduce jobs and a dispatcher is applied for assigning jobs to computing resources by enabling the consideration of blocking and data-locality. The author also discussed the dynamic power management for run-time energy saving.

Hadi Salimi and Mohsen Sharifi [10] proposed a novel batch scheduler for a consolidated multitenant virtual environment to reduce the interference of running tenant VMs. This is done by pausing VMs based on the higher impact on proliferation of the interference. An interference model considers network and processing utilizations of VMs to identify interference. The scheduler uses this model to estimate the interference of a virtualized environment in terms of the number of concurrent VMs, processor and network utilization of VMs.

From the above review of literature, the existing technique does not build a more accurate model and does not reduce system overheads effectively. To overcome these limitations, the efficient task scheduling mechanism proposed in our paper which uses the Nelder–Mead method as Interference prediction model and ACO and Cuckoo search with k-means++ scheduling algorithm. This approach is a combination of ACO and Cuckoo search algorithm with k-means++ to make the optimized decisions in order to improve the overall performance by considering the time and the throughput.

III. PROPOSED METHODOLOGY

The proposed approach is designed for the virtualized environment to reduce the execution time and increase the I/O throughput of data-intensive application. This section describes the proposed methods and technologies in detail. The proposed work has two phases: Interference prediction model and Scheduling phase.

A. Interference Prediction Model

Prediction Model extrapolates the performance of the running data-intensive application as a function of the resource consumption by virtual machine. In this approach, Nelder–Mead method [4] is utilized for modeling the relationship between the observed response and controlled variable. This is done by characterize each application by controlled variables such as the read throughput, the write throughput, incoming bandwidth, outgoing bandwidth, Number of Processors, CPU utilization in the current guest VM domain and the CPU utilization in the virtual machine monitor.

1. Nelder-Mead Method

The Nelder–Mead method is also known as downhill simplex method is a numerical method used to obtain the minimum/maximum objective function in a multidimensional space. The Nelder-Mead simplex algorithm [11], [12] is the most widely used direct search method for solving the unconstrained optimization problem. Consider the term, $\min g(a)$, where $g: R^l \rightarrow R$ is a objective function in l dimension. Given l dimension a simplex is denoted with vertices $a_1, \ldots, a_{l+1}$. The Nelder-Mead method repetitively produces a series of simplices to approximate an optimal point of $\min g(a)$. Based on the objective function values, the vertices $\{a_i\}_{i=1}^{l+1}$ of the simplex are ordered at each iteration. The objectives values are

$$g(a_1) \leq g(a_2) \leq \cdots \leq g(a_{l+1}) \quad (1)$$

where a_1 denotes the best vertex and a_{l+1} denotes worst vertex. Suppose many vertices have same objective values, then the consistent tie-breaking rules are used as suggested by Jeffrey C. Lagarias et al. [13]. The Nelder-Mead performs four operations each being associated with a scalar parameter. The operations are reflection, expansion, contraction, and shrink and the parameters are ρ (reflection), μ (expansion), τ (contraction), and δ (shrink). The values of these parameters satisfy ρ >0, μ >1, 0< μ <1, and 0 < δ <1. If $\bar{a}$ be the centroid of l best vertices then,

$$\bar{a} = \frac{1}{l}\sum_{i=1}^{l} a_i \quad (2)$$

2. Model Training and Learning

Interference description is generated for a given application by running those application in a single VM while the remaining VMs executing various workloads in the background where n VMs are involved. By doing so, a collection of information is obtained on interference effects under several background workloads. This approach can be simply automated and supports online learning of the interference prediction model for a cloud platform. When it is not possible to capture the interference relationships among several applications, this model shall be dynamically monitored and modified. This is due to the changes in operating systems, applications, virtual machines, and cloud infrastructures.

B. Ant Colony Optimization and Cuckoo Search with K-means++ Algorithm

Based on the model training phase the available virtual machine is clustered according to the CPU capacity using K-means ++ algorithm [5]. The incoming task is allocated to each cluster by Hybrid algorithm: Ant Colony Optimization (ACO) and Cuckoo search.

Ant colony optimization is an efficient algorithm for solving the computational problems that finds the optimal path through graphs. The ant deposits the pheromone during the food searching process that can be used for exchange of information, like the status of their "work". The ant searches the food source and then returns to its nest leaving behind a trail of pheromone. Each iteration is based on ant movement from one state 'a' to another 'b'. Each ant $\aleph$ performs set $A_\aleph(a)$ of optimal elaboration to its current state in each looping. The attractiveness of move α_{ab} and the trail of the move Tr_{ab} are the two moves used to measure the probability

ρ^{N}_{ab} of the moves from one state to another. The probability of the N^{th} ant from the state from a to b is

$$\rho^{N}_{ab} = \frac{(Tr^{c}_{ab})(\alpha^{d}_{ab})}{\Sigma(Tr^{c}_{ab})(\alpha^{d}_{ab})} \quad (3)$$

where Tr_{ab} represents the pheromone amount deposited from state 'a' to 'b'. 'c' is used for controlling the influence of Tr_{ab}. α_{ab} is the state transition desirability. d is for controlling the influence of α_{ab}. When the solution is completed by all the ants, Pheromone updated by the following updated trail equation is

$$\alpha^{d}_{ab} \leftarrow (1 - C\_P)\alpha^{d}_{ab} + \sum \Delta\alpha^{N}_{ab} \quad (4)$$

where C_P denotes coefficient of pheromone evaporation. $\Delta\alpha^{N}_{ab}$ represents the state amount of pheromone dumped by Nth ant.

