QoS-Aware Dynamic Caching for Destroyed Virtual Machines in Sensor-Cloud Architecture

Arijit Roy
Student Member, IEEE
Indian Institute of Technology Kharagpur, Kharagpur 721302, India
arijitroy@iitkgp.ac.in

Sudip Misra
Senior Member, IEEE
Indian Institute of Technology Kharagpur, Kharagpur 721302, India
smisra@sit.iitkgp.ernet.in

Sayan Ghosh
Indian Institute of Technology Kharagpur, Kharagpur 721302, India
sgdgp@iitkgp.ac.in

ABSTRACT

In this work, we propose a scheme, named Quality-of-Service (QoS) Aware Dynamic Caching for Destroyed Virtual Machines in Sensor-Cloud Architecture, which enables efficient caching in sensor-cloud, in the presence of heterogeneous sensor nodes. This work is one of the first attempt of its type, in which a special cache is introduced for the efficient use of sensor data in the sensor-cloud architecture, in order to maintain QoS. Considering the reutilization of sensor data, the proposed scheme is capable of keeping data of a Virtual Machine (VM) for a certain duration of time, even if it is destroyed. Therefore, the data from SDC can be used in future, if any further requests arrive, which consists of same configurations of physical sensors inside a virtual sensor. We compared the proposed caching mechanism, *Dynamic Caching for Destroyed* VMs, with the existing mechanism proposed by Chatterjee and Misra [1]. We observe that the cache hit percentage increases at least double the number of times exhibited by the existing scheme of caching. On the other hand the energy consumption and message overhead decrease by 50% and 17% respectively.

CCS CONCEPTS

• **Networks** → **Sensor networks**; • **Software and its engineering** → **Cloud computing**; *Virtual memory*;

KEYWORDS

Sensor-cloud, Wireless Sensor Networks, Dynamic caching, Destroyed virtual machine, QoS.

1 INTRODUCTION

The emerging technology of sensor-cloud [1, 2, 8] becomes an alternative to the traditional wireless sensor networks (WSNs). Sensor-cloud virtualizes the physical sensors to serve multiple remote users at the same time, thereby enabling the provisioning of *Sensors-as-a-Service* (Se-aaS). The existing architecture of the sensor-cloud consists of four layers. The bottom-most layer is the network of physical sensor nodes, which sense various physical parameters of the environment. One or more physical nodes are grouped and virtualized to form virtual sensors. The system creates a virtual machine (VM) corresponding to a user requesting the system for Se-aaS. This virtual machine uses a virtual sensor group(VSG) to get the data from the underlying physical sensor network.

The same data can be useful for a particular user at different time instants. Consequently, for storing sensor data temporarily, the concept of cache is used. The existing architecture of the sensor cloud enabled with caching [2] has the following components: (a) *Internal Cache* (*IC*), and (b) *External Cache* (*EC*). The end-users request the sensor-cloud for sensed information through the Web-interface. EC acts as a buffer between the physical sensor nodes and the sensor-cloud, whereas IC acts as the intermediary memory between the end-users or applications and the virtual sensors. The system searches for data progressively in the IC and EC. However, if the concerned data are not found, re-senses from the physical sensor network, which is the bottom most layer of the sensor-cloud architecture. We propose a novel scheme, to ensure storing of cache of destroyed virtual machines. This idea is useful to provide cached data to the re-activated virtual sensors destroyed a while ago.

1.1 Motivation

The existing sensor-cloud architecture [11] proposes the use of IC and EC. However, in real implementation, the user may need a set of heterogeneous physical sensor nodes for different applications. Therefore, in such cases, storing the data of all heterogeneous sensors in the IC is not an efficient choice in the sensor-cloud architecture. Moreover, if all the virtual machines which use a particular virtual sensor get destroyed from the cloud, eventually, the data for that virtual sensor gets removed from the caches. However, if at a later instant of time, another user requests the virtual sensor comprising of the same set of physical sensors, which were used at earlier instants, then the history of this virtual sensor is not available to the system. This results in re-sensing data from the physical sensors. Since the physical sensors are energy-constrained, thereby, redundant sensing consumes energy unnecessarily, thereby degrading the overall system performance. Therefore, storing only the information required by the end-user in the IC for heterogeneous sensor nodes suffices. We devise a novel scheme to enable storing of cache of the destroyed virtual sensors, so that they may also be used to provide history of the data to the newly created virtual sensors, in case their configurations match. To address the issue of lack of cache memory for a new end-user, we introduce the concept of *Special Dynamic Caching* (*SDC*) in the sensor-cloud architecture. The proposed scheme maintain QoS in terms of delay.

