ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

분류와 Particle Swarm Optimization을 이용한 태스크 오프로딩 방법☆

A Task Offloading Approach using Classification and Particle Swarm Optimization

존크리스토퍼 마테오1 이 재 완1*
John Cristopher A. Mateo Jaewan Lee

요 약

클라우드 컴퓨팅에서 바이오 영감 컴퓨팅 기술과 같은 연구들을 통해, 오프로딩 기법에서 새로운 차원의 솔루션이 개발되고 있다. 모바일 장비 사용의 증가 추세에 따라, 바이오 영감 기술은 모바일 클라우드 컴퓨팅의 발전에 기여하고 있다. 모바일 클라우드 컴퓨팅에서의 에너지효율적인 기법은 총 에너지 소비를 줄이기 위해 필요하지만, 지금까지의 연구는 태스크 분산을 위한 의사결정과정에서 에너지 소비에 관해 고려하지 않고 있다. 본 논문에서는 클라우드렛에서 데이터센터로의 오프로딩 전략으로 Particle Swarm Optimization (PSO) 방법을 제안하며, 이 과정에서 각 태스크는 입자(particle)로 표현된다. 입자의 수를 줄이기 위해 PSO를 적용하기 전에 K-means 클러스터링을 사용하여 수집한 태스크를 클라우드렛 상에서 분류하며, PSO 처리과정 중에는 모든 태스크를 대상으로 하지 않고 분류된 태스크에 따라 최적의 데이터 센터를 찾는다. 시뮬레이션 결과, 제안한 PSO기법이 처리 시간 관점에서는 전통적인 방법에 비해 조금 늦지만, 에너지 관점의 데이터 센터 선택에서는 우수함을 나타내었다.

☞ 주제어 : 클라우드렛, 분류, Particle Swarm Optimization , 모바일 클라우드 컴퓨팅

ABSTRACT

Innovations from current researches on cloud computing such as applying bio-inspired computing techniques have brought new level solutions in offloading mechanisms. With the growing trend of mobile devices, mobile cloud computing can also benefit from applying bio-inspired techniques. Energy-efficient offloading mechanisms on mobile cloud systems are needed to reduce the total energy consumption but previous works did not consider energy consumption in the decision-making of task distribution. This paper proposes the Particle Swarm Optimization (PSO) as an offloading strategy of cloudlet to data centers where each task is represented as a particle during the process. The collected tasks are classified using K-means clustering on the cloudlet before applying PSO in order to minimize the number of particles and to locate the best data center for a specific task, instead of considering all tasks during the PSO process. Simulation results show that the proposed PSO excels in choosing data centers with respect to energy consumption, while it has accumulated a little more processing time compared to the other approaches.

☞ keyword : Cloudlet, Classification, Particle Swarm Optimization, Mobile Cloud Computing

1. INTRODUCTION

Since the introduction of the Cloud computing concept, various services have been provided to users such as utility computing, storage services, and applications over the Internet [1]. Cloud computing offers a broad range of services models, which are a) Software as a Service, where specific applications are offered to users, b) Platform as a Service, in which consumers can develop their own applications on platforms provided by cloud, c) Infrastructure as a Service, where IT infrastructures (processing, storage, etc.) are offered by the cloud, users can choose which of these resources they can use on the fly, thanks to virtualization. Virtualization strategy on data centers is among the commonly used since it scales well and handles heterogeneity of workloads much easier.

1 Dept. of Information and Communication Engineering, Kunsan National University, Jeollabuk-do, 573-701, Korea
* Corresponding author (jwlee@kunsan.ac.kr)
[Received 17 May 2016, Reviewed 04 July 2016, Accepted 24 October 2016]
☆ This paper was supported by research funds of Kunsan National University.

According to the Natural Resource Defense Council [2], data centers located in America contains more than 12 million computer servers and the energy consumed by these data centers are enough to power all of the houses in New York City for two years. If energy-efficient mechanisms were to be adopted, data centers could save about 40% of their electrical consumption.

With advancements in technology, Cloud computing enabled support to mobile devices so that the Mobile Cloud Computing [3] was created. Because of the limited hardware resources in mobile devices, even in technological advances, resource-intensive applications runs very slowly while draining batteries faster. These types of applications require much more hardware found in smartphones. In this scenario, mobile cloud computing covers this problem, in both energy efficiency and limitation of the hardware, by enabling methods or threads of the application which required good resources to be offloaded to cloud servers, therefore reducing the amount of time for processing and reducing energy consumed in mobile devices.

