http://dx.doi.org/10.5755/j01.eee.22.1.14113 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 22, NO. 1, 2016

A New Technique for Optimizing Resource Allocation and Data Distribution in Mobile Cloud Computing

Pham Phuoc Hung [1], Tuan-Anh Bui[2], Kwon Soonil[3], Eui-Nam Huh[1]
[1]*Department of Computer Engineering, Kyung Hee University, Yongin, South Korea,*
[2]*Department of Information Technology, The Catholic University of Lovain, Louvain-la-Neuve, Belgium,*
[3]*Department of Digital Contents, Sejong University, Seoul, South Korea*
hungpham@khu.ac.kr

***Abstract*—The emergence of mobile cloud computing (MCC) brings benefits to mobile users and cloud providers. However, due to the inherent limitations of the device such as battery life time, CPU and memory capacity, a mobile thin client device (*e.g.* smart phones, tablets, iWatch, Google Glass, *etc*) cannot meet the requirements of some demanding applications. To alleviate this limitation, the mobile device should cooperate with external resources to increase its performance. Recently, current research approaches have been unable to offer an efficient, seamless computing experience. In this paper, we present a comprehensive thin-thick client collaboration that involves conventional desktop or laptop computers, known as thick clients, by allowing the thin client to borrow resources from thick clients, particularly for optimizing data distribution and utilizing MCC resources to meet Service-Level Agreements, Quality-of-Service requirements and cloud service customers' budget. Our work uses both numerical analysis and simulation to prove that our proposed architecture can improve resource allocation efficiency and achieve better performance than other existing approaches in some cases.**

***Index Terms*—Cloud computing, data distribution, thin thick client, resource allocation.**

I. INTRODUCTION

According to Gartner Inc., there were approximately 2.4 billion units of cellular phones and tablets shipped in 2013, outstripping PC sales in the same period, which were less than 315 million shipments [1]. As smart phones become more sophisticated with better hardware capability and a wider variety of software, devices running lightweight mobile operation systems (Apple's iOS, Google's Android and Microsoft's Windows Phone) will gradually supplant traditional laptop and desktop computers [2]. This "fixed-to-mobile" shift has resulted in an enormous amount of data generated by mobile devices in recent years. In a report by Cisco [3], global mobile data traffic, on a monthly basis, grew 81 percent in 2013, reaching 820 petabytes per month at the end of 2012. This number is predicted to surpass 15 exabytes by 2018. However, smart phones and tablets still do not, and probably will not soon, have power equivalent to a conventional desktop or laptop computer. They will continue to be unable to perform tasks that require heavy processing and/or large amounts of data storage. With resource inefficiency such as limited processing capability, small memory, and short battery life, mobile devices are considered as only a complement to fixed stations and not as an alternative.

Fortunately, cloud computing, with virtually unlimited resource and service provision, has arrived and is anticipated to boost the power of mobile devices. The applications of cloud computing (CC) in a mobile environment include a new computing paradigm named mobile cloud computing (MCC). MCC allows the shift, also known as offloading, of local computing-intensive tasks and data storage from mobile devices to the Internet, *i.e.* to an array of virtualized servers running on cloud provider networks. The involvement of CC in a mobile environment accommodates constraints related to performance (*e.g.*, battery life, storage, and bandwidth), environment (*e.g.*, heterogeneity, scalability, and availability), and security (*e.g.*, reliability and privacy) [4].

While cloud computing has existed for a while, its application in the mobile world is still at an early stage. Recent research has addressed MCC's drawbacks, especially those related to feasibility and performance, which has focused on allowing mobile clients to access the cloud. In [5], for example, Frank Siegemund *et al.* argue that smart objects (*i.e.* smart phones) can leverage resources and computing capabilities from nearby nodes to gain access to the cloud network instead of using a direct connection (*i.e.* through cellular network, 3G). Similarly, the authors in [6] discuss a reliable MCC architecture that emphasizes services and resource exchange among peer nodes, which simplifies the querying process of peer nodes. In [7], a guideline to create a virtual mobile cloud computing provider is proposed, based on the peer network created between nearby thin clients, to avoid the need to connect to infrastructure-based clouds. Since the network condition plays a crucial

Manuscript received 29 April, 2015; accepted 23 October, 2015.

role in guaranteeing the Quality-of-Services (QoS) of resource provisioning (*i.e.* the greater the network bandwidth, the better the QoS that can be achieved [8], [13]), thin clients should be coupled with thick clients, which come with more generous hardware resources and high speed Internet connection (DSL, Fibre, *etc.*), in order to obtain desirable cloud access [9]. However, one of the main issues in [9] is that it does not consider the budget for CC.

With that in mind, in this paper, we discuss a comprehensive thin-thick client collaboration to perform efficient data distribution along with detailed solution to perform efficient data distribution by splitting the big data into chunks according to bandwidth of Internet connection. Besides, the paper introduces a selection procedure of suitable algorithms that optimally utilize resource allocation in order to not only meet expected Service-Level Agreement (SLA) and QoS requirements but also the CC budget in order to improve users' cloud computing experience. In these algorithms, we use a multi-user multi-task technique on each virtual machine (VM) and create a group of service images (SIs) from thin clients, then integrate them into a multi-user multi-task VM. It is noteworthy that each SI now corresponds to a user on this multi-user multi-task VM. A group of service images has a location-based relation in which the thin clients are close to each other. Our objective is to calculate the number of VMs allocated for SIs to satisfy SLA as well as cloud cost for using a cloud service. In order to evaluate the eligibility and effectiveness of our work, a number of simulations were conducted and their results show that the proposed framework can significantly improve efficiency of resource allocation in the cloud, and the strategies discussed are effective enough to improve performance compared with existing approaches.

