LSTM-based Service Migration for Pervasive Cloud Computing

Haifeng Jing
School of Computer and Communication Engineering
China University of Petroleum
Qingdao, China
jinghaifeng@upc.edu.cn

Yafei Zhang
Department of Software Engineering
China University of Petroleum
Qingdao, China
z17070640@s.upc.edu.cn

Jiehan Zhou
University of Oulu
Finland.
jiehan.zhou@oulu.fi

Weishan Zhang
Department of Software Engineering
China University of Petroleum
Qingdao, China
zhangws@upc.edu.cn

Xin Liu
School of Computer and Communication Engineering
China University of Petroleum
Qingdao, China
lx@upc.edu.cn

Guizhi Min
Engineering Technology Research Institute
Huabei Oilfield Company, PetroChina
Renqiu, China
cyy_mgz@petrochina.com.cn

Zhanmin Zhang
Engineering Technology Research Institute
Huabei Oilfield Company, PetroChina
Renqiu, China
cyy_zhangzm@petrochina.com.cn

***Abstract*—Service migration in pervasive cloud computing is important for leveraging cloud resources to execute mobile applications effectively and efficiently. This paper proposes a LSTM (long and short-term memory model) based service migration approach for pervasive cloud computing, i.e., LSTM4PCC, which supports an accurate prediction of cloud resources. LSTM4PCC makes a prediction for cloud resource availability with a LSTM network and establishes a service migration mechanism in order to optimize service executions. We evaluate LSTM4PCC and compare it with the ARIMA (AutoRegressive Integrated Moving Average) approach in terms of prediction accuracy. The results show that LSTM4PCC performs better than ARIMA.**

***Index Terms*—Pervasive Cloud Computing, Service Migration, Machine Learning, LSTM**

I. Introduction

Pervasive cloud computing (PCC) is an emerging trend associated with powerful cloud computing and ubiquitous mobile computing, allowing mobile devices to hook up with their neighbor cloud services such as data storage and computation. With the integration of multimedia applications, mobile devices most commonly will provide cloud multimedia services for generating, editing, processing, and searching multimedia contents, such as images, video, audio, and graphics via the cloud [1].

Task migration in PCC expects to leverage capabilities of mobile devices by migrating the above-mentioned multimedia tasks to cloud nodes. We use task migration and service migration interchangeably in this paper. Service migration usually has the following steps [2]: looking for available resources and services; monitoring environment context; making decision for tasks scheduling and remote execution control.

Task migration architecture specifies the environment in which mobile devices are connected to and interact with cloud resources. Task migration architecture can be classified into four types: public cloud such as MAUI [3], Clonecloud [4], Cloudlet [5], Cloud of Mobile Network Operator [6], Cloud of mobile devices such as Hyrax [7] and Edgar [8]. To ensure efficient mobile application executions, migration scheduling is needed to adapt to dynamic changes in PCC.

There are static or dynamic migration scheduling algorithms [2]. Static task migration approaches include Wishbone [9], VM placement/migration [10], Clonecloud [4], Task-resource Scheduling [11], Task Scheduling with DVFS [12]. Dynamic task migration approaches include Parametric Analysis [13], (k + 1) Multi-Constraint Partitioning Algorithm [14], MAUI [3], Darwin [15], EOA [16], Cloud-Vision [17], LALTM [18], EMSO [19], MDP migration [20], OSGi-PC [21], Mobiles on Cloud Nine [22], Jade [23], decentralized computation offloading game [24] , Cloudlet based mobile cloud computing model (DECM) [25]. Yao et al. [26] presented an energy efficient task scheduling strategy (EETS) based on cost graph. However, there lacks a good decision mechanism for task migrations.

