2018 IEEE 6th International Conference on Future Internet of Things and Cloud

PASHE: Privacy Aware Scheduling in a Heterogeneous Fog Environment

Kaneez Fizza*, Nitin Auluck*, Omer Rana† and Luiz Bittencourt‡
*Department of Computer Science and Engineering
IIT Ropar,India
(kaneez.fizza,nitin)@iitrpr.ac.in
†School of Computer Science & Informatics
Cardiff University, UK
ranaof@cardiff.ac.uk
‡University of Campinus, Brazil
bit@ic.unicamp.br

***Abstract*—Fog computing extends the functionality of the traditional cloud data center (cdc) using micro data centers ($mdcs$) located at the edge of the network. These $mdcs$ provide both computation and storage to applications. Their proximity to users makes them a viable option for executing jobs with tight deadlines and latency constraints. Moreover, it may be the case that these $mdcs$ have diverse execution capacities, i.e. they have heterogeneous architectures. The implication for this is that tasks may have variable execution costs on different $mdcs$. We propose PASHE (Privacy Aware Scheduling in a Heterogeneous Fog Environment), an algorithm that schedules privacy constrained real-time jobs on heterogeneous $mdcs$ and the cdc. Three categories of tasks have been considered: private, semi-private and public. Private tasks with tight deadlines are executed on the local mdc of users. Semi-private tasks with tight deadlines are executed on "preferred" remote $mdcs$. Public tasks with loose deadlines are sent to the cdc for execution. We also take account of user mobility across different $mdcs$. If the mobility pattern of users is predictable, PASHE reserves computation resources on remote $mdcs$ for job execution. Simulation results show that PASHE offers superior performance versus other scheduling algorithms in a fog computing environment, taking account of mdc heterogeneity, user mobility and application security.**

***Keywords*-Fog Computing; Cloud Computing; Cloud data center; Micro data center;**

I. Introduction

According to various studies, the billions of devices connected to the Internet [21] generate huge amounts of data. Analyzing this "big" data is a major challenge faced by researchers today, requiring scalable solutions. Typically, this data is sent to a cloud data center (cdc) for analysis. However, there could be significant communication delay from user devices to the cdc, leading to tasks missing their real-time deadlines [4], [5]. An example of such a task could be changing the course of an autonomous vehicle upon sighting an obstacle on the road. If this task is sent to a cdc and not completed by its designated deadline, there could be possible loss of life. Another issue is that users may not be comfortable in sending their personal and sensitive data to the cdc for execution.

Fog computing addresses these issues by proposing an architecture consisting of a number of micro data centers ($mdcs$), located at the network edge, in proximity to users [10], [13]. User devices can communicate directly with the $mdcs$, which provide limited computation and storage. Moreover, the $mdcs$ are connected to the cdc. A key insight here is that the tasks with tight deadlines and security constraints could be executed on the $mdcs$, whereas tasks that have loose deadlines and no security constraints could be sent to the cdc for execution.

In practice, $mdcs$ may have variable execution capacities, i.e., they may be "heterogeneous". For example, the mdc in the home network of a user may have lower capacity than the mdc in a work network. Likewise, while traveling from home to work, the user may interact with several $mdcs$, each with a different execution capacity. This "heterogeneity" in execution capacities adds an extra layer of complexity to the scheduling of tasks on the fog network.

In this paper, we present PASHE (Privacy Aware Scheduling in a Heterogeneous Fog Environment), a security-aware real-time scheduling algorithm for heterogeneous fog networks. This algorithm accounts for mdc heterogeneity, security constraints and performance constraints of applications. Specifically, private tasks can only be executed on the local mdc of users. Semi-private tasks may be executed on trusted remote $mdcs$, and public tasks on the cdc. Subject to meeting the security constraints, tasks with tight deadlines are executed on the local or trusted remote $mdcs$. On the other hand, tasks with loose deadlines are sent to the cdc for execution. The idea is to use the $mdcs$ for executing the security/privacy and deadline constrained tasks and to use the cdc for all other tasks.

A real-life application where this algorithm would be useful is "Traffic control and management in a smart city" – a scenario outlined by the Open Fog Consortium. Various vendors (e.g. Google, Uber) [24] have been testing self-driving cars with a number of on-board sensors, a global positioning system, LIDAR cameras etc. These vehicles will generate multiple terabytes of data every day, making it challenging to send all of this data to a cloud data center. It is therefore essential to use the edge capability for partial processing of this data. These cars will be communicating with roadside units to provide navigation support. A self driving car will use various sensors to enable decision support, e.g. stop or slow down or turn. Since these tasks are real-time in nature, and involve "local decision making", sending them to the cloud for processing may not be feasible,

978-1-5386-7503-8/18/$31.00 ©2018 IEEE
DOI 10.1109/FiCloud.2018.00055

due to the significant network latencies involved. It would be important to process these tasks directly on roadside units (or in-vehicle) – i.e. in fog devices ($mdcs$). Since the $mdcs$ are in proximity to the source of data, it is much more likely that the task deadlines will be met by executing them at the edge, without sending tasks to the cloud, or cdc. Moreover, this also reduces the overall data traffic in the network, and to the cdc. The vehicle must be capable of executing tasks autonomously when it cannot connect to the cloud, and some tasks could be sensitive in nature, for example, the actual path that the user is taking, or infotainment services being used (requiring use of an mdc). Other kinds of tasks that may be used for data analytics may be executed on the cdc, which would be more than capable of carrying out these high computation tasks. This paper is organized as follows. Section II discusses related work. The model, notation and problem formulation is described in section III. Section IV discusses the proposed algorithm PASHE. The simulation results are described in section V. Section VI concludes the paper.