We divide our paper into the following sections: Section II is a review of the existing literature; Section III describes a motivational scenario in which the role of thin-thick client collaboration becomes crucial; Section IV specifies the framework architecture and algorithms; Section V details problem formulation. Section VI illustrates the implementation of our ideas along with a performance evaluation. The last section concludes the paper and suggests future work.

II. RELATED WORK

There have been numerous studies that attempt to solve some parts of the above problems. In [10], the authors propose a new approach for efficient cloud-based synchronization of a number of distributed file system hierarchies. They use a master-slave architecture for propagation of data to reduce traffic. In [11], researchers demonstrate that some resource scheduling techniques can be effective in mitigating the impacts that negatively influence application response time and system utilization. Andreolini *et al*. [16] and Fan *et al*. [12] study the impact of data transfer delay on the performance but they do not evaluate bandwidth efficiency. Gueyoung Jung *et al*. [14] present a method to parallelize a process with big data in order to increase performance in federated clouds; however, they do not consider how many resources should be used. For resource allocation, Ye Hu [15] shows that shared allocation is superior to dedicated allocation but the author does not conduct experiments with an arbitrary number of SLAs, nor does he determine how fast a server is to guarantee Quality of Service. In [17], A. Lenk *et al.* provide services to a large number of SLAs but the performance difference between shared allocation and reserved allocation is difficult to determine.

Similar to our approach, other research efforts have been made to integrate mobile devices and cloud computing. In [18], X. Luo suggests using the cloud to improve a mobile device's capability. Marinelly [19] innovates Hyrax, which allows mobile devices to use cloud computing platforms. The researcher introduces the idea of using mobile devices as resource providers, but the experiment is not integrated. Also, Zhong and Longzhao [20] discuss the integration of CC into mobile internet and are able to exemplify their arguments with typical successful business models in the market yet without particular performance benchmarking. In [21], the authors propose an autonomic resource manager to control the virtualized environment by recoupling the resource provision in order to optimize a function which integrates both SLA and the operating cost. However, the QoS is not considered in this system. For ease of understanding, we present an overview of common resource allocation approaches along with ours in Table I.

TABLE I. COMPARISON BETWEEN APPROACHES.

Algorithm	Target System	Satisfy SLA	Satisfy QoS	Satisfy Budget
Gonzalo H. *et al*. [7]	Cloud computing	No	Yes	No
Ye Hu *et al*.[15]	Heterogeneous	Yes	No	No
E. Marinelli *et al*. [19]	Mobile computing	No	No	No
Hien N.V *et al*. [21]	Cloud computing	Yes	No	Yes
Weiwei Lin *et al*. [22]	Cloud computing	Yes	Yes	No
Mathias B. *et al*. [23]	Cloud computing	Yes	Yes	No
Our approach	Mobile cloud computing	Yes	Yes	Yes

As shown in Table I, a desired approach should meet all three factors including SLA, QoS of the system and the budget of users in order to increase the reliability reputation of the service provider as well as satisfy Cloud Service Customers (CSCs). Therefore, in this paper, we introduce an extensive thin-thick client collaboration which takes into account these three factors.

III. MOTIVATING SCENARIO

As cloud computing becomes more popular, we can predict situations where mobile users can take advantage of cloud resources to solve their problems, especially when they are in public places. We use the following scenario (Fig. 1) to explain how the thin-thick client collaboration can benefit a user. A student wants to use video call and play games at the same time with her smartphone while she is walking on her university campus. Unfortunately, her 3G account, which comes with a limited monthly data package, has run out of capacity for high-speed Internet and the phone

begins to load Internet data very slowly. Alternatively, the student switches on the phone's Wi-Fi and attempts to connect to the university's wireless network (via nearest access points). Suppose that the student has a legitimate account. Once connected, she can continue using her cloud service as she moves around the university campus.

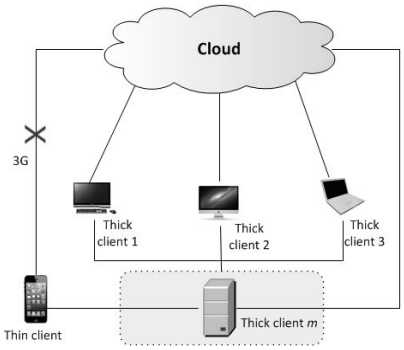

Fig. 1. Scenario for requesting a cloud service.