Considering the fact that the resources in a PCC can be considered by a number of time series data, therefore, the analysis of these data can be handled by the time series analysis techniques [27], where LSTM can be a good option to predict time series data. Based on OSGi-PC cloud architecture

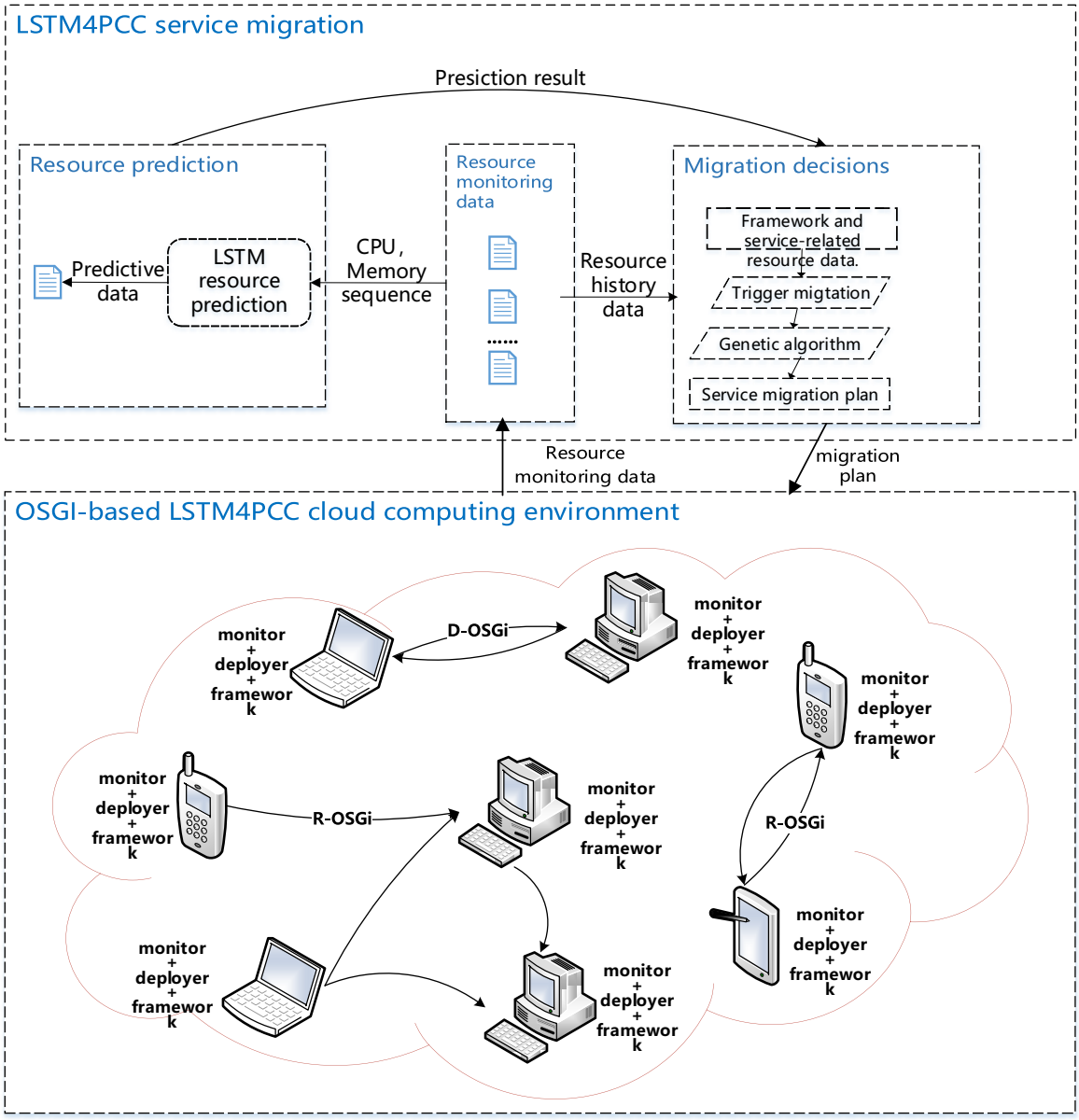

Figure 1: Overview of LSTM4PCC

[21], in this paper, we propose a deep learning based (i.e., LSTM) task migration method. LSTM4PCC, which is suitable to solve migration problems in pervasive cloud environment.