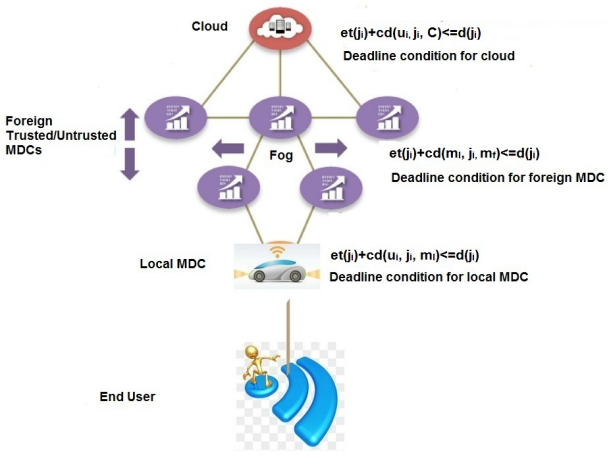

Figure 1: Edge Architecture

II. Related Work

In many cloud systems today, performance is heavily dependent on network connectivity and availability. This is especially true for real-time latency-sensitive applications [5]. Another issue is that data "belongs" to the cloud service provider, which may not be acceptable in privacy aware applications, e.g. maintaining patient records. Edge/fog computing is significant because it can make an impact in such scenarios, as it can provide much lower latencies by locating devices in proximity to users [13]. Moreover, these edge devices would be in control/ownership of users, resulting in local processing of private data. An extensive summary of edge computing has been described by Shu et al. in [22]. They explain several case studies that could benefit from edge technology, such as smart homes, smart cities, video analytics. They also describe mechanisms for handling security/ privacy, reliability, and programmability. Additional reviews of Edge computing can be found in [1], [2], [21].

In real-time systems, the defining property is that the output needs to be received within the specified deadline [4], [5]. There has been some work on resource allocation and scheduling in fog networks to achieve these objectives [7], [11]. In our previous work [10], we proposed RT-SANE, a security-aware, real-time scheduling algorithm for homogeneous fog networks. In this work, we consider mdc heterogeneity.

III. Modelling and Problem Formulation

In this section, we describe our model and notation. The edge–cloud architecture is shown in figure 1. At the lowest level are users with their devices. The top layer comprises of a cloud data center (cdc). The edge/fog layer sits between these, consisting of fog nodes/ cloudlets/ micro data centers ($mdcs$). Table I summarizes the notations. The system architecture consists of a single cloud data center (cdc) C, assumed to have sufficient capacity to execute all jobs. Users can purchase credits for executing their jobs on the cloud, e.g. from Amazon Web Services. In addition, connected to C is a set of micro data centers ($mdcs$), $M = \{m_1, m_2, m_3, ..., m_n\}$. A set of jobs $J = \{j_1, j_2, j_3, ..., j_k\}$ need to be executed on the $mdcs$ or the cdc. Each job $j_i \in J$ is dedicated to a user $u_i \in U$, where $U = \{u_1, u_2, u_3, ..., u_z\}$ is the set of users. Each job $j_i \in J$ is represented by a 3-tuple $< et(j_i), d(j_i), t(j_i) >$. Here, $et(j_i)$ represents the execution cost of job j_i, $d(j_i)$ is the job's deadline, and $t(j_i)$ is the job's security tag, representing the job's security constraints. We elaborate on these security tags later. We use the term job and task interchangeably. Each $m_x \in M$ can either be local (m_l) or remote/foreign (m_f). The implication for this is that each job j_i, based on its requirement, may either be executed on its local m_l, or on one of the remote $mdcs, m_f$, or on the cdc. Each $m_x \in M$ has a processing capability denoted by $cp(m_x)$. The $mdcs$ are heterogeneous in that they may have variable computing capacities. This means that a particular job $j_i \in J$ can take different times to execute on different $mdcs$. For the purpose of communication, a link is established between the cdc and various $mdcs$. A link between a cdc and an m_x has a bandwidth denoted as $bw(C, m_x)$. For the purpose of peer-to-peer communication, there also exist links between various $mdcs$. For example, for a particular user u_i, we may have a link between its local m_l and a foreign m_f, having a bandwidth denoted as $bw(m_l, m_f)$.

We consider jobs with three types of security tags: private (t_p), semi private (t_{sp}) and public (t_{pu}). Private jobs can execute only on the local m_l. Jobs with semi private and public tags are first sent to the m_l. If sufficient resources are not available on m_l, then they are migrated to a "preferred" foreign m_f. If semi private jobs do not get enough resources on a preferred foreign mdc, then they need to wait and can be resubmitted later, while public jobs can be moved to the cdc for execution, if they fail to get the required resources on a preferred foreign mdc.