ICDCN'18, January 04-07, 2018, India

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1.2 Contribution

In this work, we address the issue of dynamic caching in the sensor-cloud architecture in the presence of heterogeneous sensor nodes. The specific *contributions* of this work are as follows:

- We introduce the concept of *Special Dynamic Cache* (*SDC*) storage corresponding to each of the distinct end-users present inside the sensor-cloud.
- We consider the dynamic requirements of sensor data from different end-users, and thereafter, propose a scheme to assign memory dynamically from *SDC* to a particular end-user.
- The proposed scheme is evaluated rigorously, both mathematically and theoretically.

2 RELATED WORK

Sensor-cloud is a newly explored research topic. However, significant amount of research has already been undertaken on sensor-cloud. On the other hand, caching is an important issue in the domain of cloud computing. In this section, we discuss the prior work related to the fields of sensor-cloud and caching.

2.1 Sensor cloud

Yuriyama *et al.* [11] address the issues of *virtualization* in the sensor-cloud architecture. Along with virtualization, the authors propose the role of different actors such as sensor-owners and the end-user. Misra *et al.* [8] introduce the theoretical model of sensor-cloud and various operations which are needed to be undertaken inside it. The work shows significant improvement in energy consumption by using sensor-cloud rather than traditional WSN. For selecting an optimal data center (DC) in sensor-cloud, a scheme is proposed by Chatterjee *et al.* [3]. In order to select an optimal DC, the authors consider the QoS using migration cost. A framework is proposed by Neiat *et al.* [9], considering the spatio-temporal aspects of sensor-cloud. The proposed framework is mainly based on the A^* and *3D R-Tree* algorithms. Chatterjee *et al.* [1] propose a dynamic pricing scheme including hardware and infrastructure cost. The pricing scheme considers user satisfaction along with the other actors in the sensor-cloud.

2.2 Caching

In order to meet the demands and necessities of the end-users faster, the concept of caching emerged in the domain of cloud computing. Using suitable caching methods a cloud environment can improve the performance highly. Chockler *et al.*[4] propose *Cache-as-a-Service* as a feature of cloud, along with its advantages. Idachaba *et al.* [6] propose a new scheme in order to decrease cloud pollution and cloud monoploy prevalaent in the earlier architectures of caching. The authors claim that the cache-hit rate increases by using their scheme. Gordon et al.[5] shows a process of inter-VM shared cache memory to reduce latency in cache hits. According to the authors, the *nahanni memcached* reduces the delay in cache-read operations. Kantere et al. [7] propose a dynamic pricing scheme for cloud cache. The proposed scheme by the authors aims for user satisfaction. Only one work related to caching in sensor-cloud is discussed by Chatterjee *et al.* [2]. The work focus on the implementation of an *External* and *Internal* cache in sensor-cloud architecture. The authors calculate the optimal time for re-caching of external and internal caches, respectively.

2.3 Synthesis

The existing literature highlights different issues of sensor-cloud. Besides sensor-cloud, large number works considered by different authors discuss the various schemes for caching in multiple domains. However, only the work of Chatterjee and Misra [2] discuss the caching mechanism in sensor-cloud. The existing caching scheme in sensor-cloud [2] considers the existence of cache inside the sensor-cloud, which clears after the removal of a VM from the system. Consequently, this type of solution is not applicable, when there exist a huge number of sensor nodes in the system. The problem becomes challenging where it is required to tackle the situations where VMs may be destroyed in the cloud and there is a necessity to store the cache for the destroyed sensors.