The structure of this paper is as follows: Section 2 presents LSTM4PCC cloud architecture. Section 3 presents LSTM-based migration algorithm. Section 4 presents the experiment and evaluates LSTM4PCC. Section 5 concludes the paper.

II. LSTM4PCC ARCHITECTURE FOR SERVICE MIGRATION

Preliminary work [21] has set up OSGi service migration, combining with D-OSGi and R-OSGi. In this paper, we add component lookup and migration functions, and resource monitor in the OSGi-PC infrastructure.

Figure 1 presents LSTM4PCC architecture for service migration.

There are two types of LSM4PCC nodes, the common computing node (OSGi-FW) and the central decision node (OSGi-CD), as shown in Fig.2. The common computing nodes can be PC nodes, or mobile devices (including Android phones, Android tablets, etc.), and there are three important functional components: framework, deployer, and monitor. The framework component is used to uniquely identify a OSGi-FW framework, which is responsible for the service deployer and monitor in working states; the deployer is responsible for the service migration in the OSGi-FW framework and

the control of operation status, where the central migration decision node determines transfer schemes, the corresponding deployer is selected to stop emigration and start immigration services; the monitor component is responsible for monitoring available resources. The central decision nodes include a context profiler (context analysis component), a resource predictor and a decision maker component. The decision node is based on OSGi, where the context profiler component periodically communicates with each computing node to get their resource information, and the resource predictor analyzes and predicts available resources. The decision node is deployed on the PC node for operating the whole system.

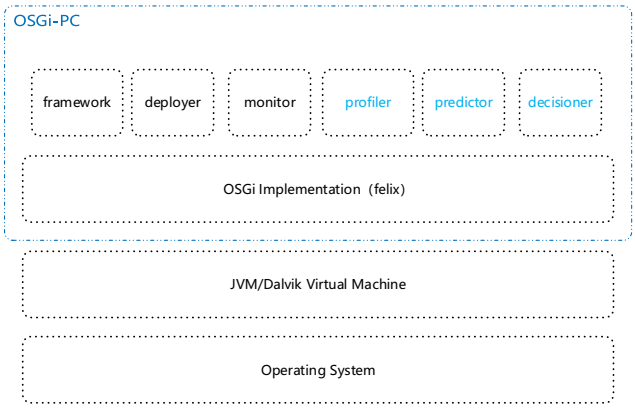

Figure 2: LSTM4PCC infrastructure

The perception of LSTM4PCC consists of three levels of contexts. The first is to percept system level contexts, mainly to obtain information on OSGi-FW frameworks, including how many OSGi-FW frameworks are running in the system, and the operation of each framework. The second is to percept the service status of each framework, including what services are running in each framework. These two level contexts can be obtained through D-OSGi and R-OSGi with the help of framework and deployer components. The third is to monitor resources, using common and performance-related memory and CPU indexes. The monitoring parameters include available memories and CPUs memories and CPUs within frameworks.

The framework memory fw_{memory} can be measured by the amount of available memory in JVM, and the CPU availability of the framework fw_{cpu} can be measured by the amount of available CPUs in JVM. In order to obtain the CPU utilization rate of bundle bd_{cpu}, the open source project JiP-OSGi[1] is modified in LSTM4PCC environment.

The perception of available resources will obtain a time series by periodically collecting data. In this case, LSTM (long and short term memory model) is potential for the prediction of time series data. LSTM4PCC uses a five-layer network to build a resources prediction model, including two LSTM layers, two Dropout layers, and a connecting layer, as shown in Fig.3.