The size of data transmitted by user u_i to m_l = $sc(u_i, j_i, m_l)$. The communication delay between user u_i and m_l is given by $cd(u_i, j_i, m_l)$. This delay can be repre-

Table I: Notations

C	Cloud Data Center
M	Set of micro-data centers
m_l, m_f	Local & foreign micro data centers
$R(m_x)$	Reliability of m_x
$UR(m_x)$	Unreliability of m_x
$l(i,j)$	Link between m_i & m_j
$R_l(i,j)$	Reliability of link between m_i and m_j
$UR_l(i,j)$	Unreliability of link between m_i and m_j
$T_l(i,j)$	Trust value of link between m_i and m_j
J	Set of all jobs/applications
U	Set of all users
u_i	Particular user $u_i \in U$
j_i	Specific job/task/application $\in J$
T_j	Set of tags assigned to jobs
t_p, t_{sp}, t_{pu}	Tags for private, semi-private, public jobs
$cp(m_x)$	capacity of m_x
$cp(C)$	capacity of C
$bw(u_i, m_l)$	bandwidth of link between user u_i and m_l
$bw(m_i, m_f)$	bandwidth of link between m_l and m_f
$ted(j_i, m_x)$	Total execution cost of job j_i on m_x
$et(j_i)$	Execution cost of an interaction of j_i
$st(j_i)$	Start time of job j_i
$ct(j_i)$	Completion time of job j_i
$d(j_i)$	Deadline for task j_i
$cd(m_l, j_i, m_f)$	Communication delay for j_i between m_l and m_f
$cd(u_i, j_i, m_l)$	Communication delay for j_i between m_l and u_i
$cd(u_i, j_i, C)$	Communication latency for j_i between u_i and C
k'	Number of jobs meeting their deadlines successfully
k	Total number of jobs submitted

sented by the following equation:

$$cd(u_i, j_i, m_l) = t + sc(u_i, j_i, m_l) + \frac{s(j_i)}{bw(u_i, m_l)} \quad (1)$$

Here, $bw(u_i, m_l)$ is the bandwidth of the communication link between the user u_i and their m_l. The size of the data transmitted by u_i is given by $s(j_i)$. The time to initialize the communication link is denoted by t. In addition, the cost of transferring the state of the job j_i is given by $sc(u_i, j_i, m_l)$. A similar relationship would hold for a foreign $mdcs$. For the sake of brevity, we omit these equations.

Each real-time job $j_i \in J$, needs to finish before its deadline.

$$st(j_i) + et(j_i) + cd(u_i, j_i, m_l) \leq d(j_i) \quad (2)$$

Here, $st(j_i)$ denotes the start time for job j_i. Jobs executing on their m_f must finish before their deadline as well.

$$et(j_i) + cd(m_l, j_i, m_f) \leq d(j_i) \quad (3)$$

The delay between m_l and m_f is modeled as:

$$cd(m_l, j_i, m_f) = t + sc(m_l, j_i, m_f) + \frac{s(j_i)}{bw(m_l, m_f)} \quad (4)$$

The delay between u_i and cdc is modeled as:

$$cd(u_i, j_i, C) = t + sc(u_i, j_i, C) + \frac{s(j_i)}{bw(u_i, C)} \quad (5)$$

The jobs executing on the cdc must also finish before their deadline.

$$et(j_i) + cd(u_i, j_i, C) \leq d(j_i) \quad (6)$$

An mdc must have sufficient spare capacity available to execute a job – referred to as the "Spare Capacity Condition", represented as:

$$et(j_i) \leq cp(m_x) - \sum et(j_x, m_x) \quad (7)$$

Here, $cp(m_x)$ stands for the execution capacity of m_x, j_x represents the set of all jobs currently assigned to m_x, and $\sum et(j_x, m_x)$ represents the sum of execution costs for all jobs already allocated to m_x. This equation needs to hold $\forall j_i \in J, \forall m_x \in M$. Here, m_x can be either m_l, or m_f.

In the proposed algorithm, if m_l does not have spare capacity available, or if the user is mobile, jobs may be executed on any of the heterogeneous foreign $mdcs$. Moreover, m_f may or may not be reliable. In order to come up with a set of reliable $mdcs$, we propose the following approach.

In case the local mdc of job j_i does not have sufficient spare capacity available (eq. 7), then, the job j_i may be executed on a foreign mdc. For this, the set of available foreign $mdcs$ are arranged in decreasing order of their bandwidth. One of the $mdcs$ from this set is selected for execution, as follows.

Each m_x has a reliability $R(m_x)$ and an unreliability $UR(m_x)$ value associated with it, which ranges between 0 and 1, i.e. $0 \leq R(m_x) \leq 1$. Intuitively, $UR(m_x) = 1 - R(m_x)$, however, these may also be defined as a fuzzy membership function. The reliability of a link between m_l and m_f is denoted by $T_l(m_i, m_f)$. Higher the value of T_l, more trusted is the foreign mdc. In this work, we assume a threshold value of 0.5, i.e for a particular m_l, if $T_l(m_l, m_f) \geq 0.5$, then m_f is trusted with respect to that particular m_l, otherwise it is untrusted. The Reliability and Unreliability values of a link $l(m_l, m_f)$ between a local m_l and one of its foreign m_f can be calculated by using the reliability and unreliability values associated with m_l and m_f as shown in eq. (8) and eq. (9):

$$R_l = \frac{R_{ml} + R_{mf}}{2} - \frac{R_{ml} + UR_{mf}}{2} + \frac{R_{mf} + UR_{ml}}{2} \quad (8)$$

$$UR_l = \frac{UR_{ml} + UR_{mf}}{2} - \frac{R_{ml} + UR_{mf}}{2} + \frac{R_{mf} + UR_{ml}}{2} \quad (9)$$

Finally, the trust value of the link between m_l and m_f, $T_l(m_l, m_f)$ is calculated as given below by using link reliability and unreliability values obtained in eq. (8) and eq. (9):

$$T_l(m_l, m_f) = \frac{R_l}{R_l + UR_l} \quad (10)$$

If, for m_l, $T_l(m_l, m_f) \geq 0.5$, then m_f is trusted for m_l.

$$T_l(m_l, m_f) >= 0.5 \quad (11)$$

The main performance metric that we consider in this work is Success Ratio (SR). This is the ratio of the number of jobs that meet their deadlines (k') and the total number of jobs considered for execution (k), in other words, $SR = \frac{k'}{k}$.