3 PROBLEM SCENARIO

We consider a sensor-cloud architecture consisting of heterogeneous sensor nodes. Further, a set of homogeneous/ heterogeneous sensor nodes comprise of a virtual sensor. For each of the end-users, a VM is created. The VM is accompanied by an *SDC* which is different from the cache, generated automatically inside every VM once it is instantiated. The *SDC* is responsible for storing the sensor data only. The size and number of partitions of *SDC* is dependent on the end-users' requirements.

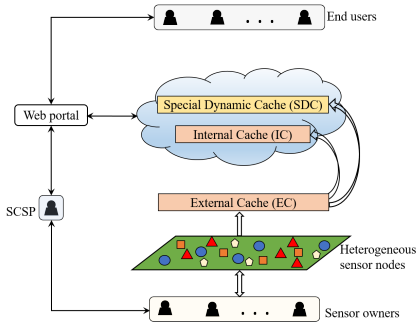


Figure 1: Sensor-Cloud architecture

3.1 Formal Definition of the Problem

The components involved in the sensor-cloud infrastructure are as follows:

- *Physical Sensor Nodes*: The physical sensor nodes represent the bottom most layer of the sensor-cloud. These sensor nodes sense physical parameters from the environment and transmit those to the upper layer. In our system, let $\mathcal{S}$ represent the set of all the physical sensor nodes. Thus, if the total number of physical sensor nodes is N, then the set can be represented as: $\mathcal{S} = \{s_{i,j}\}$, such that $1 \leq i \leq k, 1 \leq j \leq n_i$, where k represents the distinct types of physical sensor nodes present in sensor-cloud and n_i represents the number of sensors of the i^{th} type. Therefore,

$$\sum_{i=1}^{k} n_i = N$$

- *Virtual Machines*: Virtual Machine (VM) are instantiated inside the cloud. Let the total number of VMs present in the system be p. Then, the set of VMs is represented as: $\mathcal{V} = \{vm_1, vm_2, vm_3, \cdots, vm_p\}$
- *Internal Cache (IC)*: It consists of the most frequent data requested by end-users from the system. This is present within the sensor-cloud and is an intermediate layer between the end users and the virtual machines instantiated within the cloud [2].
- *External Cache (EC)*: It exists between the physical sensor network and sensor-cloud layer [2]. It stores data directly from the physical sensor nodes.

Fig. 1 represents the entire sensor-cloud architecture which we consider in this work.

3.2 Special Dynamic Cache

Definition 1. *Special Dynamic Cache (SDC), a set of two tuple $\langle id, t\rangle$ is an additional caching, which comprises of two parts – Reserve (SDC_R) and Active (SDC_A).*

In *Definition* 1, t denotes the type of the sensor data and id is the set of ordered pairs (*data*, *time*). The instant at which the request was made to the system is denoted by *time* and *data*, which contains the value either of a single sensor node or the combined value of the multiple homogeneous/heterogeneous sensor nodes, as per requirement. The *Active* part of SDC contains the cache of VMs currently active in the cloud. On the contrary, the *Reserve* part contains cache of VM that are destroyed from the system. Fig. 2 depicts the structure of SDC.

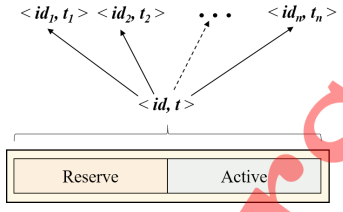


Figure 2: Special Dynamic Cache

4 SOLUTION APPROACH

Let the sizes of the two partitions be represented as $size(SDC_R)$ and $size(SDC_A)$ respectively. The total memory of SDC is assumed to be $\mathcal{M}$. Memory management of SDC is essential to achieve proper utilization of SDC. The active partition of the SDC requires to be maximized for storing the content of all possible active virtual machines in the cloud. The reserve partition also need to be maximized to an extent such that it can store cache of at least a single VM in itself. Thus the optimization function can be represented mathematically as,

$$maximize \;\; size(SDC_A) \tag{1}$$

subject to,

$$size(SDC_R) + size(SDC_A) = \mathcal{M} \tag{2}$$

$$\sum_{i=1}^{n} \hat{S}(x_i) \leq size(SDC_A) + size(IC) \tag{3}$$

$$size(SDC_R) \geq \sum_{j=1}^{m} \hat{S}(y_j) \tag{4}$$

$$C_{SDC} \times size(SDC_R) < min((C_M \times size(SDC_R)), C_{sensor}) \tag{5}$$

$$C_{IC} \times size(SDC_R) < min((C_M \times size(SDC_R)), C_{sensor}) \tag{6}$$

where
x_i: Any i^{th} active virtual sensor
n: The number of active virtual sensors
y_j: Any j^{th} virtual sensor with cache inside IC
m: The number of virtual sensors with their cache content in IC.
M: Memory of cloud