Related to mobile cloud computing, Cloudlets [4] was introduced in mobile cloud computing, in which they are established between mobile devices and the cloud. Cloudlets can be in places that are near to the users, where it can provide services with minimum network latency. Moreover, the advantages of using cloudlets can are as follows: a) the cloudlet can do pre and post processing to shorten the processing time; b) do the whole task; and c) select the data center appropriate for the task to achieve the efficient throughput. With the implementation of cloudlets, energy consumption in mobile devices is reduced, as processes are offloaded to available resources that are much more powerful than the mobile devices itself.

Particle Swarm Optimization [5] has been introduced by Kennedy and Eberhartin in 1995. Particles fly around a dimensional space in search for a better position, like a group of birds looking for food. This approach is a collective intelligence method, which means that in finding solutions, it considers its own solution (cognitive/personal) and a solution that is best (global/social) among the others. Both of these solutions are then used in order to locate a position that is considered as a proper solution. Each particle would assess different locations, in each iteration, it would obtain their personal and global best positions, and move according to these values. After the stopping criteria have been met, the particles in the population choose the optimal solution.

With the use of cloudlets in classifying tasks and selecting which data centers are appropriate for offloading, pre-processing tasks in the cloudlet lessens the overall workload in data centers, and selection of data centers for the tasks to be offloaded makes sure that tasks are offloaded to data centers that can process them with the least amount of energy consumed. This paper uses particle swarm optimization in groups of tasks collected by the cloudlet in order to find the appropriate data center. Each task is considered as a particle that searches for a solution around the dimension space containing different data centers and will find a data center to which the tasks are then offloaded.

2. RELATED WORKS

2.1 PARTICLE SWARM OPTIMIZATION ON CLOUD COMPUTING

Particle Swarm Optimization has yielded good results in the field of cloud computing. It is described as a self-adaptive search for a globally optimal solution. It is comparable to other population-based algorithms, like Genetic algorithms. The optimization depends on both the cognitive and social behaviors of each particle.

Pandey et al. [6] introduced particle swarm optimization for workflow scheduling of applications in cloud computing to minimize the total cost of the resource usage used by applications found in cloud computing environments. It can also tackle load distribution of tasks to determine available resources. Both the computation and transmission costs were included in PSO, in order to minimize the execution costs of application workflows. However, energy consumption was not included.

Yin et al. [7] applied Particle swarm in distributed systems to minimize the system cost and resource usage. A hybrid version of the optimization is introduced in searching for a near-optimal task assignment. Its strategy is to examine each particle's vector and leave the best one that has the highest value of the vector. Although an improved version was introduced, it did not take into account the total energy consumed of resources.

Baby [8] used PSO algorithm in load balancing of tasks in virtual machines in a best-fit manner. The study compared it to a honeybee behavior of load balancing. One advantage of the particle swarm over the honeybee method is that particle swarm checks all virtual machines available and assigns each task to a proper virtual machine, but this process will take too long in real situations, where a vast amount of virtual machines has to be checked individually.

Task scheduling in virtual machines is proposed in [9, 10], where Awad et al. [9] applied PSO for task assignment to virtual machines that resides with hosts. A load balancing mutation algorithm checks the best solution of the PSO that handles tasks which failed to be assigned. Also, energy consumption of the hosts was not included.

Al-maamariet al. [10] proposed a dynamic adaptive particle swarm optimization (DAPSO) where it improved the inertia weight in global searches paired with a Cuckoo Search (CS) algorithm in improving inertia weight in local searches. It includes minimizing the makespan of tasks as well as maximizing resource utilization in virtual machines, although it does not include energy consumption.

2.2 TASK DISTRIBUTION TO CLOUD RESOURCES

Task scheduling and selecting among data centers are important issues that have an impact in the response time in task distribution. Energy consumption of cloud resources is also considered as a factor that is used to create or improve the approaches regarding this scope, as energy efficient mechanisms in data centers are very important in order to reduce the amount of energy consumed by these resources.

Nirubahet al. [11] proposed an energy-efficient task scheduling algorithm combining two energy-efficient schemes, in which each task is evaluated to each available server. First, each task is categorized into types like reading file contents, updating data, etc. As the processing time was calculated, each server assesses the energy consumption of tasks. It will choose the server that will consume the least amount of energy with respect to the completion time of the task.