Table II: Mapping of Jobs to $mdcs$ or cdc

	m_l	$m_f(r_j)$	$m_f(ur_j)$	cdc
t_p	Y	N	N	N
t_{sp}	Y	Y	N	N
t_{pu}	Y	Y	Y	Y

The problem we solve in this paper may be stated as follows: Given a set of jobs J, a set of heterogeneous $mdcs$ M with different execution capacities, and trust levels, and a cdc, C, maximize $SR = \frac{k'}{k}$, while ensuring that jobs get executed on their preferred $mdcs : m_f(t_1), m_f(t_2), ...etc.$, according to table II.

IV. PASHE Algorithm

In the proposed $PASHE$ algorithm, all $mdcs$ are heterogeneous, meaning that they may have different processing capabilities, denoted as $cp(m_x)$. The implication is that each job may take a different amount of time to execute on a different mdc. The available $mdcs$ are divided into two sets: M_F, the set of all foreign $mdcs$, and M_L, the set of all local $mdcs$. From the set of foreign $mdcs, M_F$, we obtain the set of trusted foreign $mdcs$, given by $M_F(T)$. All the local $mdcs$ in the set M_L are assumed to be reliable. A set of trusted foreign $mdcs, M_F(T) = \{m_f(t_1), m_f(t_2), m_f(t_3), ..., m_f(t_d)\}$. This set consists of the $mdcs$ that are trusted, i.e. the $mdcs$ for which the value of $T_l \geq 0.5$. For a particular job j_i associated with a particular user u_i, the set of trusted foreign $mdcs, M_F(T)$ is sorted in decreasing order of bandwidth from the user's m_l, such that $bw(m_l, m_f(t_1)) \geq ....bw(m_l, m_f(t_d))$. This implies that $cd(m_l, j_i, m_f(t_1))..... \leq cd(m_l, j_i, m_f(t_d))$. This sorted list of trusted $mdcs$, then becomes $M_F(T)_{sorted}$. The first member of this list becomes the most preferred foreign mdc for job execution, the second member becomes the next most preferred foreign mdc, and so on.

Algorithm 1 PASHE

Calculate ct, cd, $\forall j_i \in J$.
Populate Queue Q with tag t_p, t_{sp} *and* t_{pu}
Arrange Q in increasing order of deadlines
$\forall j_i$ *with tag* t_p:
if (j_i satisfies $st(j_i) + et(j_i) + cd(u_i, j_i, m_l) \leq d(j_i)$ && $et(j_i) \leq cp(m_x) - \sum et(j_x, m_x)$ on m_l) **then**
    schedule j_i *on local mdc* m_l.
else
    re-submit job later
$\forall j_i$ *with tags* t_{sp} *or* t_{pu}:
if (j_i satisfies $st(j_i) + et(j_i) + cd(u_i, j_i, m_l) \leq d(j_i)$ && $et(j_i) \leq cp(m_x) - \sum et(j_x, m_x)$ on m_l) **then**
    schedule j_i *on local mdc* m_l.
else
    $Preferred - MDC()$.
Calculate SR ratio ($\frac{k'}{k}$).

Algorithm 2 Preferred-MDC( )

Generate list of trusted foreign $mdcs, M_F(T)_{sorted}$.
$\forall j_i$ *with tag* t_{sp}:
$spFinished = 0$;
for (i=1 to length(List) **do**.
    if $m_f(t_j)$ & m_l satisfy $T_l(m_l, m_f) >= 0.5$ && $m_f(t_j)$ satisfies $et(j_i) + cd(m_l, j_i, m_f) \leq d(j_i)$ && $et(j_i) \leq cp(m_x) - \sum et(j_x, m_x)$ **then**
        schedule j_i *on* $m_f(t_j)$
        $spFinished = 1$
        $break$;
    else
        i++;
if ($spFinished == 0$) **then**:
    re-submit job later.
$\forall j_i$ *with tag* t_{pu}:
$puFinished = 0$;
for (i=1 to end of List L) **do**.
    if $m_f(t_j)$ satisfies $et(j_i) + cd(m_l, j_i, m_f) \leq d(j_i)$ and $et(j_i) \leq cp(m_x) - \sum et(j_x, m_x)$ **then**
        schedule j_i *on* $m_f(t_j)$
        $puFinished = 1$*;*
        break;
    else
        i++;
if ($puFinished$) $== 0$ **then**:
    if j_i satisfies $et(j_i) + cd(u_i, j_i, C) \leq d(j_i)$ **then**
        schedule j_i *on cdc.*
else
    return;

In PASHE, the completion time (ct) and communication delay (cd) is calculated for each job $j_i \in J$. Next, all jobs with security tags t_p, t_{sp}, and t_{pu}, are sorted in an increasing order of their deadlines [4], and added to a queue Q. All private jobs are scheduled on their local m_l. If the local mdc does not have spare capacity available, the job needs to be re-submitted later. The proposed algorithm tries to execute semi private and public jobs on the local m_l. However, if the m_l does not have a sufficient spare execution capacity available, then the jobs are sent to their preferred foreign $m_f(t_1)$. The preferred foreign $m_f(t_1)$ is selected from the set of trusted foreign $mdcs, M_F(T)$ for that particular local m_l on the basis of their link reliability value T_l. A semi private or public job is sent to the preferred $m_f(t_1)$, which has the highest link bandwidth with m_l. If the preferred trusted $m_f(t_1)$ is not available, semi private jobs are sent to the next preferred, trusted foreign $m_f(t_2)$, and so on. The public jobs are sent to the cloud data center cdc.