Let us consider this scenario more closely. After getting connected to the wireless network, the student begins to send requests for the cloud services. These requests will be handled by a group of computers that consists of a broker computer and several other computers, or thick clients, each of which can access different cloud providers. The broker first receives user requests before establishing connections with the associated cloud providers to deliver users' requests up to the cloud. Results of the fulfilled requests (i.e. video or music stream) are returned to the thick clients via the chosen paths. In this scenario, which is potentially backed by high-speed broadband connections (DSL, cable, WiMAX), service paths between thick clients and cloud networks can have considerably higher bandwidth than the one between the thin client and cloud networks which use a mobile network like 3G. Consequently the former can be expected to offer much higher service quality than the latter, which means returned data can be delivered to thick clients in a relatively short time. The broker now collects the returned data from involved thick clients, combines the data and delivers the aggregate data to the request sender. Thereby, the thin client takes advantage of multiple thick clients' relay to enhance the data distribution from cloud networks, which enhance its computing capability. When there are multiple thin clients requesting cloud services, the request-response procedure should work the same way as described above, except that the broker has to remember different request initiators so that later it can dispatch the returned data to the correct recipient.

The above scenario shows the potential benefit of utilizing joint work between thin clients and thick clients in a typical mobile cloud computing environment. Such collaboration increases the opportunity for using resources efficiently and optimizing data distribution between the mobile thin client and cloud network. With that in mind and with the CC platform staying ready and emerging on the market, our paper describes how to create a network design based on thin-thick client collaboration. We attempt to address the following problems:

– Performing the resource allocation, *i.e.*, grouping some location-based Service Images to co-use the VMs in the cloud and calculating the number of VMs to satisfy SLAs and CSCs' budget.

– Determining the strategy in order to distribute data to thin clients to meet QoS requirements.

IV. System Architecture

The following section describes our system architecture to address the above issues.

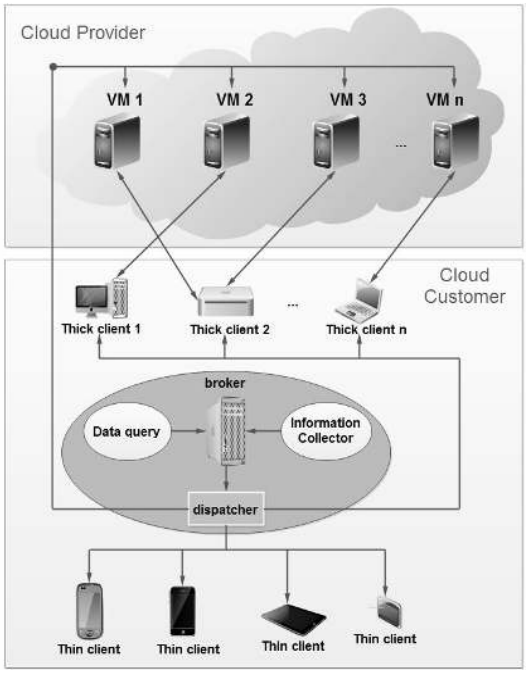

Fig. 2. Layering architecture of the proposal.

Unlike many other designs in the research literature that follows the 1/*m*/1 model, that is, there is a cloud server at one end of the transmission, a client at the other end and multiple paths between the two ends. Our proposed method implies a 1/*m*/*m*/1 model, where cloud service providers (1) send chunks of data through multiple paths (*m*) to multiple thick clients (*m*) that transfer this data to a broker in the scenario. The broker combines the received data then delivers it to the intended end user (1). As illustrated in Fig. 2, our architecture has two layers. It includes (1) a Cloud Provider layer, which contains Virtual Machines (VMs), and (2) a Cloud Service Customer layer, where thin clients and thick clients reside. In the second layer, there is a thick client functioning as a centralized management node, also known as a broker, which receives all computation requests from users and manages the processor's profiles (processing capacity, network bandwidth) as well as the results of the data query returned from VMs. In particular, it sends data to clouds in a single connection but when VMs send data to a cloud service customer layer, the data will be divided into different parts with different sizes before being delivered to thick clients in multiple connections. Next, the broker combines and sends the data to corresponding users.

V. Problem Formulation

A. Cloud Provider Layer

To make data processing more efficient, we also perform data training to classify and assign SIs. The one that has a

higher priority can be transferred first and vice versa. In addition to that, data are divided into different chunks of different sizes, according to VM capacities, then assigned to VMs. The larger data chunks are assigned to VMs with higher capacity. The best VMs are selected based on their capabilities using the Greedy algorithm. The issue here is how to determine the minimum number of VMs in order to meet expected SLAs and budgets of CSCs for running a service *s*.

By utilizing the cumulative distribution function (CDF) of the response time, denoted by $F(x)$ [16], we can measure achievable QoS, represented by the Probability in which response time *RT* remains below a threshold *x*, when the number of serving VMs is changed. That is to say, the minimum number *m* of VMs can be obtained by gradually increasing *m* until $F(x)$ reaches the target probability and CC's budget. As a result, this *m* is supposed to be satisfied for SLAs and the CSCs' finance. The CDF of response time can be described as follows

$$F(x) = \text{Probability}(RT < x) = \begin{cases} 1 - e^{-\mu x} - k\mu e^{-\mu x} x, & \text{for } \sigma = m-1, \\ 1 - e^{-\mu x} - k\mu e^{-\mu x(m-\sigma)}\left[\dfrac{1 - e^{\mu x(1-m+\sigma)}}{1-m+\sigma}\right], & \text{for } \sigma \neq m-1, \end{cases} \quad (1)$$

where $\sigma = \lambda / \mu$, $k = P(0)\dfrac{\sigma^m}{m!} \times \dfrac{m}{(m-\sigma)}$,

$P(0) = \left(\sum_{n=0}^{m-1} \dfrac{\sigma^n}{n!} + \dfrac{m\sigma^m}{m!(m-\sigma)} \right)^{-1}$, λ is arrival rate of SLA job processed at VMs with queue model M/M1 and μ is a service rate.