Gu et al. [12] introduced a task placement scheme to Geo-distributed data centers. Different operational expenditures were considered because electricity costs of each data center vary depending on the location. A Dynamic Voltage Frequency scaling (DVFS)-aware was proposed which includes the electricity charge diversities of data centers while guaranteeing the quality of service. Round trip time was not considered as one of the restraints with regards to obtaining a better quality of service.

Soyata et al. [13] designed a mobile-cloudlet-cloud architecture that minimizes the overall response time of the application by considering the communication latencies and computational power of cloud servers at diverse locations and cloudlets. They developed a face recognition application, by which face detection is processed on cloudlets, and recognition process is processed in data centers. Faces found during detection are distributed among data centers. Overall response time decreases as the number of cloud server's increase and is more effective when cloudlets are present.

Soyata et al. [14] used the same architecture for military purposes, in which devices, such as cameras that has an object recognition feature can offload tasks to cloudlets installed nearby, therefore reducing the amount of time consumed due to the limited resources in the device. Using cloudlets accelerates application's response time, although additional hardware is needed for installing the cloudlets.

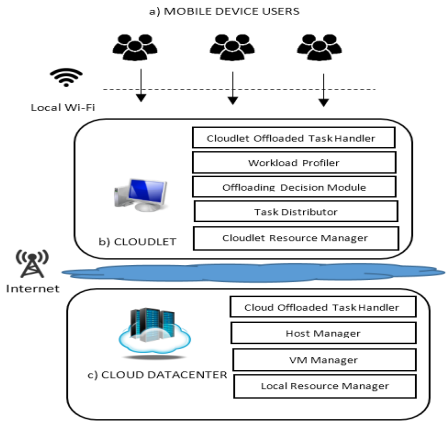
In both studies they used two mechanisms in distributing tasks to the cloudlet and cloud servers, which are the fixed and greedy approach. The fixed approach distributes all of the tasks evenly to all resources, and the Greedy approach arranges the computing resources based on their last response time and will give the task to the server which can complete the task with the least time. Only the response time was included during the process, not the energy consumed on cloud servers.

Although [13 & 14] utilized cloudlets in order to reduce the workload on mobile devices, they did not consider energy consumption in data centers. In this paper, both the transmission cost and energy consumption are used as parameters in deciding which cloud resources they should use.

3. PROPOSED APPROACH

3.1 SYSTEM ARCHITECTURE

Figure 1 shows the proposed system which consists of three components. The a) mobile device component is composed of mobile devices that are connected to a local Wi-Fi with an established cloudlet. Mobile devices can offload tasks to a cloudlet. The b) cloudlet component provides the mobile users with resources, doing pre and post processing of tasks. Other tasks which are not assigned or can't be executed in the cloudlet because the resources are not enough can be assigned to the cloud data centers. Cloudlet acts as the bridge between mobile devices and cloud resources. The c) cloud data center component, contains a large pool of resources available for tasks offloaded from the cloudlets to be executed. This type of tasks requires a lot of processing power, or it may require certain data in which the cloudlets lack. The data centers then return the results back to the cloudlet. The cloudlet can do post processing to some tasks before it returns finally the results to the mobile device. The components and their modules found in Figure 1 are as follows:



(Figure 1) The proposed system architecture

Cloudlet Layer

- Cloudlet Offloaded Task Handler – this module collects the tasks offloaded from mobile devices.
- Workload Profiler – this module classifies the collected tasks by clustering them according to their task parameters. Tasks are classified according to the available cloud services offered, compute-intensive tasks, memory-intensive tasks are examples.
- Offloading Decision Handler – this module proceeds to decide to which data center these tasks are offloaded. After the tasks are clustered and classified, each cluster of tasks will undergo the particle swarm process, where each task would be assigned as a particle. Once the PSO process has been finished these clusters of tasks are assigned to the specific data center.
- Task Distributor – this module will distribute the tasks assigned to their respective data centers after the particle swarm optimization has been applied.

Cloud Layer

- Cloud Offloaded Task Handler – this module will accept tasks assigned to it from the cloudlets. It will store the tasks until it has been assigned to virtual machines for processing.
- Host Manager – this module manages each host found in the data center. It contains the list of virtual machines and their respective metrics.
- Virtual Machine (VM) Manager – this module manages each virtual machine (VM) created.