V. Results and Discussion

A. Simulation setup and Parameters

We consider a simulation with 12 users, each user u_i has its own dedicated job j_i. The execution costs of jobs range between (500-5500) MIPS. The number of $mdcs$ that have been considered here is 3, each with a different computing capacity, which range between (400-2200) MIPS. It is assumed that a single cdc with a computing capacity of 44800 MIPS is present.

All simulations have been run on a modified iFogSim simulator which is capable of modeling the characteristics of heterogeneous $mdcs$ and a cdc [12]. The iFogSim simulator was found to be most suitable, as it is specially meant for resource management in the Fog environment. Parameters such as latency, network congestion, energy consumption and cost have been considered. All iFogSim simulations have been run on a machine with an Intel Xeon 2.40 GHz Processor and 4GB of RAM. In iFogSim, a class with the name *HeterogeneousMdcs* has been created. This class contains 12 different independent applications, each having one module. The deadline of each application along with its $MIPS$ capacity requirement has been declared in this class. The capacity of the cdc and the communication delays have also been specified. By varying the capacity of the $mdcs$ for different iterations of the simulation in the *HeterogeneousMdcs* class, a heterogeneous execution environment has been created. The *UpdateAllocatedMips* function in the *FogDevice* class assigns a MIPS load to the various modules. This has been done in order to implement *FCFS* along with the existing time shared scheduling method in iFogSim. Next, a job priority has been created, which contain jobs in a non decreasing order of their deadlines or in $FCFS$ order. The implementation keeps a check on whether an application completes its execution or not. If so, then the application is removed from the queue and the remaining $MIPS$ of the mdc is allocated to some other job. In order to evaluate the performance of all given scheduling algorithms, the following metrics have been used in the simulation:

1) Success Ratio (*SR*): This is defined as $(\frac{k'}{k}) * 100$, i.e. the percentage of the number of jobs meeting their deadlines to the total number of jobs submitted for scheduling.
2) Heterogeneity Factor (*HF*): defined as $\frac{c_p(m_{fastest}) - c_p(m_{slowest})}{average(c_p(m_z))}$. Here, $c_p(m_{fastest})$ is the execution speed of the fastest mdc, $c_p(m_{slowest})$ is the execution speed of the slowest mdc, and $average(cp(m_z)$ is the average of the execution speeds of all $m_z \in M$. In the simulations, the HF value has been increased from 0 to 1.6, each time by a value of 0.2. This varies the level of heterogeneity of the execution capacities of the $mdcs$.
3) Deadline Factor (*DF*): This metric specifies the range over which the job deadlines are varied. The interpretation is that a low value for DF implies that the overall deadlines in the system are "tight". Likewise, high DF values imply "loose" deadlines. A lower bound for the job deadline $d(j_i)_{LB}$ was calculated, equal to $ect(j_i)$. Here, $ect(j_i)$ is the earliest time that job j_i can start. Next, the average deadline value of the system d_{avg} was calculated. This value was then multiplied by a factor of 1 to 4 to get a range of deadline factor values.

1) Effect of mdc heterogeneity on Performance: We investigate the effect of *mdc* heterogeneity (HF) on Success Ratio (SR). In *cdc-only*, all jobs are sent to the cdc for execution, leading to considerable communication delay. On the other hand, in *PASHE*, jobs are first sent to the $mdcs$. For the simulation, we consider delay between an mdc and a user device to be 2ms, and between the cdc and a user device to be 100ms. In *cdc-only*, public jobs are sent to the cdc, while private and semi private jobs do not get any chance for execution. Therefore, $mdcs$ do not have any role to play here. Due to the large communication delay between the user and the cdc, only public jobs with loose deadlines can complete their execution. Hence, the performance results in a straight line (with 25% jobs meeting their deadlines). PASHE results in an increased *SR* because all private jobs are executed on the local m_l, semi private jobs are sent to either the local m_l, or to a reliable foreign mdc. In the next simulation, the performance of *PASHE* is compared with $FCFS$ in a heterogeneous Fog Environment. The number of *mdcs* has been fixed to 3, with computing capabilities of *mdcs* ranging from (350–2700) MIPS. The results for this simulation are shown in Figure 3.

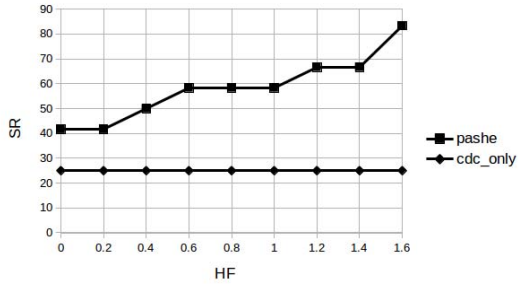

Figure 2: Effect of HF on SR

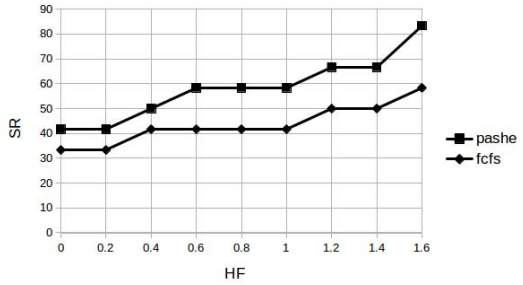

Figure 3: Effect of HF on SR

The *HF* has now increased from 0 to 1.6, and its effect on SR can be observed – increasing *HF* also leads to an increase in *SR* for both *FCFS* and PASHE. In both algorithms jobs are sent to *mdcs* as well as the *cdc* for execution. It is seen that by varying the computing capability of *mdcs*, possibly by increasing it instead of fixing it to some constant value, a larger number of jobs are able to meet their deadlines. However, PASHE demonstrates a better performance than *FCFS*, as it schedules jobs with earlier deadline first unlike *FCFS*. If jobs arrive at the same time, then they are scheduled according to their job *ids*.