With regard to resource allocation methods, available VMs can be categorized into two types: Shared Allocation (SA) and Reserved Allocation (RA). In SA, the number of VMs in SA is denoted m_{Shared} and the arrival jobs (SLAs) are combined into a single stream while in RA whose the number of VMs is denoted m_{Reserved}, each arrival job has its own dedicated VMs. These allocations can be illustrated in Fig. 3. In the first type, Shared Allocation, all SLAs have the same CDF for the response time and arrival rate $\lambda = \sum_{i=1}^{k} \lambda_i$. Therefore, the minimum number of VMs m_{Shared} to meet *k* SLAs is given by

$$m_{\text{Shared}} = \max(m_1, m_2, ..., m_i, ...m_k), \quad (2)$$

where m_i is the number of VMs required for SLA_i of the i^{th} thin client. Let m_{Reserved} be the smallest number of VMs required to meet *k* SLAs in Reserved Allocation. So m_{Reserved} is calculated as follows

$$m_{\text{Reserved}} = \sum_{i=1}^{k} m_i, \quad (3)$$

In addition, the algorithm also considers the resource cost paid by cloud service customers for using cloud resources that are used to execute requested services. The cost is charged according to the utilized resources.

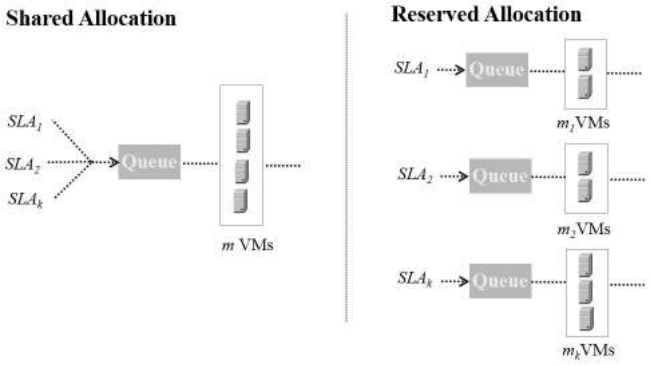

Fig. 3. Resource allocation strategy.

Let $C^s_{VM_j}$ be the cost of running service *s* on the VM which belongs to type *j* for a time unit. The type *j* here is RA or SA. In so doing, we can determine the cost $C(s)$ for using the VMs as follows

$$C(s) = \sum_{j \in \{RA, SA\}} m_j \times C^s_{VM_j}, \quad (4)$$

where m_j is the number of VMs which belongs to type *j*. Using the cost in (4) and CDF $F(x)$, we can calculate the minimum number of VMs which satisfy both SLA and the CC's budget by applying Algorithm 1.

Algorithm 1 Determine the minimum number of VMs
Input: λ *// arrival rate;*
μ *// service rate*
SLA(*x, z*) *// x : response time (threshold);*
// z : target probability
ϕ *// budget*
Output: *m* *// minimum number of VMs required*
float $\sigma = \lambda / \mu$
function determineMinVM (σ, μ, *x, z,* ϕ) {
if (σ == (*int*) σ) *m* = (*int*) σ ;
else *m* = (*int*)*Math.floor*(σ) + *1* ;
end if
while (*F*(*x*) <=*z* && *C*(*s*) $\leq \phi$) *m*++ ;
end while
return *m;*
}

Furthermore, when 2 requesters have the same SLAs, Shared Allocation obviously has the same or better performance than Reserved Allocation ($m_{\text{Shared}} \leq m_{\text{Reserved}}$). However, if SLA_1 and SLA_2 are different for SA and RA, it is not easy to determine whether SA or RA is better. An example is shown in Table II in term of SLA. In the first case, m_{Reserved} is better than m_{Shared} but the reverse is true in the second case.

TABLE II. EXAMPLE OF TWO CASES.

Case	λ_1	x_1, y_1	λ_2	x_1, y_1	m_{Reserved}	m_{Shared}
1	3.9	3, 0.7	3	6, 0.85	10	11
2	3.9	3, 0.85	2.9	5, 0.6	12	10

Because of this performance-related difference in various circumstances, it is important to select a suitable strategy, either Shared or Reserved Allocation, to satisfy SLA_1, SLA_2.

Moreover, the provisioned VM resource is also supposed to guarantee QoS requirements. Let $E(SLA)$ be the average number of VMs required to meet k the given SLAs over the range of considered arrival rates

$$E(SLA) = \frac{1}{k}\sum_{0}^{k} f(k, x, y). \quad (5)$$

Then we set the SLA difference D between SLA_1 and SLA_2. D is given by

$$D = \left|E(SLA_1) - E(SLA_2)\right|. \quad (6)$$

Our Algorithm 2 below specifies a preferred allocation strategy to satisfy SLAs and QoS requirements as well as budget. It uses the SLA difference table (Table III) to present the relationship of D and angle α. In the table, each range of D is predetermined by changing the arrival rate λ_1, λ_2 in (0, 30) and the angle is the corresponding average angle of the SLA difference in each range. The angle α is defined by sin $\alpha = \lambda_2 /sqrt(\lambda_1.\lambda_1 + \lambda_2.\lambda_2)$.