[1]http://www.codeforge.com/article/364713

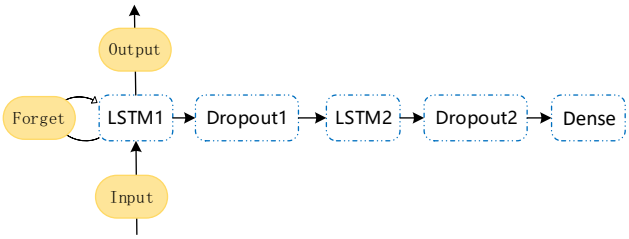

Figure 3: LSTM4PCC prediction model

Some test data are used to select different model parameters for training the LSTM model. The training of memory prediction model using different model parameters is shown in Table I.

III. LSTM BASED SERVICE MIGRATION

Fig.4 illustrates the process of making service migration decision and deployment as follows.

1) Resources on each node at different time have different weights (including forecast data). We calculate the weight of consumption of CPU and memory resources with $mean_{cpu} = \sum_{i=t}^{i=t+s} w_i cpu_i$ and $mean_{memory} = \sum_{i=t}^{i=t+s} w_i memory_i$. Based on $mean_{cpu}$ and $mean_{memory}$, the framework of nodes would be sorted from high to low by resource consumption. Then two lists are defined, which stand for resources in poor (listat) and resources in rich (listas), as shown in Fig.5. Then a formula is defined to measure the quality of resources:

$$D = \frac{mean}{std} \tag{1}$$

The $mean$ represents the weighted mean of available memory or CPU resources. std represents the corresponding standard deviation, so D can comprehensively measures the size and stability of the CPU or memory resources.

D is calculated for each node in listat and listas, and then the nodes are sorted in descending order D, and listbt and listbs are obtained after considering the variance parameters correspondingly, as shown in Figure 5 (listb). If a framework's $mean_{cpu}$ or $mean_{memory}$ exceeds the threshold $cpu_{threshold}$ or $memory_{threshold}$, this framework would be considered as the framework for migration, otherwise LSTM4PCC does not request a migration operation.

2) Next will select the appropriate target migration framework fw_{target} based on their resource situation $fw_{migrated}$. Choosing a target framework fw_{target} would be divided into three cases: a) If $fw_{migrated}$ belongs to the memory resource in poor, the destination framework fw_{target} is selected from the free memory list listbt which has the minimum D. b) If $fw_{migrated}$ belongs to the CPU resource in poor, the target framework would be selected from the optimal

Table I: Train and validation loss with different parameters of LSTM

Sequence	LSTM1	Droup1	LSTM2	Droup2	Dense Layer	Train Loss	Validation Loss
50	(1,50)	0.2	(50,70)	0.2	(70,1)	0.0214	0.0237
50	(1,50)	0.2	(50,100)	0.2	(100,1)	0.0213	0.0127
60	(1,60)	0.2	(60,80)	0.2	(80,1)	0.0228	0.0276
60	(1,60)	0.2	(60,100)	0.2	(100,1)	0.0225	0.0178

