Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 8, Issue. 2, February 2019, pg.86 – 92

QoS-Aware Load Balancing Approach (QALBA) for Task Scheduling and Resource Selection using Enriched-Look ahead HEFT (E-LHEFT) Algorithm

K. Mohamed Sathik

Department of Computer Science, College of Arts and Sciences, Tanomah, King Khalid University, Abha, Saudi Arabia
mdsathik@gmail.com

Abstract— The present work discusses the service level agreement based optimization approach to satisfy both the mobile users and cloud service providers in the proposed MCC framework. The QoS-aware load balancing approach (QALBA) was implemented the task scheduling and resource selection using Enriched-Look ahead HEFT (E-LHEFT) algorithm by exploiting MAUI architecture and Pareto principle. The implementation results of the QALBA methodology with the baseline methods were presented.

Keywords— QALBA, MCC framework, E-LHEFT, MAUI architecture, Algorithm

I. INTRODUCTION

In MCC environment, the mobile application requires a high level of responsiveness, and it demands intensive computing resources in a mobile device to execute the sophisticated applications [1]. The remote execution and code offloading in MCC environment have created a significant impact on the capabilities of smartphones and availability of cloud remote servers. The major impacts on MCC framework related to local execution, computation offloading, and remote execution are load balancing, task scheduling, and resource selection. Load balancing is the process of balancing the load between schedulable tasks of mobile applications with selected cloud resources. Balancing the load between the tasks and resources are impacted from two broad main categories: task scheduling and resource allocation. Task scheduling in the mobile cloud is crucial for executing the requests of a similar application from multi-tenants. In resource allocation [2], the tasks are allocated to mobile core processors or available cloud resources. Task scheduling management increases resource consumption and reduces turnaround time during the task execution on a mobile device and the cloud. QoS is the mutual attempt of service performance that decides the user's degree of satisfaction with a particular service. To preserve the device energy and improve the user experience, the QoS-aware load balancing approach (QALBA) implements task scheduling and resource selection using Enriched-Look ahead HEFT (E-LHEFT) algorithm by exploiting MAUI architecture and Pareto principle. Furthermore, this chapter presents the implementation results of the QALBA methodology with the baseline methods.

II. AN OVERVIEW OF EXISTING LOAD BALANCING APPROACHES

TS-QoS [3] algorithm prioritizes the tasks depends on the special attributes of the tasks and achieves load balancing in cloud computing. It performs dynamic prioritization to adopt FCFS strategy using expected completion time of the task [4]. Green spot algorithm [5] preserves the energy-efficient battery

level of the mobile device for various cloud-based mobile applications. Genetic algorithm (GA) [6] based scheduling strategy achieves load balancing and reduces dynamic migration. HACAS [7] and PA-LBIMM [8] ensures in satisfying the users' demand with high QoS and load balancing. GA, TS-QoS, and HACAS algorithms initially perform scheduling followed by the load balancing but do not consider concurrent monitoring system. The conventional Look ahead variation of the Heterogeneous Earliest Finish Time (L-HEFT) algorithm [9] schedules the tasks of an application using precedence requirements of the tasks. Resource selection depends on only EFT (Earliest Finish Time) value of each task and corresponding children tasks in the task graph. Look ahead algorithm reduces the makespan about 20% compared to the HEFT algorithm due to the consideration of the currently selected task and also upcoming tasks. The L-HEFT algorithm does not work on the feasibility of a higher look ahead depth of the mobile applications. Even though the existing task scheduling algorithms provide QoS and achieve load balancing in a stringent mobile cloud environment, many priorities based scheduling techniques affect the low priority of users. The scheduling strategy suffers by the uncertainty when there is the waiting time of the task is high in the queue. Also, the existing migration based load balancing after scheduling the task increases the processing delay.

Pareto principle

Pareto principle based task-group-mapping of cloud resources maintains the load balancing based on the length of the task group and server utilization or current load on the server. Each application has the sequence of tasks that executes on cloud and mobile device. For instance, each offloaded move to the next position in a chess game is the task to the cloud. Several mobile users offload the tasks at the same time in which similar EFT of tasks is considered as a group for execution. In proposed QALBA approach, task grouping, and Pareto principle process simultaneously to select the PM resource for the task execution. The proposed approach divides the length of the task group as high and low. Similarly, utilization of PM is categorized into high and low utilization. Pareto principle depends on the 80/20 rule. The proposed algorithm states that 80% of the length of the task group schedules on 20% of PM utilization or load, and 20% of length of task group schedules on 80% of PM utilization or load. Hence, it maintains the load balance of all active servers while executing the application in a cloud environment.