TABLE III. SLA DIFFERENCE.

D	α
[0, 20)	0
[20, 40)	20
[40, 66)	50
[66, 88)	70

In order to guarantee QoS, we have to know how fast a VM should be. Hence, we should apply a rule [8] as follows

$$E[N] = \frac{\rho}{(1-\rho)}, \quad (7)$$

where $\rho = \frac{\lambda}{\mu}$, variable $E[N]$ denotes the expectation of the number of jobs in the system. Therefore, the expectation of processing time is

$$E[T] = \frac{E[N]}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda}. \quad (8)$$

Finally, to satisfy QoS, it is required that

$$\mu \geq \frac{1}{E[T]} + \lambda. \quad (9)$$

In agreement with this formula, we can determine the service rate of the VM. For example, suppose we want $E[T] \leq 15$s, $\lambda = 1$ job/sec. The rate of VM we need is greater than 16/10.

Algorithm 2 Determine the allocation strategy
Input: λ_1 ,λ_2 *//arrival rate,*
μ *// service rate*
SLA_1, SLA_2
E *// expectation of processing time*
ϕ *// budget*
Output: *SA, RA // Allocation Strategy*

```
Function determineAllocStrategy(λ1, λ2, SLA1, SLA2,E,μ, ϕ ) {
  Calculate SLA difference D
  Get the corresponding angle α from the SLA difference table
  if (μ >=(1/E[T] + λ1)&&μ >= (1/E[T] + λ2) && (C(s) ≤ ϕ ))
        if (Math.asin (λ2 /sqrt(λ1.λ1 + λ2.λ2)) <= α)
     return RA
        else
              return SA
  end if
}
```

B. Cloud Service Customer Layer

1) Distributing data from the cloud server to thick clients

After resource allocation, we consider the distribution of source data from the cloud server to thick clients. For clarity, we give important definitions and assumptions for our system. The data distribution process happens as follows: First, each block of data is split into chunks $\{ch_1, ch_2,..., ch_n\}$ with different sizes depending on bandwidth. Let $w(ch_i)$ be the size of a chunk ch_i ; b_i be the bandwidth from a VM to a thick client. Therefore, time spent transferring a chunk ch_i from VMs to a thick client is $w(ch_i)/b_i$. For parallelization, the time to transfer chunks to thick clients should be equal, as illustrated in (10):

$$\frac{w(ch_1)}{b_1} = \frac{w(ch_2)}{b_2} = \cdots = \frac{w(ch_i)}{b_i} = t. \quad (10)$$

Set

$$S = w(data) = \sum_{i=0}^{n} w(ch_i) = t\sum_{i=0}^{n} b_i. \quad (11)$$

Thus

$$w(ch_i) = t \times b_i = \frac{S}{\sum_{i=0}^{n} b_i} \times b_i. \quad (12)$$

According to this value, we can determine the size of each chunk to adapt to the bandwidth of each connection.

2) Combining the data then transfer it to a thin client

After data from cloud service has been received, instead of using peer-to-pear synchronization between all thick clients, which might make communications more complex, the broker thick client receives data from others thick clients to decrease the complexity due to firewall between thick clients, before transferring it to corresponding thin client.

VI. IMPLEMENTATION AND DISCUSSION

In this section, the results of numerical experiments are presented to evaluate the efficiency of SA and RA and compare our approach's performance with other approaches in terms of processing time. The comparison method is the one which has one processor receiving data from the cloud provider.

A. Experimental Settings

In this experiment, characteristics of our target system are presented. We use a PC which has one Intel Core TM i7 965

and 8 GB of RAM. The OS is Windows 7 Professional. The algorithm is simulated on the CloudSim [24], which is a framework for modelling and simulation of cloud computing infrastructures and services, in Java with jdk-7u7-i586 and Netbeans-7.2. After setting the CloudSim library and building a data center, we virtualized 25 VMs in the platform, 10 thin clients and 7 thick clients. The available bandwidths between processors are from 10 Mbps to 512 Mbps so as to make different connection capacities between processors. In the bandwidth range, 512 Mbps is used for processors which have the strongest connection capacity, and 10 Mbps is used for processors which have the weakest one. The topology of all processors is fully connected.

All the parameters in the simulations have different arrival rates λ, response times, target probabilities y and some big files for the above algorithms to estimate the required minimum number of VMs for two types of resource allocation, and data distribution time. With regards to SLA settings, we set values for parameters of response time and target probability as 1 to 10, and 0.1 to 0.99, respectively. Meanwhile, speed values of requests and response services include 0.2–3.9 range for arrival rate and 1–4 range for service rate.

B. *Experimental Results*

The following figures show the simulation results of our experiment. The results prove SA and RA have almost the same impacts when they meet the same SLA with a different arrival rate λ or response time RT or target probability y. As illustrated in Fig. 4 and Fig. 5, when the arrival rate or target probability increases, the minimum number of VMs also increases.