2) Effect of Deadline Factor on Performance: In this simulation we compared the performance of $FCFS$ and $PASHE$ by varying the Deadline Factor (DF). The number of jobs considered is 12. Four $mdcs$ with different computing capacities $cp(m_x)$ were considered. The communication

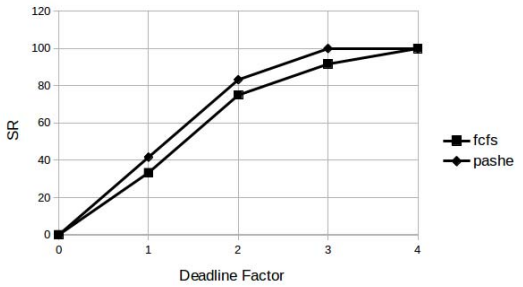

Figure 4: Effect of DF on SR

delay between a user device and an mdc was kept at 2ms, and between the cdc and user device as 100ms. The result is shown in Figure 4. We observe that performance (SR value) increases when DF is increased. This is because increasing the DF reduces the constraint on completion time, and a larger number of jobs are able to meet their deadlines. We can also observe that PASHE results in a higher SR as compared to $FCFS$, as it employs a better heuristic: Earliest Deadline First (EDF) which schedules applications according to their deadlines, ensuring that more jobs are able to finish execution before their deadlines. In $FCFS$, jobs are scheduled in the order in which they arrive, irrespective of their deadlines. Interestingly, after a certain threshold DF value, both the algorithms result in an *SR* as high as 100%. At this instant, the deadlines become so loose that all the jobs finish well before time, irrespective of the scheduling algorithm used.

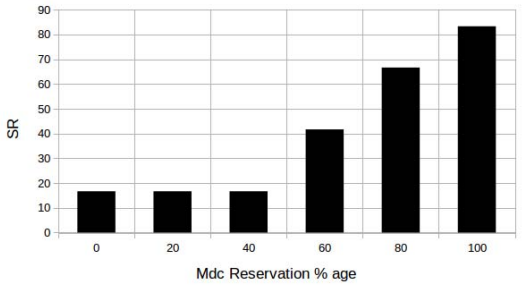

Figure 5: Effect of $ResourceReservation$ on SR

3) Effect of Resource Reservation on Performance: In this simulation, we studied the effect of resource reservation on system performance. The authors in [23] observe that in their study, it is possible to predict the mobility patterns of users with as much as 93% accuracy. Moreover, they also observe the accuracy of this prediction is largely independent of the distance that the users cover. For example, while traveling from home to work, users may take pretty much the same path everyday. Our proposed idea is that if the mobility pattern of a user is fixed, this can be used to reserve some portion of the computation capacity of the foreign $mdcs$ in the users' path. The proposed algorithm $PASHE$ with this reservation module added, becomes $PASHE\_R$. In this simulation, we considered three $mdcs$ with computing capacity values $cp(m_x)$ of 780, 2090, and 350 MIPS respectively. The jobs were a mix of private jobs, semi-private jobs and public jobs, but the ratio of semi-private jobs was kept higher. Here, 50% of the jobs were semi-private while private and public jobs were 25% each. The MIPS requirements of jobs range from (350–5500). The public jobs are allowed to execute on foreign $mdcs$ if the $mdcs$ have sufficient spare capacity available . In this case, semi private jobs may miss their deadlines on their preferred $mdcs$. This is because public jobs are computationally expensive, and may consume all the computing capacities of foreign $mdcs$, and still fail to meet their deadline. Therefore, it makes sense to reserve some mdc capacity for semi-private jobs only. Figure 5 shows that reservation of 0%, 20%, and 40% of the mdc computing power gives a constant performance without much improvement. This can be explained as follows. For cases in which a smaller percentage of mdc computing power is reserved for semi-private jobs, resource hungry public jobs occupy the $mdcs$ for the most part, and hence, deny the $mdcs$ to the semi-private jobs. In fact, even the public jobs are unable to meet their deadlines. However, as the amount of resource reservation for semi-private jobs is increased, a larger number of them can finish their execution before their deadlines, leading to a higher SR value.