E-LHEFT task scheduling algorithm for MCC

Input: List of input tasks
Output: Scheduled tasks with QoS and load balancing
Step 1: Initially assign task groups m=0
Step 2: Declare tasks of an application i= 0 to total number of tasks (T_{L}_size)
Step 3: Declare cloud and mobile resources j= 0 to total number of available cloud resources (R_{L}_size)
Step 4: Compare processing requirements of each task (T_{LENi}) with threshold value (α)
If ($T_{LENi} > \alpha$) then tasks of an application as cloud tasks (Ti^{C}) or ($T_{LENi} < \alpha$) then tasks of an application as mobile tasks (T_i^{M})
// E-LHEFT task scheduling algorithm
Step 5: Rank tasks of an application using upward rank value of each cloud task $R_U(T_i)$ from DAG
For unscheduled tasks, find out highest $R_U(T_i)$ from $\{R_U(T_i)\}$ and put into selection priority variable
//Task grouping and Pareto principle based PM selection
Step 6: Assign children of selected cloud task to task list of an application $\{T_L\}$
Step 7: Consider cloud tasks from $\{T_L\}$
Step 8: Check total lower length of task group with total higher length of task group $\{Total\_GT_{LEN}(L) \leqslant Total\_GT_{LEN}(H)\}$
Groups current task and similar execution time of children tasks as GT_{Lm}
Splits GT_{Lm} into lower length task group $GT_{Lm}(L)$ and higher length task group $GT_{Lm}(H)$
Step 9: Consider all tasks to schedule and all resources
Step 10: Find expected load (utilization) of PM
Assign $GT_{Lm}(L) \rightarrow \min(PM_j^{U}) \forall$ active PMs
Assign $GT_{Lm}(H) \rightarrow \max(PM_j^{U}) \forall$ active PMs
// VM selection
Step 11: Calculate expected completion time of each task T_i (E_{CT})
Step 12: Consider length of task group GT_{LEN}, file size and VM specification VM_{jS}
Step 13: Check ($GT_{LEN}=VM_j^{S}$ && $VM_j(T_i(E_{CT})) \leqslant VM_{jS}$) then
Schedule tasks in task group list GT_{Lm} into Virtual Machine resources VM_j

III.EXPERIMENTAL EVALUATION

To demonstrate the importance of the proposed approach, numbers of experiments were carried out in a Cloud simulator environment. The proposed approach compares with existing algorithms such as GA, TS-QoS, Min-Min, and PA-LBIMM in a cloud environment. Furthermore, the proposed algorithm is compared with existing Green spot, EAPA and HACAS in a mobile cloud environment.

A. *Experimental setup*

MCC environment is highly dynamic in nature with a surplus amount of mobile user requesting the similar application at a time. In this proposed work, Cloud Sim tool simulates the task scheduling based grouping method and load balancing model of cloud resources. The simulation environment simulates mobile cloud framework by mobile user, mobile application model, and remote cloud server to run large scale mobile applications. Cloud executes the resource-intensive and compute-intensive tasks of an application on the remote server, and mobile executes the part of the task of an application based on mobile resource availability. The location of task processing varies according to the current task processing in mobile and cloud. The Cloud Information Service (CIS) provides the information about available cloud resources and resource name, ID, the total number of machines, total processing elements, and processing capability of each machine. Table 1 shows the simulation parameters and simulation values were used in experimental evaluation. Experimental setup considers 30 Chess OCR applications with various specifications and each application partitioned into the number of tasks with different length. Makespan of application ranges from 0 second to 1 second. The processing capabilities of PM resources vary between 1800, 2000, 4000, 6000 MIPS.