3.2 WORKLOAD/DATA CENTER CLUSTERING AND CLASSIFICATION

The system is composed of a cloudlet with a set of tasks waiting to be offloaded to a set of data centers, each task differs from requirements (operating system required, the amount of RAM, and type of operation, etc.) and data centers with different available resources. Tasks from mobile devices are collected on the cloudlet, to which they are classified, for example, tasks with high CPU requirement can be classified as compute-oriented tasks, while tasks that need to retrieve data from databases can be classified as database-oriented tasks. Our goal is to use K-means for clustering and classifying both tasks and data centers, then matching the tasks with a specific application type to data centers with more available resources that corresponds to the application

type, for example, tasks which are compute-intensive requires more CPU, so they require data centers with a high availability of the processing resource.

The K-means clustering [15, 16] is used in grouping similar objects through the means of determining whether the parameters of said objects are closer to a cluster head or centroid, by measuring the Euclidean distance of each object to k-number of clusters. The following are steps in grouping objects using the K-means:

1. Specify the number of K clusters. For this paper, K is specified to the number of services the cloud offers.
2. During the first step, the centroids are randomly chosen among the objects. Afterward, calculation of centroid values is done by getting the average values of the point found in the cluster.
3. Calculate non-centroid object's distance from each assigned centroid. This can be done by using the Euclidean distance solver, Equation 1. The object is then assigned to the centroid closest to it.

$$d(x,y) = \sqrt{(x1-x2)^2 + (y1-y2)^2} \quad (1)$$

4. Recalculate the value of centroid for each cluster.
5. Repeat step 3 until all the centroids do not move or change.

The K-Means clustering uses the parameters found in tasks, where ti=<Ct, Mt, Bt>, Ct is the requested CPU, Mt is the memory required and Bt is the bandwidth requested.

For the parameters used in clustering data centers, let di = <Cd, Md, Bd>, where Cd is the available CPU, Md is the available memory and Bd is the available bandwidth. These will help classify these tasks and match them to data centers with enough resources.

3.3 PARTICLE SWARM ON DATA CENTER SELECTION OF TASKS

In Particle Swarm Optimization mechanism, the particles adjust its movement according to the calculated velocity, which is influenced by its best position and the position of the best particle in finding the solution. The performance of each particle is measured by a fitness value. The fitness value in this paper is to find a solution with the least energy consumed in data centers. Equations 2 and 3 provide a way each particle to move across the dimensional space.

$$V_p = V_p + c_1 * rand * (pBest - \text{Pos}) + c_2 * rand * (gBest - Pos) \quad (2)$$

$$Pos = Pos + V_p \quad (3)$$

In Equation 2, V_p represents the velocity of the particle. This value determines the speed and is added to the current position of the particle. The personal best, *pBest*, is the location of the best solution of one particle, whether it is the current solution or the ones before if the current is not optimal compared to last solutions it has traveled. After the *gBest*is obtained once all of the particles have their personal best solution, they are ranked according to their fitness value, and the particle with the highest/lowest value is chosen, and the location of that solution is considered to be the *gBest*.

Pos is the current position of the particle. The *rand* generates a random number between 0 and 1, while *c1* is the coefficient of the cognitive component and *c2* is the coefficient of the social component, and both of these values are usually close to 2. Changing these two values can affect the velocity of the particle. A higher coefficient in *c1* results to a velocity that follows its own solution, and a higher coefficient value in *c2* will result to following the best particle in the population. In this case, the values of the coefficients are equal. After the velocity of the particle has been determined, it will update its current position, in Equation 3.

The fitness solution of each data center can be assumed by two factors, one is the amount of time the task is processed, which can be calculated by the task's length divided by the resource it requires (CPU), and energy consumed during processing. The fitness value is showed in Equation 4.

$$Fv = Energy_{DC} = Time_{Task} * Power\text{DC} \quad (4)$$

$$Power_{DC} = Pow_{min} + (Pow_{max} - Pow_{min}) * util \quad (5)$$

To calculate the power generated by the data center, we

used Equation 5, where Pow_{max} is the energy consumption at 100%, Pow_{min} is the energy consumption at idle state and *util* is the current resource utilization of a data center. Energy consumed by data centers is the basis of the particle swarm optimization

The steps in using the Particle Swarm Optimization algorithm in task offloading are as follows:

1. For each cluster of tasks, initialize the particles on the dimension space randomly. each task represents a particle.
2. Proceed to evaluate the fitness value of each particle to the nearest data center, estimating the amount of energy consumed using Equation 5. This will be the particle's location of the current best solution at initialization.
3. Compare the fitness value of particle to its previous best position if another solution is nearby. If the *pBest* value is better than the previous *pBest*, set the current *pBest* location as the best personal solution of the particle.
4. Choose the particle with the best fitness value and obtain that particle solution location, and it will be the global best solution, *gBest*. In this case, the particle with the least fitness value is chosen to have the *gBest* value.
5. Compare the current *gBest* value to the previous value. If the current value is better than the previous *gBest*, set the current value as the global best value of the swarm.
6. After obtaining the location of both *pBest* and *gBest*, calculate the particle's velocity.
7. Update each particles position using the current velocity of the particle.
8. Check if a stop criterion is met. If it is met, terminate the process. If not, go back to step 2.

During the process, each cloudlet is represented as a particle. The particle position depends on 2D values of the task, which is random during initialization. In each iteration, each particle will get their personal best by obtaining the nearest data center and calculating its fitness value. If on the next iteration, a new data center is assigned to a particle, it will calculate the new location's fitness value and compare to its best personal fitness value, and if it is less, since our goal here is to obtain the least amount of energy, it will be assigned the new personal best solution for that particle. In finding the global best solution, once all of the particles obtain their personal best solution, one particle with the lowest fitness value is selected and the solution's location is considered to be the global best solution. The processing overhead in selecting for the particle with the best fitness value depends on the number of *n* particles in the process. When both the personal and global best solution have been found, it will proceed on calculating the velocity for each particle. Then it will update its position by adding the velocity to the current position of the particle.

As soon as the particles start moving from one data center to another, it will continually check for fitness values of each data center. The location of the best data center for the whole iteration is determined as soon as all particles gathered up on one data center during the nth iteration, checking if each particle's nearest data center are the same. It will assign all of the tasks to the best data center chosen, therefore add data center j's current load. If this condition is met, the iteration will stop, then proceed to another cluster of tasks, if there is still available.

4. SIMULATION AND EVALUATION RESULTS

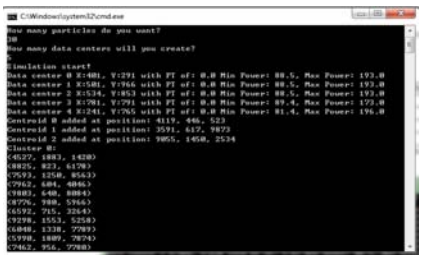
4.1 SIMULATION SETUP

(Table 1) 10 out of 20 Data Center with their properties

Unit Name	Processor Type	Avg. Watts @ 100%	Avg. Watts @ idle
ASUS RS160-E5	Intel Xeon L5430 (2.66 GHz)	173	89.4
Altos R520	Intel Xeon E5430 (2.66 GHz)	206	135
Gateway GR160 F1	Intel Xeon L5520 (2.26 GHz)	196	81.4
Acer AT350 F1	Intel Xeon L5530 (2.40 GHz)	193	88.5

The simulation on K-means and Particle Swarm Optimization was done and implemented using JavaSDK build 1.7. Task parameters are generated randomly, and these were clustered and classified according to three types, A) Computer-intensive tasks, B) Memory-intensive tasks and C)

Database-intensive tasks. The power consumption of data centers is based on [17]. The following data centers used in the simulation are found in Table 1. Figure 2 shows a sample of the simulated data centers with their respective parameters used in the process. Data centers in the simulation have the same total resources for CPU, Memory, and Bandwidth.



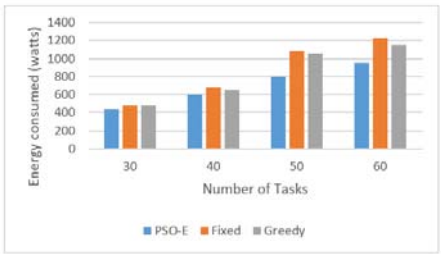
(Figure-2) Data center creation and addition of centroids for K-Means process.