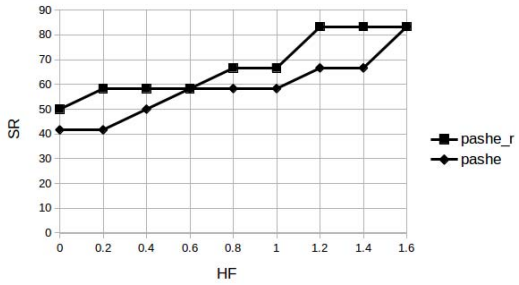

Figure 6: Effect of HF on SR

The next simulation compares the performance of $PASHE\_R$ versus simple $PASHE$ in a heterogeneous fog environment. The HF value has been varied from 0 to 1.6. The number of jobs considered is 12 and the number of $mdcs$ considered is 3. The result for this simulation is shown in Figure 6. In this scenario, 35% of the computation capacity of the foreign $mdcs$ has been reserved for semi-private jobs. This ensures that $PASHE\_R$ results in a larger number of semi-private jobs being able to meet their deadlines than would have been possible without reservation. This translates to a higher *SR*. Also note that this reservation causes more of the high computation public jobs to be sent to the cdc. In comparison, the *SR* values for $PASHE$ are lower, due to the absence of reservation for semi-private jobs. This leads to public jobs blocking the $mdcs$, causing the semi-private jobs to miss their deadlines. Hence, the SR values are lower versus the values obtained by $PASHE\_R$.

The next simulation compares the performance of $PASHE\_R$ and $cdc-only$. The simulation setup remains the same as the previous case. The results are shown in Figure 7. From the figure, we observe that $PASHE\_R$ offers a higher SR than *cdc-only*, partly due to employing

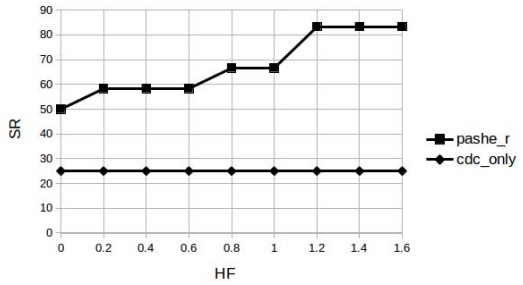

Figure 7: Effect of HF on SR

the $mdcs$ for job allocation, and partly due to the 35% reservation for semi private jobs. This results in a larger number of jobs completing their execution successfully. On the other hand, the *cdc-only* algorithm does not employ the edge for scheduling and all jobs are sent to the cdc, resulting in low SR values. Interestingly, we observe that increasing HF has no effect on SR, which is shown in terms of a flat line in Figure 7. This is because HF corresponds to the heterogeneity of the $mdcs$, which are not used to schedule jobs in *cdc-only*.

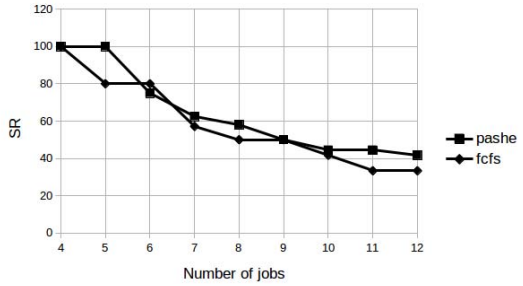

Figure 8: Effect of number of jobs on SR

4) Effect of Number of jobs on Performance: The goal of this simulation is to analyze the effect of increasing the system load (i.e. the number of jobs). The mdc computing capacities have been taken as 500, 1500, 2200 MIPS respectively. Initially, we consider 4 jobs in the system. Three "small" jobs with capacity requirement of 240, 500 and 700 MIPS execute on 3 different $mdcs$, and the one "large" job with capacity requirement of 3600 MIPS executes on the cdc. The results for this simulation are shown in figure 8. We observe that $PASHE$ performs well, in general, versus $FCFS$. As the number of jobs increases from 4 to 12 with computing requirements of 150, 110, 180, 4400, 3500, 200, 5500, and 257 MIPS respectively, the performance (SR value) degrades. However, $PASHE$ still performs better than $FCFS$. This can be attributed to the fact that $PASHE$ follows the EDF heuristic which schedules job according to their deadlines, while $FCFS$ schedules jobs in the order that they arrive in the ready queue. Furthermore, from Figure 8, we observe that when the number of jobs is less, the $mdcs$ are capable of executing all jobs successfully but when the number of jobs increases, then the $mdcs$ fail to execute them because of their limited resources.

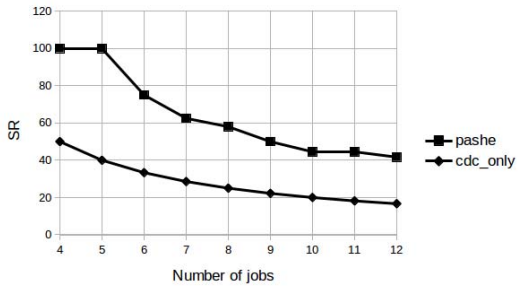

Figure 9: Effect of number of jobs on SR

The next simulation in this section compares the performance of PASHE and *cdc-only*. The setup remains same as in the previous simulation. The results are depicted in Figure 9. Initially, when only four jobs are present, $PASHE$ offers an SR value of 100%, as all the jobs meet their deadline, because the three $mdcs$ provide sufficient execution resources. In *cdc-only*, all jobs are sent to the cdc, and not to the $mdcs$, so many of them fail to meet their deadlines. This is the case even when the number of jobs is less. This is due to the large communication delay between the user jobs and the cdc. When the job count is increased from 4 to 12, performance of both the algorithm degrades. This is because of the additional computation load placed. As the $mdcs$ have limited computing capacity, this increased load leads to more deadline misses, and a reduction in the SR value. As we can observe in figure 9, PASHE gives superior performance compared to *cdc-only*, even when job count increases. This is due to the fact that $PASHE$ employs the $mdcs$ for scheduling jobs. Only when the capacity of the $mdcs$ is exhausted, are the jobs sent to the cdc.