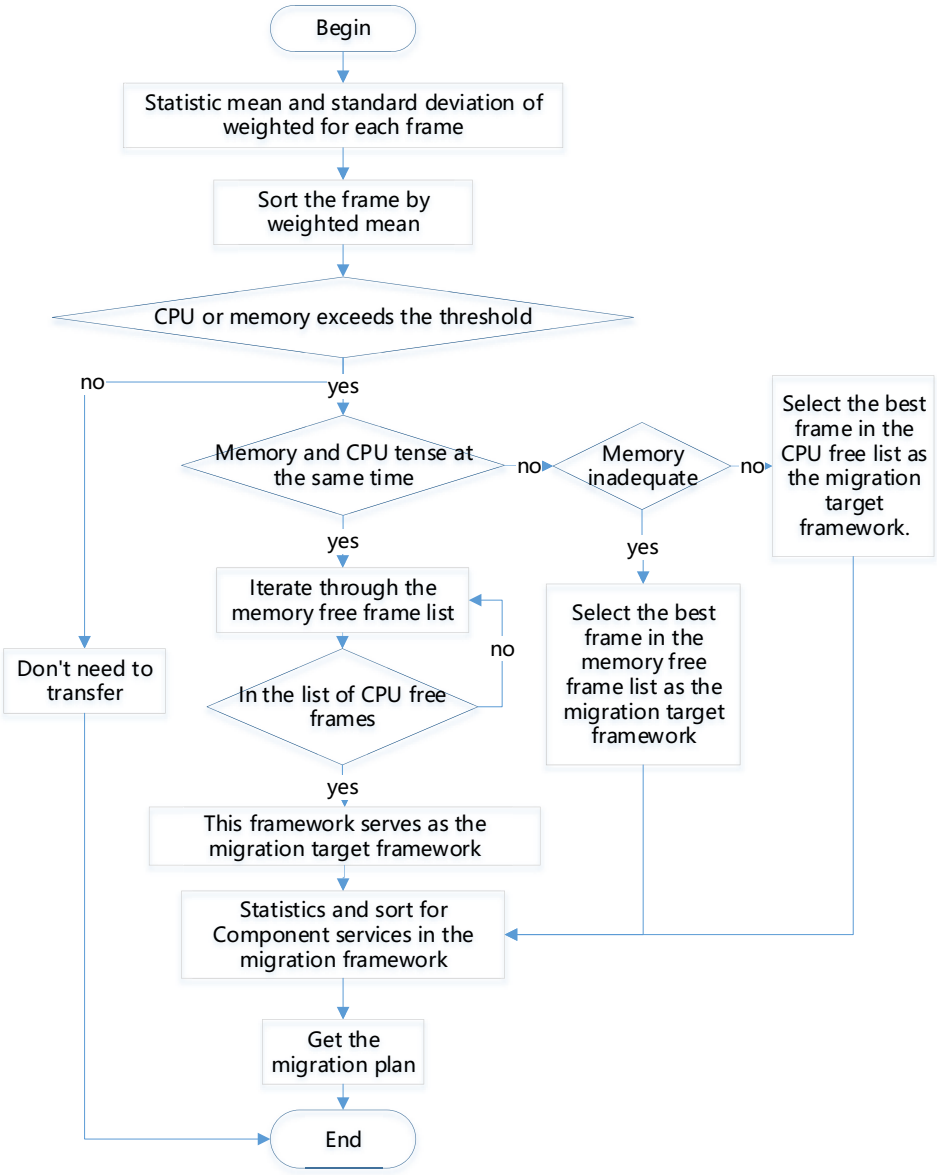

Figure 4: Service migration in LSTM4PCC

node in the CPU free listbs. c) If CPU and memory are both relatively tight, the priority will be given to the memory. If the framework is also located in the CPU free list, then select the framework as the target framework. For all the above three cases, if the targeting framework is not found, the migration can't be executed immediately.

3) After LSTM4PCC obtains $fw_{migrated}$ and fw_{target}, it needs to evaluate the resources consumption for system components. If the current state of $fw_{migrated}$ belongs to the CPU resource in poor, it will select the CPU service with the highest mean as the migration service

Table II: Configurations of evaluations

VM \ parameter	OS	CPU	memory	storage	IP	role
VM1	Ubuntu16.04	Single core 2.50GHZ	1GB	20GB	192.168.182.130	decision node
VM2	Ubuntu16.04	Single core 2.50GHZ	1GB	20GB	192.168.182.134	Cloud computing node
VM3	Ubuntu16.04	Single core 2.50GHZ	512MB	20GB	192.168.182.135	Cloud computing node
VM4	Ubuntu16.04	Single core 2.50GHZ	512MB	10GB	192.168.182.137	Cloud computing node
Android	Android4.4	Single core 2.50GHZ	512MB	5GB	192.168.182.133	mobile node

Table III: Frameworks resource info under the migration example

IP	CPU weighted mean	CPU mean /standard deviation	Memory weighted mean	Memory mean /standard deviation
192.168.182.137	99.24%	0.9924/0.1045	9.4630MB	9.4630/2.1626
192.168.182.135	98.77%	0.9877/0.0937	8.5428MB	8.5428/1.9791
192.168.182.134	95.21%	0.9521/0.1127	10.6752MB	10.6752/2.2371
192.168.182.133	-	-	0.1022MB	0.1022/0.0256