4.2 EVALUATION RESULTS

To determine whether using the Particle Swarm Optimization (PSO) in distributing cluster of tasks matched to a cluster of data centers is efficient, we included two task distribution mechanisms, fixed and greedy. Fixed approach distributes tasks evenly to all data centers, and Greedy selection prefers data center with the least known processing time. Our approach (PSO-E) in which particle swarm optimization was applied is to obtain the least amount of energy consumed by data centers. Energy consumed by used data centers is the basis for the performance of each approach.

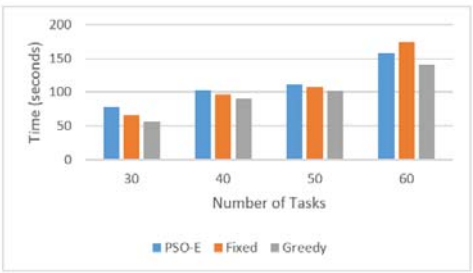
In calculating for the energy consumption of data centers, we used Equation 4 as the basis, we assumed that all of the hosts used in a data center are heterogeneous, each with energy consumption varying from different utilization levels. After placing all the tasks on each approach, energy consumed is calculated and compared to other mechanisms.

Figure 3 shows the total accumulated time used for processing the tasks in data centers. Fixed approach distributed each task to data centers, ignoring performance parameters. As for Greedy selection, it selects and assigns each task to data centers that can process tasks with minimum time, obtaining the least amount of time of the three. PSO-E obtained the most in terms of total accumulated processing time at 30-50 cloudlets. However, Fixed approach tops all others at 60 tasks, because distributing equally on all data centers does not yield optimal results.



(Figure 3) Total accumulated processing time of tasks in each approach with 20 data centers.

Figure 4 shows how much energy in watts each approach had consumed. In here, we can see PSO-E focuses on finding a solution where energy consumption is minimal, therefore obtaining the least energy consumed at all number of tasks. Fixed approach distributes evenly all of the tasks to all data centers, which leads to using all of them, thus keeping them underutilized. The greedy approach uses fewer data center compared to Fixed, but PSO-E utilized the least amount of data centers and therefore produces the least amount of energy consumed. Using PSO in task distribution compared to the Fixed and Greedy approach is much better because in finding a solution, each particle would include its own solution and at the same time, the particle which holds the best solution, therefore, would help in making decisions as the particles keep moving in finding the optimal solution.



(Figure 4) The total amount of energy consumed of each approach with 20 data centers with regards to a number of tasks.

5. CONCLUSION

Deploying lots of cloudlets is possible in the future, because of its innovations and improvement in mobile cloud computing. Particle Swarm Optimization uses a bio-inspired technique where each particle finds a solution through the means of both personal best and social best solutions, allowing them to converge on possible global solutions which can benefit if not all, most of the particles. The velocity of each particle is affected by these values. Each particle evaluates data centers which are nearest to them according to their energy consumption, obtaining personal bests solutions. Particles will keep moving until clusters had obtained their best data centers. Best data centers for each cluster were obtained when each particle of clusters converged together in one data center. Simulation results showed that application of PSO in task distribution resulted in a less energy consumption of data centers.

In future research on the application of PSO to offloading of tasks in cloudlets, a dynamic of approach can be applied, for example putting weights on the particle velocity calculation. It can choose whether to favor more the personal or global solution. Also, a trade-off between processing time and energy consumption can be considered as well. A dynamic approach can be used where it adjusts whether to prioritize a data center with least processing time or least energy consumption.

참 고 문 헌(Reference)

[1] Dillon, T., Wu, C., Chang, E., "Cloud computing: issues and challenges", Advanced Information Networking and Applications (AINA), 24th IEEE International Conference, 2010.
http://dx.doi.org/10.1109/AINA.2010.187.

[2] Delforge, P., "America's Data Centers Consuming and Wasting Growing Amounts of Energy", Natural Resource Defense Council, August 2014.
https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts-energy.

[3] Warkehar, P., Gaikawad, V. T., "Mobile Cloud Computing, Approaches and Issues", International Journal of Emerging Trends & Technology in Computer Science, vol. 2, issue, March – April 2013.
http://dx.doi.org/10.1016/j.simpat.2014.05.009.

[4] Satyanarayanan, M., Bahl, P., Caccres, R., Davies, N., "The Case for VM-Based Cloudlets in Mobile Computing", IEEE Pervasive Computing, vol. 8, issue 4, pp. 14-23, 2009.
http://dx.doi.org/10.1109/MPRV.2009.82.