Definition 2. *sizeone($\hat{S}$) is the size of one ordered-pair, (data,time) for any virtual sensor inside SDC or IC.*

Equation (3) signifies the SDC_A and IC must hold at least one entry for each of the VMs currently active. The size of IC is determined keeping in mind the interval of re-caching using the process in the work [2]. On the other hand, SDC_R must be able to store at least one ordered pair for each of the virtual sensors currently with its cache in IC, which is represented in Equation (4). SDC is responsible to provide the history to a new VM too. Th cache content of a virtual sensor residing in IC is one of the most frequently used one. Therefore, there is a good chance the same virtual sensor will get activated again, even though it is unused currently. Thus, SDC_R must be able to store content of IC at least.

Definition 3. *Cost (C) is the overhead required to transfer data from one location to another.*

Cost, C, can be represented mathematically,

$$C = \{C_{i,j} : \text{cost from i to j, where i and j are two locations}\} \tag{7}$$

We consider the cost for the followings:
C_{SDC}: $i = SDC_R$ to $j = SDC_A$
C_{IC}: $i = SDC_R$ to $j = IC$
C_M: $i = \hat{M}$ to $j = EU$
C_{sensor}: $i = any s \in \mathcal{S}$ to $j = EU$

Equations (5) and (6) represent the cost of moving data within SDC or from SDC to IC should be less than the cost for retrieving / re-sensing. We use the *approximate solution approach* to simplify Equations (5) and (6). Also, assume that SDC_R stores cache-content of a single virtual sensor. To make calculations simpler, we assume that the cost of inter-SDC data exchange is less than both memory transfer and re-sensing.

Thus,

$$C_{SDC} \times size(SDC_R) < C_M \times size(SDC_R) \tag{8}$$

$$\implies C_{SDC} < C_M \tag{9}$$

$$C_{SDC} \times size(SDC_R) < C_{sensor} \tag{10}$$

The $size(SDC_R)$ is bounded within the range obtained from the Equations (4) and (10). As the size of the reserve and active partitions

are flexible (dynamic nature), the size of the reserve partition is in the specified range. The remaining part can be dedicated to SDC_A.

Just after the onset of the cloud system, the reserve partition can be made equal to the size of IC itself as $\hat{S}(x_i)$ for $x_i \in$ IC can be equal to IC itself at maximum. After that the sizes can be modified suitably and dynamically. Hence initially,

$$size(SDC_R) = size(IC) \text{ and } size(SDC_A) = \mathcal{M} - size(IC) \quad (11)$$

Based on *Importance Factor*, $\mathcal{I}$, the space allocation within SDC_A and IC is decided. $\mathcal{I}$ is defined for each distinct virtual sensor cache in the system. End-users who use the same group of virtual sensor nodes will be assigned the same cache storage unit in SDC and IC. In order to determine the value of $\mathcal{I}$ for a virtual sensor, the required parameters are – *number of VMs associated* ($\hat{N}$), the *number of physical sensor nodes associated* ($\hat{n}$), and the *number of end-users request for this virtual sensor* (f).