ACKNOWLEDGEMENT

This research was partially funded by the European Commission H2020 programme under grant agreement no. 688941 (FUTEBOL), as well as from the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through Brazilian National Research and Educational Network (RNP) and CTIC. LFB would like to thank CNPq and CAPES for the financial support.

VI. CONCLUSION

In this work, we propose $PASHE$, a novel algorithm for resource scheduling in fog environments, supporting edge $mdcs$ which are "heterogeneous" and able to take user mobility into account. The proposed approach is able to take job requirements (security and completion deadline) into account, to identify the most suitable location to execute the job, i.e. a local vs. remote mdc or a centralized cdc. The proposed algorithm also allows for reservation of bandwidth on remote $mdcs$. The approach has been validated through the widely used fog simulator, iFogSim, where an extension has been provided to model heterogeneous $mdcs$. The outcome of this work can be used to support capacity planning of $mdcs$ at the network edge.

REFERENCES

[1] Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are, Cisco White Paper, 2015.

[2] A. V. Daesterjerdi and R. Buyya, Fog Computing: Helping the Internet of things to realize their potential, IEEE Computer, vol. 49, no. 8, pp. 112-116, 2016.

[3] D. Evans, The Internet of Things, how the next evolution of the Internet is changing everything, Cisco White Paper, April 2011.

[4] C. L. Liu and J. W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the ACM, vol. 20, no. 1, 1973, pp. 46-61.

[5] J. W. S. Liu, Real-Time Systems, Prentice Hall, April 2000, 592 pages.

[6] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, INDICES: Exploiting edge resources for performance aware cloud hosted services, First IEEE/ACM International Conference on Fog and Edge Computing, Madrid, Spain, May 14, 2017, pp. 75-80.

[7] M. I. Naas, P. R. P. Orange, J. Boukhobza, L. Lemarchand, iFogStor: an IoT data placement strategy for fog infrastructure, First IEEE/ACM International Conference on Fog and Edge Computing, Madrid, Spain, May 14, 2017, pp. 97-104.

[8] D. Klusacek, B. Parak, G. Podolnikova, A. Urge, Scheduling scientific workloads in private cloud: problems and approaches, The Tenth IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Austin, USA, December 2017, pp. 9 - 18.

[9] C. B. Hauser, S. R. Palanivel, Dynamic network scheduler for cloud data centres with SDN, The Tenth IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Austin, USA, December 2017, pp. 29 - 38.

[10] A. Singh, N. Auluck, O. Rana, A. Jones, S. Nepal, RT-SANE: Real time security aware scheduling on the network edge, The Tenth IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Austin, USA, December 2017, pp. 131 - 140.

[11] L. Bittencourt, J. D. Montes, R. Buyya, O. Rana, M. Parashar, Mobility aware application scheduling in fog computing, IEEE Cloud Computing, 2017, pp. 34 - 43.

[12] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of things, edge and fog computing environments, CORR abs, vol. 1606.02007, 2016. [Online]. Available: http://arxi.org/abs/1606.02007.

[13] M. Satyanarayanan, G. Lewis, E. J. Morris, S. Simanta, J. Boleng, K. Ha, The role of cloudlets in hostile environments, IEEE Pervasive Computing, October, 2013, pp. 40-49.

[14] M. Satyanarayanan, P. Bahl, R, Caceres, and N. Daview, The case for vm-based cloudlets in mobile computing, IEEE Pervasive Computing, vol. 8, issue 4, October 2009, pp. 14 - 23.

[15] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh and R. Buyya, Fog computing: principles, architectures, and applications, Internet of Things: Principles and Paradigms, R. Buyya and A. Dastjerdi (eds), Morgan Kaufmann, ISBN: 978-0-12-805395-9, Burlington, Massachusetts, USA, May 2016.

[16] I. Petri, O. Rana, J. Bignell, S. Nepal, and N. Auluck, Incentivising Resource Sharing in Edge Computing Applications, The 14th International Conference on Economies of Grids, Clouds, System and Services (GECON 2017), Anglet, France, September 19-21, 2017

[17] D. Klusacek, and B. Parak, Analysis of mixed workloads from shared cloud infrastructure, the Twenty First Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), June 2017, Orlando, USA.

[18] S. Di, D. Kondo, and W. Cirne, Characterization and comparison of cloud versus grid workloads, International Conference on Cluster Computing (CLUSTER), September 2012, Beijing, China, pp. 230 - 238.

[19] A. Ometov, J, Kannisto, S. Andreev, Y. Koucheryavy, Safe, secure executions at the network edge: Coordinating Cloud, Edge, and Fog Computing, IEEE Software, vol 35, issue 1, 2017, pp. 30 - 37.

[20] W. Zhang, Z. Zhang, H. C. Chao, Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management, IEEE Communications, December 2017, pp. 60 - 67.

[21] A. Nordrum, Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated, IEEE Spectrum, August, 2016.

[22] W. Shu, J. Cao, Q. Zhang, Y. Li, and L. Xu, Edge computing: vision and challenges, IEEE Internet of Things Journal, vol. 3, no. 5, October, 2016, pp. 637-646.

[23] C. Song, Z. Qu, N. Blumm, and A. Barabasi, Limits of Predictability in Human Mobility, Science, vol. 327, February 2010, pp. 1018 - 1021.

[24] M. Gerla, "Internet of vehicles: From intelligent grid to autonomous cars," 2017 International Conference on Networking, Architecture, and Storage (NAS), Shenzhen, China, 2017, p. 1.