TABLE I
SIMULATION SETUP

Simulation parameters	**Simulation value**
Number of tasks	10-150
Input file size	10-1000MB
Average task length	250-12500KB
Number of requests	500-2500
Number of mobile cores	3-5
Number of physical machines	10-50
Number of virtual machine instances	100-1000
Number of mobile applications	30
Execution time	0-35 minutes
Percentage of load change	10-50%
CPU Speed	30GHz-300GHz
PM MIPS	1500-6000MIPS
VM MIPS	100-1500MIPS
VM memory	4GB-32GB
VM bandwidth	200-1200Kbps
VM processing element	2

B. *Evaluation metrics Makespan*

It is the total time taken to complete a mobile application execution in the mobile, and the cloud server, representing the time difference between the finish time of the exit task and start time of an entry task.

Energy level: It refers that the battery (energy) level of the mobile device while running a specific mobile application in the mobile cloud environment. Hence, the energy consumption of the mobile device is based on application completion time and energy cost.

Load balancing: It denotes the percentage of load balancing process required by the system that is the percentage of load deviation among the allocated cloud resources.

Average Resource utilization: It is the average value of utilizing each cloud resource among the allocated resources while executing a specific application.

C. *Experimental results and analysis*

The experimental results demonstrate the performance variation of the QALBA approach while experimenting with the conventional task scheduling, resource allocation, and load balancing techniques. The performance variation can be revealed using four unique metrics such as Number of tasks Vs Makespan, Time Vs Energy level, Number of tasks Vs Load balancing, and Number of tasks Vs Average resource utilization.

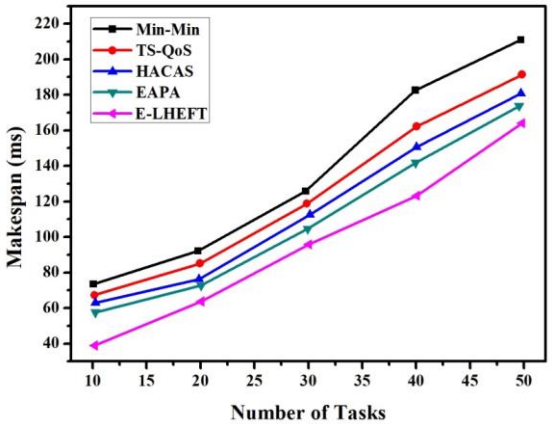

Fig. 1 Number of tasks Vs Makespan

Fig. 1. shows the makespan value while varying the number of tasks and the corresponding numeric values are shown in the Table 2. Comparison with the existing algorithms proves that the proposed QALBA approach minimizes the makespan value significantly using grouping method and Pareto principle. The proposed E-LHEFT algorithm groups the task request for execution. Each task group contains a different number of tasks 75 based on the task file size. Pareto principle based group of task processing enhances the performance of mobile devices in terms of energy and application completion time. Therefore, the proposed E-LHEFT algorithm minimizes the makespan value by 5.9% when increasing the number of tasks from 10 to 50 but, in the same case, the existing scheduling algorithms increases the makespan value by 6.4% to complete the same mobile application execution in the mobile cloud environment.

TABLE II
NUMBER OF TASKS VS MAKESPAN

Number of Tasks	Makespan (ms)				
	E-LHEFT	Min-Min	TS-QoS	HACAS	EAPA
10	40	74	67	63	57
20	65	92	85	77	72
30	96	126	119	112	104
40	123	182	162	151	141
50	165	210	191	182	173

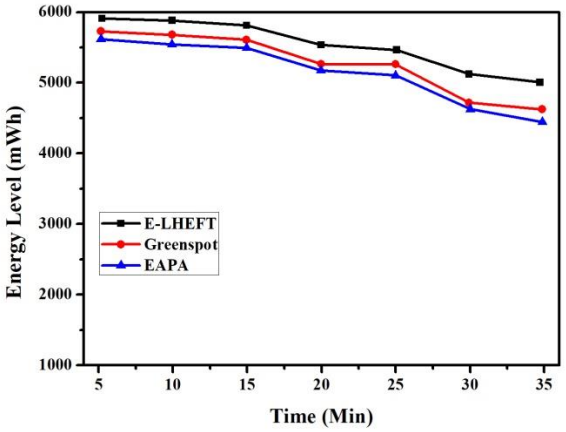

Fig. 2 Time Vs Energy level

Fig. 2. shows the reduction of mobile device battery level based on time variants while executing the tasks in mobile cores and remote cloud server. Table 3 represents the implementation results in numeric values of Fig. 2. The proposed E-LHEFT algorithm minimizes the energy consumption of the mobile device while executing the offloaded tasks in the cloud by grouping the tasks. Initially, MAUI architecture saves the mobile device battery level while scheduling the tasks. This scheme focuses on the Pareto principle based

task grouping which shortens the overall processing time of an application. Therefore, the proposed E-LHEFT algorithm consumes only 14.4% of battery energy from the initial energy level when executing the mobile application with the duration of 35 minutes but, the EAPA and green spot consumes 19.82% and 18.03% respectively.