Proposition 1. *$\mathcal{I}$ is directly proportional to $\hat{N}$, $\hat{n}$ and f.*

Proof. Consider two cache partitions in SDC_A namely, c_1 and c_2. The set of VMs associated with c_1 be $\mathcal{V}_1 = \{vm_{i_1}, vm_{i_2}, \cdots, vm_{i_{\hat{N}_1}}\}$ and with c_2 be $\mathcal{V}_2 = \{vm_{j_1}, vm_{j_2}, \cdots, vm_{j_{\hat{N}_2}}\}$, where $1 \le i_1, i_2, \cdots, i_{\hat{N}_1} \le p$ and $1 \le j_1, j_2, \cdots, j_{\hat{N}_2} \le p$. Thus $\hat{N}_1 = |\mathcal{V}_1|$ and $\hat{N}_2 = |\mathcal{V}_2|$. Additionally, assume that $\hat{N}_1 > \hat{N}_2$.
We see c_1 supplies data to a larger set of VMs, consequently handling more end-user requests. To increase the cache hit ratio, c_1 must be made larger in size than c_2. Hence we conclude,

$$\mathcal{I} \propto \hat{N} \quad (12)$$

Similarly, for $\hat{n}$, let the number of physical nodes associated with c_1 be $\hat{n}_1$ and with c_2 be $\hat{n}_2$. Also, assume $\hat{n}_1 > \hat{n}_2$. Without caching, more data needs to be sensed for supplying user demands to the VMs associated with c_1 than with c_2. Therefore, $\mathcal{I}$ of c_1 is greater than c_2. Hence, we conclude,

$$\mathcal{I} \propto \hat{n} \quad (13)$$

Let the number of requests made to virtual sensors corresponding to c_1 and c_2 be f_1 and f_2 respectively. Further, let us assume, $f_1 > f_2$. as data in c_1 is requested more often, so allotting more space to c_1 helps to store more history and thus the accuracy of data returned from the system for a larger number of requests is improved. Thus,

$$\mathcal{I} \propto f \quad (14)$$

□

Definition 4. *Importance Factor ($\mathcal{I}$) : Importance Factor decides the importance level of virtual sensor cache present in either SDC or IC, and $\mathcal{I}$ is a function of $\hat{N}$,$\hat{n}$, and f.*

Mathematically,

$$\mathcal{I} = k\hat{N}\hat{n}f \quad (15)$$

where k is the proportionality constant.

Theorem 4.1. *The rate of change of the size of SDC_A with respect to the number of physical sensors in a virtual sensor is constant.*

Proof. Let v_i denote the i^{th} VS and $\hat{n_i}$ the number of physical sensors involved in v_i. As denoted earlier, $\hat{S_i}$ is the *sizeone* of the virtual node v_i.

As $\hat{n_i}$ is dependent of the end-user request, we have,

$$\hat{n_i} = g(r) \quad (16)$$

where r is the end-user request.

The sizeone of the node increases gradually, with the increase in the number of physical sensor nodes associated with a particular VS cache. Therefore, $\hat{S_i}$ is directly proportional to $\hat{n_i}$.

$$\hat{S}(x_i) \propto \hat{n_i} \quad (17)$$

Equation (17) is written as,

$$\frac{d\hat{S}(x_i)}{d\hat{n_i}} = \alpha_1 \quad (18)$$

where α_1 is the proportionality constant.

Thus, the rate of change of the size of SDC_A with respect to the number of physical sensors in a virtual sensor is $\frac{\partial SDC_A}{\partial \hat{n_i}}$.

Applying chain rule, we have,

$$\frac{\partial size(SDC_A)}{\partial \hat{n_i}} = \frac{\partial size(SDC_A)}{\partial \hat{S}(x_i)} \times \frac{\partial \hat{S}(x_i)}{\partial \hat{n_i}} \quad (19)$$

We assume the inequality in Equation (3) to be an equality and proceed. Thus,

$$\sum_{i=1}^{n} \hat{S}(x_i) = size(SDC_A) + size(IC) \quad (20)$$

Partially differentiating Equation (20) with respect to $S(\hat{x_i})$, we have,

$$\sum_{i=1}^{n} \frac{\partial \hat{S}(x_i)}{\partial \hat{S}(x_i)} = \frac{\partial (size(SDC_A) + size(IC))}{\partial S(\hat{x_i})} \quad (21)$$

$$\implies 1 = \frac{\partial size(SDC_A)}{\partial S(\hat{x_i})} + \frac{\partial size(IC)}{\partial S(\hat{x_i})} \quad (22)$$