Cuckoo Search (CS) is a powerful optimization algorithm with the feature of simplicity. It uses a single parameter pa with population size n. This approach is easy to implement and worthy for the imitation of the breeding behavior. Thus it is applied in various combinatorial optimization problems. The cuckoo search provides better performance than other Meta Heuristic algorithms. In CS, each egg in the nest represented as a solution. The main motivation of CS is to obtain the best solution by replacing the solution which are not so-good. The equation of deriving the new solution from the application of random walk and Levy flights is

$$X_{t+1} = (X_t + jG_t) \quad (5)$$

G_t is normal distribution obtained by Levy flights. 'j' represents a fixed number of iterations to determine distant of a random walker.

```
Step 1: Initialization
        Initialize the random nests ($R_n$)
        Pheromone trail, heuristic information ($H_{info}$),
Step 2: Get Input task from 1 to n
Step 3: Apply transition rules
Step 4: for each task $T_1$ to $T_k$ do
        For each virtual machine $Vm_1$ to $Vm_l$ do
        Assign $T_i = Vm$
        End for
        End for
Step 5: perform random walk by cuckoo search from equation 5
Step 6: Pheromone updation
        For each pheromone $ph_1$ to $ph_n$ do
Step 7:   Evaluate $\rho^{N}_{ab}$ using equation 3
Step 8: perform the pheromone tail updation by equation 4
Step 9: if current task $\geq$ n task
        Then n++ and go to step 2
Step 10: Return value
```

Figure 1 Pseudo code of ACO and Cuckoo Search Algorithm

A novel optimization algorithm is developed by combining the advantage of ant colony optimization and cuckoo search. The drawback of ACO has been overcome by CS algorithm. In ant colony optimization, ant moves in random directions in order to find food source around the colony. Pheromone is deposited on the path by the ant. While solving optimization problems a local search is performed which takes considerably more time. The above draw backs can be overcome by using cuckoo search. Cuckoo search overcomes the above problem by performing the local search in ant colony optimization. One main advantage of cuckoo search is that, distinct parameter is used apart from population size. In this approach the hybrid algorithm is used to schedule the task in the available VM in cluster. Figure 1 shows the procedure is performed in each cluster to effective schedule the task in available resources

IV. RESULTS AND DISCUSSION

The experiment is developed in Java language using the Net beans IDE with the hardware configurations: Intel Core2 duo CPU with 3.40 GHz speed, 4GB RAM size and 500 GB hard disk capacity. Cloudsim, simulator tool is used for simulating the virtualized environment and Mysql is used as database. The proposed work is implemented using different file types such as Images, Pdf and Text files. Hybrid ACO and cuckoo with K-means++ algorithm obtains the efficient result with the performance metrics such as throughput, execution time.

The throughput is used to measure the number of task completed per second. Through this parameter, the efficiency of the proposed work can be determined.

Table 1 and Figure 2 gives the throughput comparison for the hybrid ACO and Cuckoo search with k-means++ and the existing PSO based K-means++ algorithm. The proposed work has throughput of 0.48 MB per second which is higher than existing technique.

Table 1 Performance evaluation on Throughput

Algorithms	***Throughput (MB/ second)***
ACO and Cuckoo search with K-means++	0.48
PSO based K-means++	0.2

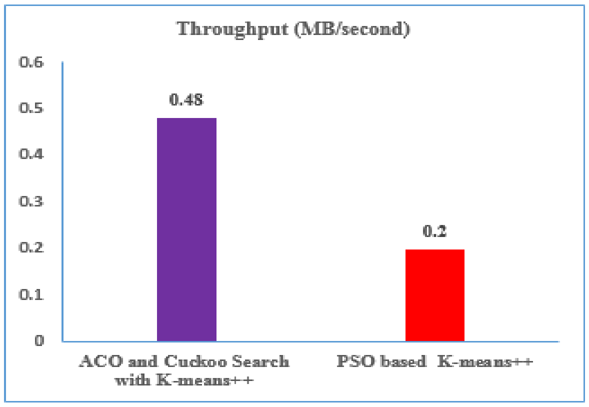

Figure 2 Performance evaluation on Throughput

Table 2 and Figure 3 gives the execution time of hybrid approach compared with existing method. The proposed ACO and cuckoo with k-means++ obtains 8 seconds whereas PSO based K-means++ obtains 20 seconds.