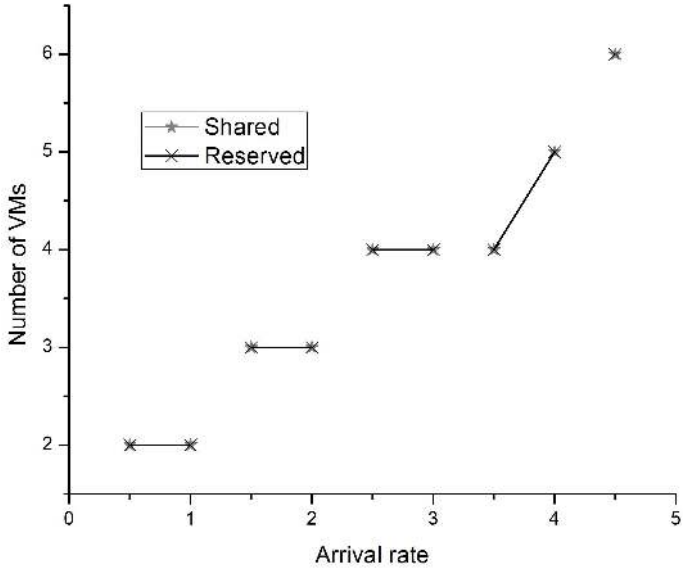

Fig. 4. SA and RA with different arrival rate.

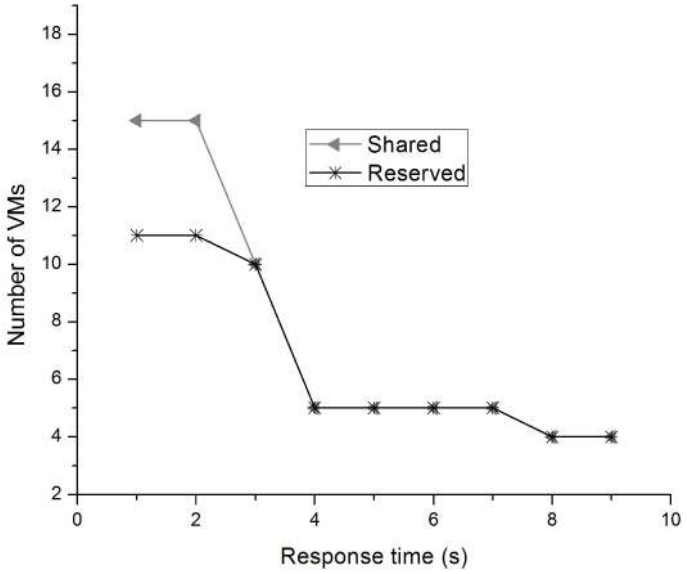

Fig. 6. SA and RA with different response time.

On the other hand, Fig. 6 show that the minimum number of VMs required to meet an SLA decreases when the response time increases. It is noteworthy that the reserved allocation is more expensive than shared allocation. Therefore, when the response time required is very small, although the number of VMs m_{Reserved} is high, it is still less than the number of VMs m_{Shared} because of the budget limitation.

Regarding the probability that satisfies SLA (as illustrated in Fig. 7 and Fig. 8), it has been observed that when the

arrival rate of the requests increases, the probability decreases. Nonetheless, the probability is proportional to the response time required.

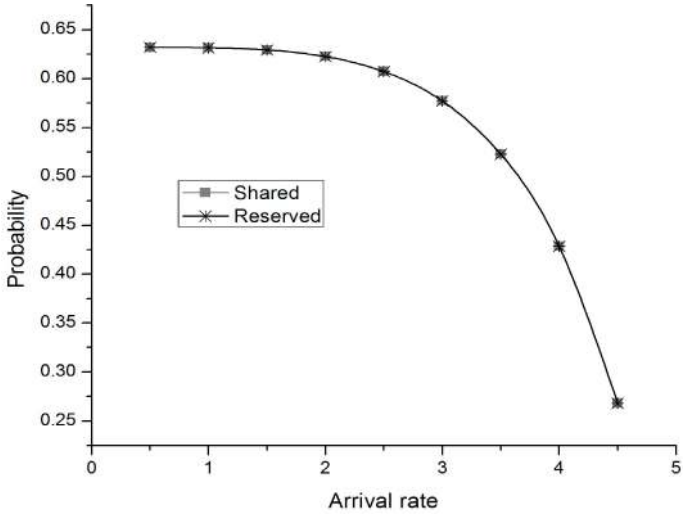

Fig. 7. Probability with different arrival rate.

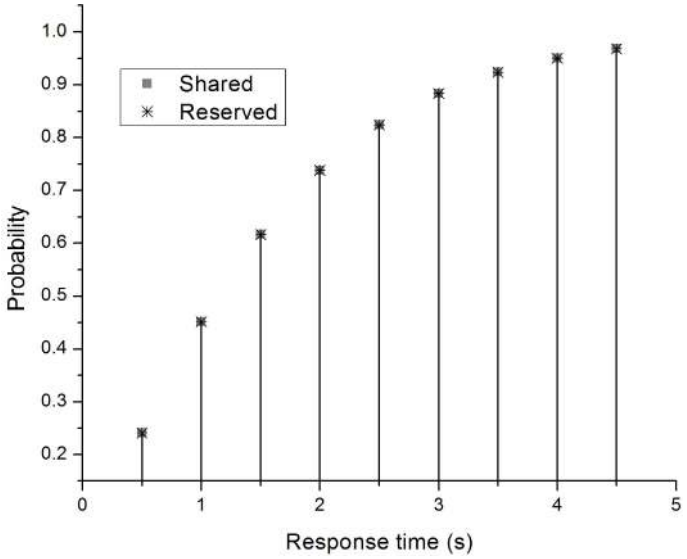

Fig. 8. Probability with different response time.