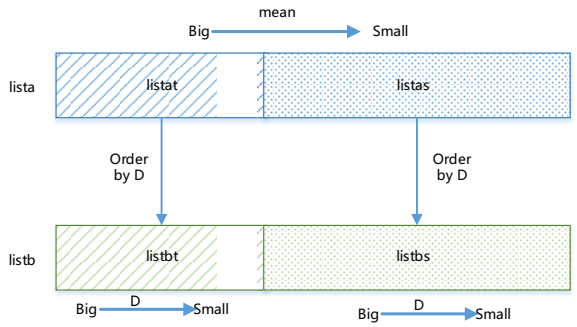

Figure 5: LSTM4PCC resource allocation

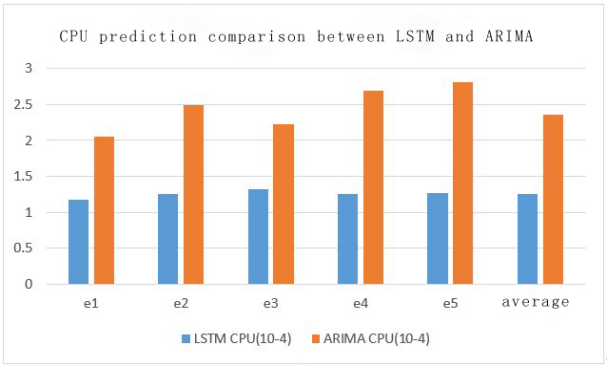

Figure 6: Compare LSTM based method with ARIMA

$bd_{migrated}$. If $fw_{migrated}$ belongs to the memory resource in poor, it will select the service with the highest memory consumption as the $bd_{migrated}$. If CPU and memory are both inadequate, it will repeat the step 2).

IV. Evaluations

The hardware platform used in the experiment is a ThinkPad E460 with four cores Intel Core i7 6500U, with four ubuntu16.04 virtual machines and an Android 4.4 virtual machine, which is configured as listed in Table II.

The experimental results are shown in Figure 6.

To measure the prediction accuracy of LSTM4PCC, the mean square deviation is defined as Formula 2.

$$MSE(t,s) = \frac{\sum_{i=t}^{i=t+s}(\hat{l_i} - l_i)^2}{s} \quad (2)$$

We compare the LSTM4PCC network model with the ARIMA autoregressive moving average algorithm. After autocorrelation and co-correlation analysis, we select p, d, q values of ARIMA as (2,1,3) to make these comparisons. We select Ubuntu node (192.168.182.134) as test data. Five different experiments are conducted as shown in Fig.6. Obviously, LSTM4PCC model performs much better than ARIMA.

To show the effectiveness of LSTM4PCC algorithm, the migration instances are illustrated in Table III, where the 192.168.182.133 node framework doesn't have CPU information because it is an Android node.

By analyzing the running state of each framework in a certain period of time, 0.95 and 9 are selected as the thresholds of memory and CPU resources. For the memory resource of a mobile node, the migration threshold is set to 0.10. For the migration instance in table III, LSTM4PCC algorithm selects 192.168.182.135 node (its weighted average of available memory is less than the threshold) for $fw_{migrated}$, Ubuntu node 192.168.182.134 as fw_{target}. With these threshold constraints, LSTM4PCC migrates the service with high consumption to the Ubuntu node with rich memory resource.

V. Conclusion

Service migration in pervasive cloud environment is an effective way for scaling resources on the fly to meet the increasing demand. Most of the existing works focus on a single migration algorithm, without applying LSTM into migration decision making. This paper proposes a LSTM based service migration method (i.e., LSTM4PCC) that is efficiently predicting the available memory and CPU resources. The

experimental results show that our LSTM4PCC migration algorithm performs over the ARIMA.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (No. 61309024), the Program on Innovation Method Fund of China (Grant No. 2015010300), the Key Research Program of Shandong Province (No. 2017GGX10140) and also supported by Fundamental Research Funds for the Central Universities.