TABLE III
TIME VS ENERGY LEVEL

Time (Min)	Energy level(mWh)		
	E-LHEFT	Greenspot	EAPA
5	5912	5773	5680
10	5875	5720	5595
15	5870	5678	5520
20	5561	5356	5208
25	5477	5305	5153
30	5170	4778	4678
35	5042	4663	4515

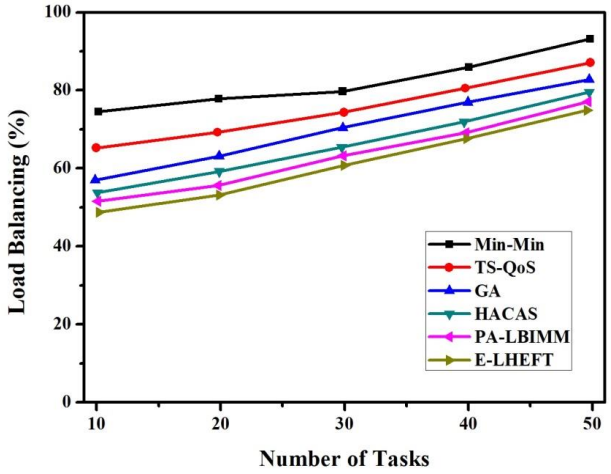

Fig. 3 Number of tasks Vs Load balancing

The number of mobile users and mobile user requests are not stable in the dynamic mobile cloud environment; hence, the load of each PM varies according to the number of mobile users' requests. The load deviation values of all resources are shown in Fig. 3 and corresponding numeric values are shown in Table 4. From the experimental graph, it is observed that the load deviation of E-LHEFT algorithm decreases compared to existing algorithms. The E-LHEFT algorithm balances the load of PM resources using concurrent processes of the task scheduling and load balancing in a mobile cloud environment. Each physical host handles the load balancing based on the consideration of availability and utilization of the host. Therefore, Pareto principle based task-group-mapping of cloud resources minimizes the load deviation value in the proposed approach. In resulting, the proposed E-LHEFT algorithm maintains the load deviation within 75%, but the existing methods have the load deviation values between 77% to 92%, when the number of tasks is 50.

TABLE IV
NUMBER OF TASKS VS LOAD BALANCING

Number of Tasks	Load balancing (%)					
	E-LHEFT	TS-QoS	GA	HACAS	PA-LBIMM	Min-Min
10	48	65	57	53	51	74
20	53	69	63	59	55	78
30	61	74	70	65	63	79
40	67	80	77	71	69	85
50	74	87	82	79	76	92

Accordingly, the E-LHEFT algorithm upholds the average resource utilization from 81% to 98%, but the existing algorithms maintain the average resource utilization from 63% to 88% alone when increasing the number of tasks in the mobile cloud environment.

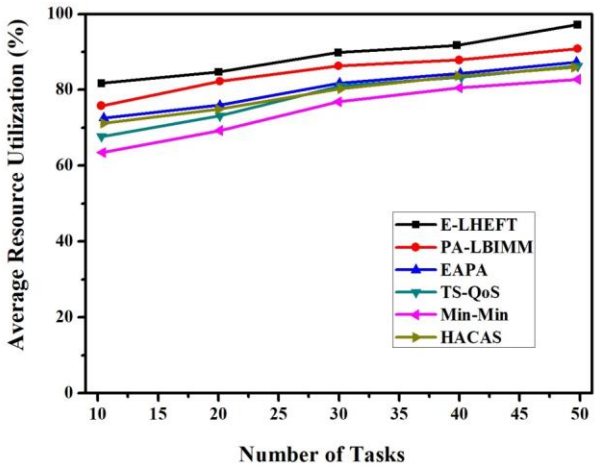

Fig. 4 Number of Tasks Vs Average Resource Utilization

Fig. 4 shows the average resource utilization that varies with the number of tasks for scheduling on cloud and mobile resources. Table 5 illustrates the numeric values of Fig. 4. The proposed E-LHEFT algorithm increases the average resource utilization based on the task grouping and resource capabilities matching method using Pareto principle. The proposed resource utilization depends on the task group size and started VMs on a particular server. The technique processes more number of applications by satisfying load balancing. Also, the E-LHEFT algorithm optimally utilizes the allocated resources rather than initiating a new PM and VM resources, which intends to reduce the energy consumption in cloud server.