Table 2. Performance evaluation on Execution time

Algorithms	*Execution time (Seconds)*
ACO and Cuckoo search with K-means++	8
PSO based K-means++	20

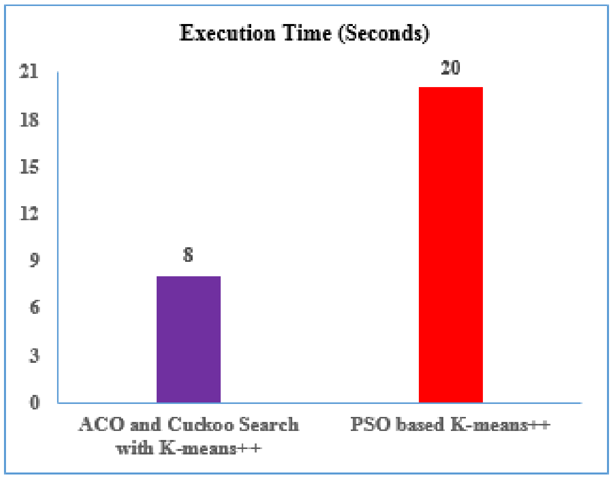

Figure 3 Performance evaluation on Execution time

V. CONCLUSION

The proposed work combines the advantages from Ant Colony Optimization and Cuckoo search algorithm. A novel scheduling algorithm based on ACO and cuckoo search with k-means++ has been proposed in order to improve the performance of cloud applications in virtualized environment. Nelder-mead method has been used in an interference prediction model. The experiment of proposed research work showed that the hybrid ACO and cuckoo search with K-means++ achieves the better performance than PSO based K-means++ algorithm. The future work is expected to be in the direction of implementing different scheduling algorithms with various file types.

VI. REFERENCES

[1] Tsai, Jinn-Tsong, Jia-Cen Fang, and Jyh-Horng Chou. "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm." Computers & Operations Research 40, 3045-3055, 2013.

[2] Zhang, Y., H. Franke, J. E. Moreira, and A. Sivasubramaniam. "A comparative analysis of space-and time-sharing techniques for parallel job scheduling in large scale parallel systems", IEEE Transactions on Parallel and Distributed Systems (2002).

[3] Angelou, Evangelos, Konstantinos Kaffes, Athanasia Asiki, Georgios Goumas, and Nectarios Koziris. "Improving virtual host efficiency through resource and interference aware scheduling." arXiv preprint arXiv:1601.07400 (2016).

[4] Ali AF, Tawhid MA., "A hybrid cuckoo search algorithm with Nelder Mead method for solving global optimization problems", SpringerPlus. 2016, 5:473. doi:10.1186/s40064-016-2064-1.

[5] Bahmani, Bahman, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. "Scalable k-means++." Proceedings of the VLDB Endowment 5, no. 7 (2012): 622-633.

[6] Rui Han, Junwei Wang, Siguang Huang and Chenrong Shao "Interference-aware Component Scheduling for Reducing Tail Latency in Cloud Interactive Services." In Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on, pp. 744-745. IEEE, 2015.

[7] Wei Zhang, Sundaresan Rajasekaran, Timothy Wood and Mingfa Zhu. "Mimp: Deadline and interference aware scheduling of hadoop virtual machines" In Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pp. 394-403. IEEE, 2014.

[8] Chin-Fu Kuo, Chi-Sheng Shih and Tei-Wei Kuo. "Resource allocation framework for distributed real-time end-to-end tasks." In Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on, vol. 1, IEEE, 2006.

[9] Yu-Chon Kao and Ya-Shu Chen., "Data-locality-aware MapReduce real-time scheduling framework", Journal of Systems and Software Volume. 112, 65-77 (2016).

[10] Hadi Salimi and Mohsen Sharifi., "Batch scheduling of consolidated virtual machines based on their workload interference model." Future Generation Computer Systems, volume 29, issue 8, 2057-2066, 2013.

[11] J. A Nelder, R. Mead, "A simplex method for function minimization", Comput. J. 7, 308–313 (1965)

[12] Luersen, M., Le Riche, R. & Guyon, F, "A constrained, globalized, and bounded Nelder–Mead method for engineering optimization", Structural and Multidisciplinary Optimization 27: 43, 2004 doi:10.1007/s00158-003-0320-9

[13] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright, "Convergence properties of the Nelder-Mead simplex algorithm in low dimensions", SIAM J. Optim. 9(1), 112–147 1998.

[14] A. P. Nirmala, R Sridaran, "Towards Performance Improvement of Cloud Applications under Virtualized Environment using PSO based K-means++ Algorithm ", International Journal of Engineering and Technology (IJET), Volume 7, Issue 5, Pages 1557-1563, ISSN : 0975-4024. Oct-Nov 2015