REFERENCES

[1] J. Zhou, H. Zhu, M. Ylianttila, and M. Rautiainen, "Mobile multimedia cloud computing: An overview," in *Cloud Computing and Digital Media*. Chapman and Hall/CRC, 2014, pp. 22–41.

[2] W. Zhang, S. Tan, F. Xia, X. Chen, Z. Li, Q. Lu, and S. Yang, "A survey on decision making for task migration in mobile cloud environments," *Personal and Ubiquitous Computing*, vol. 20, no. 3, pp. 295–309, 2016.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, 2009.

[6] Z. Sanaei, S. Abolfazli, A. Gani, and M. Shiraz, "Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing," in *Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on*. IEEE, 2012, pp. 14–19.

[7] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using mapreduce," Carnegie-mellon univ Pittsburgh PA school of computer science, Tech. Rep., 2009.

[8] M. Black and W. Edgar, "Exploring mobile devices as grid resources: Using an x86 virtual machine to run boinc on an iphone," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*. IEEE, 2009, pp. 9–16.

[9] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, "Wishbone: Profile-based partitioning for sensornet applications." in *NSDI*, vol. 9, 2009, pp. 395–408.

[10] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*. IEEE, 2010, pp. 87–92.

[11] A. Gorbenko and V. Popov, "Task-resource scheduling problem," *International Journal of Automation and Computing*, vol. 9, no. 4, pp. 429–441, 2012.

[12] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[13] C. Wang and Z. Li, "Parametric analysis for adaptive computation offloading," in *ACM SIGPLAN Notices*, vol. 39, no. 6. ACM, 2004, pp. 119–130.

[14] S. Ou, K. Yang, and A. Liotta, "An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems," in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*. IEEE, 2006, pp. 10–pp.

[15] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, and C. C. Young, "Workload migration into clouds challenges, experiences, opportunities," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 164–171.

[16] N. Buchbinder, N. Jain, and I. Menache, "Online job-migration for reducing the electricity bill in the cloud," in *International Conference on Research in Networking*. Springer, 2011, pp. 172–185.

[17] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Computers and communications (ISCC), 2012 IEEE symposium on*. IEEE, 2012, pp. 000 059–000 066.

[18] R. K. Ma and C.-L. Wang, "Lightweight application-level task migration for mobile cloud computing," in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. IEEE, 2012, pp. 550–557.

[19] R. Niu, W. Song, and Y. Liu, "An energy-efficient multisite offloading algorithm for mobile devices," *International Journal of Distributed Sensor Networks*, vol. 9, no. 3, p. 518518, 2013.

[20] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1350–1354.

[21] W. Zhang, L. Chen, X. Liu, Q. Lu, P. Zhang, and S. Yang, "An osgi-based flexible and adaptive pervasive cloud infrastructure," *Science China Information Sciences*, vol. 57, no. 3, pp. 1–11, 2014.

[22] L. Gkatzikis and I. Koutsopoulos, "Mobiles on cloud nine: efficient task migration policies for cloud computing systems," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 204–210.

[23] H. Qian and D. Andresen, "An energy-saving task scheduler for mobile devices," in *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*. IEEE, 2015, pp. 423–430.

[24] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[25] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[26] D. Yao, C. Yu, H. Jin, and J. Zhou, "Energy efficient task scheduling in mobile cloud computing," in *IFIP International Conference on Network and Parallel Computing*. Springer, 2013, pp. 344–355.

[27] W. Zhang, W. Guo, X. Liu, Y. Liu, J. Zhou, B. Li, Q. Lu, and S. Yang, "Lstm-based analysis of industrial iot equipment," *IEEE Access*, vol. 6, pp. 23 551–23 560, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2825538