TABLE V
NUMBER OF TASKS VS AVERAGE RESOURCE UTILIZATION

Number of tasks	Average Resource Utilization (%)					
	E-LHEFT	TS-QoS	EAPA	HACAS	PA-LBIMM	Min-Min
10	81	67	72	71	76	63
20	85	73	76	74	82	69
30	90	81	82	82	86	77
40	91	84	85	85	89	81
50	98	87	87	87	91	83

IV. CONCLUSIONS

In summary, the design and development of MCC framework by enriching load balancing approach with task scheduling, and resource selection modules are presented. The proposed Enriched-Look ahead Heterogeneous Earliest Finish Time (E-LHEFT) algorithm for MCC framework using task prioritization, task selection, task grouping, Pareto principle, and processor selection are explained clearly. The Mobile Assistance Using Infrastructure (MAUI) mobile application model is utilized in QALBA approach for computation and data storage execution in outside of the mobile device. The significant performance improvement is attained by focusing on the QoS-aware mobile application execution on a remote server in mobile cloud environment. Eventually, it illustrates the evaluation results of the QALBA along with conventional mobile cloud task scheduling, resource allocation, and load balancing approaches when experimenting using the mobile gaming applications. In the proposed approach, the mobile device energy consumption is calculated using the energy profile model in MCC framework. The main contributions of this research work are computation offloading for multiple mobile user requests, task prioritization using an upward rank value for both current task and successor tasks, higher and lower length task groups, load balancing using Pareto principle, and processor selection in local (mobile cores) and remote execution (cloud server) in QoS-aware load balancing approach. The QALBA approach increases the provisioning of SaaS and PaaS services in the dynamic mobile cloud environment. The proposed E-LHEFT algorithm satisfies the QoS requirements in terms of minimizing the overall application completion time by 5.7%, significantly maintains the energy level of the mobile device by 14% to 19% on average, improves scalability and maintains the load deviation among physical and virtual machine resources within 75%, and maximizes the average resource utilization of cloud resources by 90% by processing more number of mobile client application requests..

ACKNOWLEDGEMENT

The author thank College of Arts and Sciences, Tanomah, King Khalid University, Saudi Arabia

REFERENCES

[1] R Hasan, MM Hossain, and R Khan, "Aura: an IoT based cloud infrastructure for localized mobile computation outsourcing", 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp.183-188, 2015.

[2] S Nastic, M Vögler, C Inzinger, HL Truong, and S Dustdar, "rtGovOps: a runtime framework for governance in large-scale software-defined IoT cloud systems", 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp.24-33, 2015.

[3] Xiaonian Wu,Mengqing Deng, Runlian Zhang, Bing Zeng, and Shengyuan Zhou, "Task scheduling algorithm based on QoS-driven in cloud computing", Elsevier transaction on Information Technology and Quantitative Management, Vol.17, pp.1162-1169, 2013.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing", IEEE transaction on Pervasive Computing, Vol.8, No.4, pp.14–23, 2009.

[5] Namboodiri, Vinod, and Toolika Ghose, "To cloud or not to cloud: A mobile device perspective on energy consumption of applications", IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks, pp.1-9, 2012.

[6] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment", Journal of Computers, Vol.7, No.1, pp.42–52, 2012.

[7] Xianglin Wei, Jianhua Fan, Ziyi Lu, and Ke Ding, "Application scheduling in mobile cloud computing with load balancing", Hindawi Publishing Corporation Journal of Applied Mathematics, pp.1-13, 2013.

[8] Huankai Chen, Frank Wang, Na Helian, and Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing", IEEE transaction on PARCOMPTECH, pp.1-8, 2013.

[9] Bittencourt, Luiz F., Rizos Sakellariou, and Edmundo RM Madeira, "DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm", IEEE 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp.27-34, 2010.