The size of IC is a constant. Thus,

$$\frac{\partial size(SDC_A)}{\partial S(\hat{x_i})} = 1 \quad (23)$$

From Equations (19) and (23), we get,

$$\frac{\partial size(SDC_A)}{\partial \hat{n_i}} = 1 \times \frac{\partial S(\hat{x_i})}{\partial \hat{n_i}} \quad (24)$$

Using Equation (18) in Equation (24), we get,

$$\frac{\partial size(SDC_A)}{\partial \hat{n_i}} = \alpha_1 \quad (25)$$

Therefore, the rate of change of $size(SDC_A)$ with respect to the number of physical sensors ($\hat{n_i}$) in a virtual sensor is a constant. □

Proposition 2. *The size of SDC_A is independent of the number of VMs associated with a VS cache, provided the total number of active virtual sensors in the cloud remains constant.*

Proof. Equations (2)-(6) for determining the size(SDC_A) do not contain the number of VMs associated with a particular VS.

Let the number of VSs in the system be constant. If the number of VMs associated with any VS is increased, the size of the corresponding VS cache is not affected. Since SDC_A is a collection of VS caches, it also remains unaffected. □

4.1 Size of VS Cache within SDC_A and IC

$\mathcal{I}$ of a VS decides its size in SDC_A and IC. VS caches with higher $\mathcal{I}$ are alloted more space as compared to the lower one. Hence, cache allotment is dynamic, since $\mathcal{I}$ changes dynamically. The total size of SDC_A is fixed in the system. Therefore, the sum of all the VS cache sizes must be equal to the size of SDC_A alloted to the cloud system. Thus, we have,

$$\sum_{i=1}^{k} size(c_i) = size(SDC_A) \tag{26}$$

where c_i denotes a VS cache.

4.2 IC and EC in the System

Similar to SDC, IC too comprises of a 2-tuple, $\langle id, t \rangle$, where id is the actual data from different heterogeneous sensor nodes and t is the type of the sensor data. On the other hand, EC stores the single most recent reading for each of the virtualized physical sensor nodes present. EC consists a set of 3-tuples, $(sensor - reading, time, T)$, where $sensor-reading$ denotes the reading of the data, $time$ denotes the time to which the data corresponds to, and T is the type of physical sensor node. The sizes of IC and EC are fixed, and are determined using the method described in [2], in order to optimize the re-caching interval.

4.3 Data Request Handling in Sensor-Cloud using SDC

- *Responding to user requests*: On receiving requests through a Web-portal from the end-user, our system first searches for data progressively in IC, SDC_A and then EC. Wherever a hit is encountered, the data are returned. On failing to find data in the EC, it re-senses data from the sensor network and caches into SDC.
- *On destroying a VM*: When a VM is destroyed from cloud, system checks if the VS caches related to this VM are associated with any other VM. Otherwise, at this instant, the VS cache is moved to SDC_R.

Let c be the VS cache, which needs to be moved to SDC_R. In case, SDC_R is not full, cache c is included in SDC_R.

$$SDC_R = SDC_R \cup \{c_i\} \tag{27}$$

Otherwise, a cache block is replaced in SDC_R by c, using the LRU technique [10].

5 PERFORMANCE EVALUATION

5.1 Simulation Setup and Result

We evaluated our scheme based on the *cache hit percentage*, *energy consumption due to cache-miss* and *message overhead during cache-miss*. We analyze and discuss the results of the simulation setup. The plots in Fig. 3 show the cache hit percentage for 250 and 500 nodes, respectively. Figs. 4 and 5 show the energy consumption (nJ) during cache miss and message overhead (kb) for cache miss, respectively. The cache percentage in DCD is much higher than that for the existing architecture due to the presence of SDC. The energy consumption and message overhead in DCD during cache miss is substantially less. With the increase in the number of end-user requests, the energy consumption and overhead increase as more number of misses are encountered. Since in our proposed scheme, the number of cache misses get reduced, the message overhead is also significantly reduced compared to the existing architecture.