[5] Satyanarayanan, M., Bahl, P., Caccres, R., Davies, N., "The Case for VM-Based Cloudlets in Mobile Computing", IEEE Pervasive Computing, vol. 8, issue 4, pp. 14-23, 2009.
http://dx.doi.org/10.1109/MPRV.2009.82.

[6] Pandey, S., Wu, L., Guru, S. M., Buyya, R., "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400-407, 2010.
http://doi.ieeecomputersociety.org/10.1109/AINA.2010.31.

[7] Yin, Y., Yu, S., Wang, P., Wang, Y., "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems", Computer Standards & Interfaces, vol. 28, issue 4, pp. 441-450, 2006.
http://dx.doi.org/10.1016/j.csi.2005.03.005.

[8] Baby, A., "Load Balancing in Cloud Computing Environment using PSO Algorithm", International Journal for Research in Applied Science and Engineering Technology, vol. 2, issue 4, April 2014.
http://www.ijraset.com/fileserve.php?FID=349.

[9] Awad, A. I., El-Hefnawy, N. A., Abdel-kader, H. M., "Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments", International Conference on Communication, Management and Information Technology, vol. 65, pp. 920-929, 2015.
http://dx.doi.org/10.1016/j.procs.2015.09.064.

[10] Al-maamari, A., Omara, F. A.," Task Scheduling using PSO Algorithm in Cloud Computing Environments", International Journal of Grid Computing, vol. 8, no. 5, pp. 245-256, 2015.
http://www.sersc.org/journals/IJGDC/vol8_no5/24.pdf.

[11] Nirubah, T. J., John, R. R., "Energy-Efficient Task Scheduling Algorithms for Cloud Data Centers", International Journal of Research in Engineering and

Technology, vol. 3, issue 3, March 2014.
http://esatjournals.net/ijret/2014v03/i03/IJRET20140303059.pdf.

[12] Gu, L., Zeng, D., Barnawi, A., Guo, S., Stojmenovic, I., "Optimal Task Placement with QoS Constraints in Geo-distributed Data Centers using DVFS", IEEE Transactions on Computers, vol. 64, no. 7, pp. 2049-2059, 2015.
http://dx.doi.org/10.1109/TC.2014.2349510.

[13] Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., Heinzelman, W., "Cloud-Vision: Real-time Face Recognition using a Mobile-Cloudlet-Cloud Acceleration Architecture", International Symposium on Computers and Communications, July 2012.
http://dx.doi.org/10.1109/ISCC.2012.6249269.

[14] Soyata, T., Muraleedharan, R., Langdon, J., Funai, C., Ames, S., Kwon, M., Heinzelman, W., "COMBAT: mobile-Cloud-based cOmpute/coMmunications infrastructure for BATtlefield applications", Modeling and Simulation for Defense Systems and Applications vol. 7, 2012.
https://www.cs.rit.edu/~jmk/papers/combat-spie.pdf.

[15] Panchal, B., Kapport, R. K., "Dynamic VM Allocation algorithm using Clustering in Cloud Computing", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, issue 9, 2013.
https://www.ijarcsse.com/docs/papers/Volume_3/9_September2013/V3I9-0119.pdf.

[16] Kordinariya, T. M., Makwana, P. R., "Review on determining number of Cluster in K-means Clustering", International Journal of Advance Research in Computer Science and Management Studies, vol. 1, issue 6, November 2013.
http://www.academia.edu/5514429/Review_on_determining_number_of_Cluster_in_K-Means_Clustering.

[17] Standard Performance Evaluation Corporation, http://www.spec.org/

◐ 저 자 소 개 ◑

존크리스토퍼마테오 (John Cristopher A. Mateo)
2015년 West Visayas State University, Philippines
BS in Information Technology
2015년～현재 Kunsan National University, South Korea, Graduate Student in Master's Course
관심분야 : Cloud computing, mobile cloud computing, intelligent algorithms

이 재 완 (Jaewan Lee)
1984년 중앙대학교 이학사-전자계산학
1987년 중앙대학교 이학석사-전자계산학
1992년 중앙대학교 공학박사-컴퓨터공학
1996년 3월~1998년 1월 한국학술진흥재단 전문위원
1992년~ 현재 군산대학교 교수
관심분야 : 분산 시스템, 운영체제, 유비쿼터스 시스템, 클라우드 컴퓨팅 등