We next calculated the effect of the case where SA and RA have to meet multiple SLAs rather than a single one. It is obvious to see that with a proper budget, the strategy with SA is more resource efficient than with RA, as illustrated in Fig. 9, where SA uses fewer VMs than RA as the number of SLAs increases.

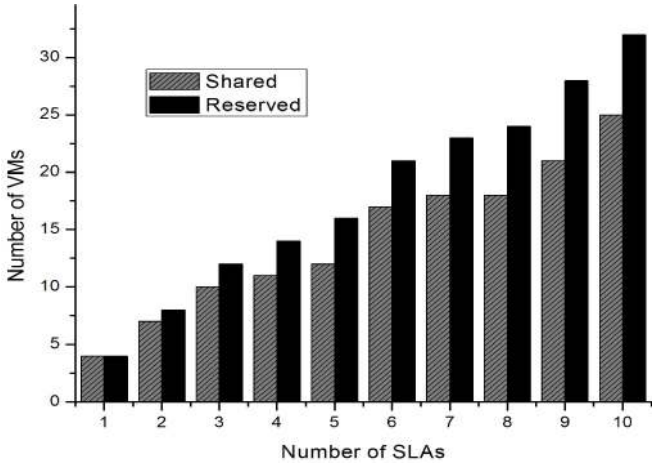

Fig. 9. SA and RA with different numbers of SLA.

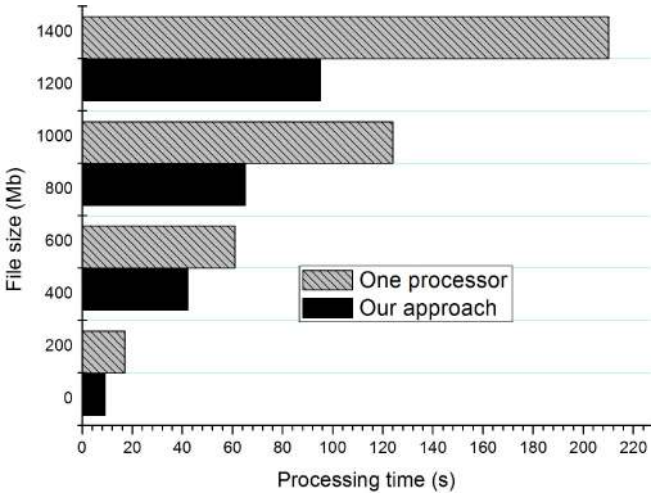

Fig. 10. Comparison of our approach with other.

This is because while the former strategy may cause the expectation of processing time to get worse if merging arrival jobs increases variability, it in fact uses shared processing resources and so requires fewer VMs than the latter. RA, to the contrary, sacrifices more resources and

thus can provide a guaranteed rate.

In addition, we further compare the processing time to transfer big data from a source to a destination of our system and one other approach, which has one processor receiving data. Based on the experimental results in Fig. 10, it is obvious that our approach produces better performance.

VII. CONCLUSIONS

This paper proposes a novel network architecture and algorithms to optimize data distribution and computing resources on the mobile cloud platform. Specifically, our approach combines resource-rich devices, or thick clients, and resource-limited ones, or thin clients, to form a collaborative network on the client slide. This collaboration, mainly leveraging the power of thick clients, can calculate and select sufficient cloud resources (measured by the number of VMs) and optimally deliver data between cloud networks and mobile clients. In so doing, the expected quality of service requirements can be satisfied. Additionally, from the multiple network resource allocation strategies being discussed, we develop an algorithm that selects the most suitable strategy so that required SLAs and CSCs' budget can be met. By carefully measuring and comparing our method with other existing ones, we have shown the novel algorithmic technique in our proposal can increase the efficiency of resource allocation and utilization with a suitable strategy. These particular advantages are visually illustrated by the minimised number of VMs used, and shorter execution time for data distribution.

Future work includes advancing the research proposal into a real-world implementation, for example, an in-campus or inter-campus cloud streaming and processing service for students and professors at universities and educational institutions. Successful experiment can be expanded into larger-scale deployment into considered public places where heavy processing needs are high. With the planned implementation, we can thoroughly observe real-world operation, performance and work out any resulting problems or shortcomings. Moreover, we will focus on improving the quality of service in order to improve the cloud service experience. Our approach may be used to deploy successful cloud-based business models to mobile users. Furthermore, we are implementing the proposed architecture using Google Glass based augmented reality and context aware services.