ALGORITHM 1: DCD

```
Inputs: VMID: ID of the Virtual Machine which the end-user requests
Output: Data requested by user
Begin
  Select the required VM with the VMID
  for each vnode in the requested VM
  if vnode.active == 1
    Search in IC, if found then flagIC = True
    if flagIC = False
      Search in SDC_A, if found flagSDC = True
      if flagSDC = False
        Search in EC, if found flagEC = True
        if flagEC = False
          Re-sense from physical sensor network
          Re-cache into EC, SDC_A and IC
        endif
      endif
    endif
  else
    Search in SDC_R, if found re-cache to SDC_A and IC
    Remove from SDC_R and set flagSDC = True
    if flagSDC == False
      Re-sense from physical sensor network
      Re-cache into EC, SDC_A and IC
    endif
  endif
End
```

Table 1: Simulation Parameters

Parameters	Values
Deployment Area	500 m × 500 m
Deployment	Uniform and random
Number of end-user requests	20 – 120

6 CONCLUSION

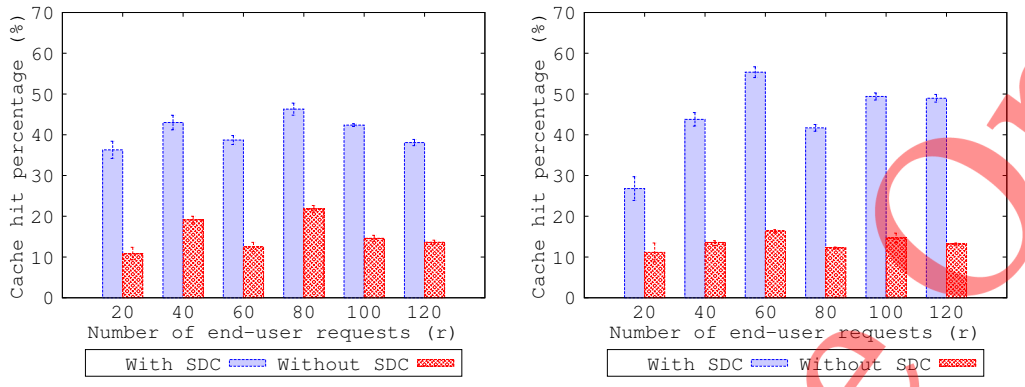
In this work we propose a new caching technique for heterogeneous virtual sensors in sensor-cloud to achieve caching for a newly created virtual sensor. In order to maintain QoS we introduce an additional caching component namely *Special Dynamic Cache* (SDC), which increases the number of cache hits, and reduces the energy consumption and message overhead considerably.
In the future, the work can be further extended considering the QoS of data transmission in the proposed model.

ACKNOWLEDGMENT

The first author of this work is partially funded by project file no. 9/81(1293)/17 sponsored by the Council of Scientific and Industrial Research (CSIR), Govt. of India. The second author of this work is partially supported by project file no. 184-17/2017(IC) sponsored by University Grants Commission (UGC)-UK India Education Research Initiative (UKIERI) Joint Research Programme (UKIERI-III). The authors would like to thank the anonymous reviewers for their constructive suggestions.

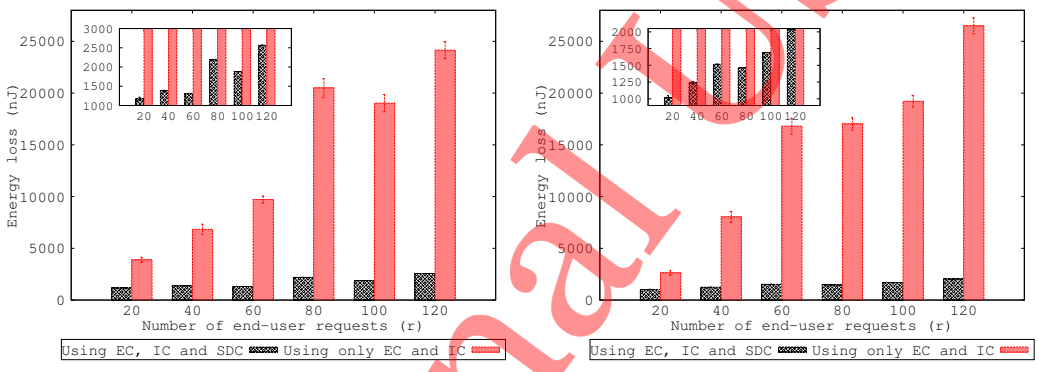
REFERENCES

[1] S. Chatterjee, R. Ladia, and S. Misra. 2015. Dynamic Optimal Pricing for Heterogeneous Service-Oriented Architecture of Sensor-cloud Infrastructure. *IEEE Transactions on Services Computing* 99 (2015).
[2] S. Chatterjee and S. Misra. 2014. Dynamic and adaptive data caching mechanism for virtualization within sensor-cloud. In *Proceedings of IEEE International*