REFERENCES

[1] Gartner, Inc., 2013. [Online]. Available: http://www.gartner.com /newsroom/id/2408515

[2] Gartner, Inc, "Gartner says worldwide enterprise IT spending to reach $2.7 trillion in 2012". [Online]. Available: http://www.gartner.com/ newsroom/id/1824919

[3] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019. [Online]. Available: http://www.cisco.com/ c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[4] H. T. Dinh, C. Lee, D. Niyato, P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches", *Wirel. Commun. Mob. Comput*., 2011. [Online]. Available: http://dx.doi. org/10.1002/wcm.1203

[5] F. Siegemund, C. Floerkemeier, H. Vogt, "The value of handhelds in smart environments", in *Personal and Ubiquitous Computing*, vol. 9, no. 2, pp. 69–80, 2005. [Online]. Available: http://link.springer.com/ chapter/10.1007%2F978-3-540-24714-2_22

[6] P. Zhang, Peng, "A QoS-aware system for mobile cloud computing", in *IEEE Int. Conf. Cloud Computing and Intelligence Systems (CCIS)*, 2011. [Online]. Available: http://dx.doi.org/10.1109/CCIS.2011. 6045122

[7] G. Huerta-Canepa, D. Lee, "A virtual cloud computing provider for mobile devices", in *(MCS 2010)*, USA, 2010. [Online]. Available: http://dx.doi.org/10.1145/1810931.1810937

[8] N. Tien-Dung, V. N. Mui, Huh Eui-Nam, "Service image placement for thin client in mobile cloud computing", in *Int. Conf. Cloud Computing (CLOUD)*, USA, 2012. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2012.39

[9] Pham Phuoc Hung, Eui-Nam Huh, "Collaboration of thin-thick clients for optimizing data distribution and resource allocation in cloud computing", in *IT Convergence and Security*, Korea, 2013, pp. 685–693. [Online]. Available: http://link.springer.com/chapter/10. 1007%2F978-94-007-5860-5_81

[10] S. Uppoor, M. D. Flouris, A. Bilas, "Cloud-based synchronization of distributed file system hierarchies", in *IEEE Int. Conf. Cluster Computing Workshops and Posters*, 2010, pp. 1–4. [Online]. Available: http://dx.doi.org/10.1109/CLUSTERWKSP.2010.5613087

[11] J. Delgado, L. Fong, "Efficiency assessment of parallel workloads on virtualized resources", in *Fourth IEEE Int. Conf.*, 2011. [Online]. Available: http://dx.doi.org/10.1109/UCC.2011.22

[12] Pei Fan, Ji Wang, "Toward optimal deployment of communication-intensive cloud applications", in *IEEE Int. Conf. Int. Conf. Cloud Computing (CLOUD)*, 2011, pp. 460–467. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2011.54

[13] M. Kwok, "Performance analysis of distributed virtual environments", Ph.D. dissertation, University of Waterloo, Ontario, Canada, 2006. [Online]. Available: https://uwspace.uwaterloo.ca/handle/10012/2928

[14] N. Gueyoung Jung, "Synchronous parallel processing of big-data analytics services to optimize performance in federated clouds", in *IEEE 5th Int. Conf. Cloud Computing*, USA, 2012, pp. 811–818. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2012.108

[15] Ye Hu, Johnny Wong, "Resource provisioning for cloud computing", in *Conf. Center for Advanced Studies on Collaborative Research*, 2009, pp. 101–111. [Online]. Available: http://dx.doi.org/10.1145/1723028.1723041

[16] M. Andreolini, S. Casolari, M. Colajanni, "Autonomic request management algorithms for geographically distributed internet-based systems", in *Second IEEE Int. Conf. Self-Adaptive and Self-Organizing Systems, (SASO 2008)*, 2008, pp. 171–180. [Online]. Available: http://dx.doi.org/10.1109/SASO.2008.32

[17] A. Lenk, M. Klems, "What's Inside the Cloud? An Architectural Map of Cloud Landscape", *ACM/IEEE Symposium on Cloud Computing Challenges*, pp. 23-31, Vancouver, 2009. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2009.5071529

[18] X. Luo, "From augmented reality to augmented computing: a look at cloud-mobile convergence", in *Int. Symposium on Ubiquitous Virtual Reality*, 2009, pp. 29–32. [Online]. Available: http://dx.doi.org/10.1109/ISUVR.2009.13

[19] E. Marinelli, "Hyrax: cloud computing on mobile devices using mapreduce", Master thesis Draft, Computer Science Dept., CMU, 2009.[Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord &metadataPrefix=html&identifier=ADA512601

[20] L. Zhong, "Cloud computing applied in the mobile internet", in *7th Int. Conf. computer science and education (ICCSE)*, 2012. [Online]. Available: http://dx.doi.org/10.1109/ICCSE.2012.6295061

[21] Hien Nguyen Van, "SLA aware virtual resource management for cloud infrastructures", in *IEEE Int. Conf. Computer and Information Technology*, vol. 1, pp. 357–362, 2009. [Online]. Available: http://dx.doi.org/10.1109/CIT.2009.109

[22] Lin Weiwei, Peng Baoyun, "Novel resource allocation model and algorithms for cloud computing", in *Int. Conf. Emerging Intelligent Data and Web Technologies*, 2013. [Online]. Available: http://dx.doi.org/10.1109/EIDWT.2013.18

[23] B. Mathias, L. Y. Chen, "Opportunistic service provisioning in the cloud", in *IEEE Fifth Int. Conf. Cloud Computing*, 2012. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2012.85

[24] Cloudsim, "A framework for modeling and simulation of cloud computing infrastructures and services". [Online]. Available: https://code.google.com/p/cloudsim/downloads/list