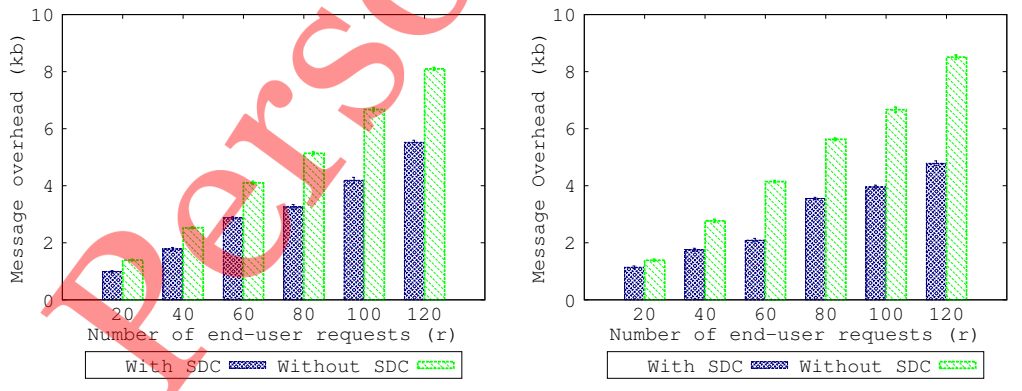
(a) Number of node = 250 (b) Number of node = 500

Figure 3: Cache hit percentage



(a) Number of node = 250 (b) Number of node = 500

Figure 4: Energy consumption



(a) Number of node = 250 (b) Number of node = 500

Figure 5: Message overhead

Conference on Advanced Networks and Telecommuncations Systems. 1–6.

[3] S. Chatterjee, S. Misra, and S. Khan. 2015. Optimal Data Center Scheduling for Quality of Service Management in Sensor-cloud. *IEEE Transactions on Cloud Computing* PP, 99 (2015), 1–1.

[4] Gregory Chockler, Guy Laden, and Ymir Vigfusson. 2010. Data Caching As a Cloud Service. In *Proceedings of the* 4^{th} *ACM International Workshop on Large Scale Distributed Systems and Middleware (LADIS '10)*. New York, NY, USA, 18–21.

[5] Adam Wolfe Gordon and Paul Lu. 2011. Low-latency caching for cloud-based web applications. *NetDB* (2011).

[6] U. Idachaba and F. Wang. 2015. A Community-Based Cloud Computing Caching Service. In *Proceedings of IEEE International Congress on Big Data*. 559–566.
[7] Verena Kantere, Debabrata Dash, Gregory Francois, Sofia Kyriakopoulou, and Anastasia Ailamaki. 2011. Optimal service pricing for a cloud cache. *IEEE Transactions on Knowledge and data engineering* 23, 9 (2011), 1345–1358.
[8] S. Misra, S. Chatterjee, and M. S. Obaidat. 2014. On Theoretical Modeling of Sensor Cloud: A Paradigm Shift From Wireless Sensor Network. *IEEE Systems Journal* PP, 99 (2014), 1–10.
[9] Azadeh Ghari Neiat, Athman Bouguettaya, Timos Sellis, and Zhen Ye. 2014. Spatio-temporal composition of sensor cloud services. In *Proceedings of IEEE International Conference on Web Services*. 241–248.
[10] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. 2008. *Operating System Concepts* (8th ed.). Wiley Publishing.
[11] M. Yuriyama and T. Kushida. 2010. Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing. In 13^{th} *International Conference on Network-Based Information Systems*. 1–8.