Pers Ubiquit Comput (2016) 20:295–309
DOI 10.1007/s00779-016-0915-y

ORIGINAL ARTICLE

A survey on decision making for task migration in mobile cloud environments

Weishan Zhang[1] · Shouchao Tan[1] · Feng Xia[2,3] · Xiufeng Chen[4] · Zhongwei Li[1] · Qinghua Lu[1] · Su Yang[5]

Received: 16 October 2015 / Accepted: 30 March 2016 / Published online: 21 April 2016

Abstract The key idea of MCC is using powerful back-end computing nodes to enhance capabilities of small mobile devices and provide better user experiences. An effective and widely used approach to realize this is task migrations. Decision making is an important aspect of migrations which affects the feasibility and effectiveness of task migrations. There have been a number of research efforts to MCC which help make decisions for task migrations. In this paper, we present a comprehensive survey on decision making for task migrations in MCC, including decision factors and algorithms. We observe that there are still some challenges such as comprehensive context awareness, unified migration standards, large-scale experiments, more involvement of latest achievements from artificial intelligence, and flexible decision-making mechanisms. The paper highlights these issues and challenges to attract more efforts to work on MCC.

Keywords Cloud computing · Task migration · Decision making · Mobile cloud · Context awareness

✉ Weishan Zhang
zhangws@upc.edu.cn

Shouchao Tan
tansc.upc@foxmail.com

Feng Xia
f.xia@ieee.org

Xiufeng Chen
chenxiufeng@hisense.com

Zhongwei Li
lizhongwei@upc.edu.cn

Qinghua Lu
dr.qinghua.lu@gmail.com

Su Yang
suyang@fudan.edu.cn

[1] School of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China

[2] School of Software, Dalian University of Technology, Dalian 116620, China

[3] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116620, China

[4] Hisense TransTech, Ltd., Qingdao, China

[5] College of Computer Science and Technology, Fudan University, Shanghai 200433, China

1 Introduction

Smart mobiles have been becoming popular because of their increasing processing capabilities, convenience, and portability. It is predictable that mobile devices will replace the traditional laptop and desktop computers in the near future [1]. However, mobile systems have inherent limitations, including short battery life and limited computing and storage capabilities [2]. Enhancing capabilities of mobile devices can be achieved by a new computing paradigm called mobile cloud computing (MCC) [3].

MCC enhances and optimizes mobile devices' computing capability and it can alleviate storage and mobility problems of mobile devices. It brings new types of services and facilities for mobile users to take full advantages of cloud computing. A number of MCC solutions have been proposed, such as Misco [4], Clonecloud [5], and Hyrax [6]. March [7] developed a component based mobile application framework leveraging cloud resources. Lu [8] proposed a conceptual architecture to perform screen rendering tasks inside the cloud. SOPHRA [9] uses a cloud centric middleware to enable mobiles to host web services in the mHealth domain. In [1], the author

presented a new mobile cloud architecture, leveraging the collaboration between mobile devices and conventional desktop or laptop computers to empower mobile computing, particularly aimed at optimizing data distribution and resources' utilizations so that expected Quality-of-Service (QoS) requirements can be met. They also proposed an algorithm to select an optimal resource allocation strategy to satisfy various service level agreements (SLA). MCC has been used in various applications including mobile commerce, mobile learning, and mobile gaming [3].

Enhancing capabilities of small devices in MCC can be achieved by migrating applications, and offloading computing intensive or data intensive tasks to proximate or remote cloud nodes [10, 11], which is called task migration in MCC. In some occasions, the most effective application migrations should be able to simultaneously meet multiple user requirements [12]. A computation migration is also confronted with several key challenges such as context awareness and remote execution control [13]. Considering all of the problems, we need a good migration scheme about which part to migrate, when and where to migrate. So decision making for task migration is an important issue which can affect application executions including security, performance, power consumption of mobile devices [3] and of whole systems, and hence affect experiences of mobile users.

In this paper, we present a relatively comprehensive survey on the decision making of task migrations. The motivation is to discuss some important issues in task migration processes including what kind of decision factors should be considered and what kind of decision mechanisms are used. Besides, we present some challenges in order to point out future research directions.

The rest of the paper is structured as follows. Section 2 gives an overview of mobile cloud computing and introduces the task migration in MCC including generally steps and architectures for task migrations. A survey of related work and decision-making approaches for task migration is given in Sect. 3. In Sect. 4, A comprehensive comparison and discussions to migration decision approaches are presented. Some challenges of the task migration in MCC are listed in Sect. 5. Section 6 concludes the paper.

2 Overview of task migration in MCC

The MCC forum defines MCC as follows:

> 'Mobile cloud computing at its simplest refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and MC to not just smart phone users but a much broader range of mobile subscribers' [14].

There are some other definitions, such as Aepona [15] and Liu [16]. We can describe the general idea of mobile cloud computing as the following architecture shown in Fig. 1. Potentially, MCC has a number of advantages such as extending battery lifetime, improving data storage capacity, enhancing performance, and dynamic provisioning [3].

2.1 Reasons and steps to do task migrations

There are a number of reasons to conduct task migrations. First, mobile applications are getting more complicated and more computing intensive [17]. Second, mobile users would like to obtain better performance when using mobile devices which have inherent limitations with respect to a built-in memory, battery, storage capability, and furthermore, poor bandwidth [2]. In a word, the reason for doing task migrations in mobile cloud environment is that there are mismatches between requirements and the mobile devices' capabilities.

In order to make migrations meaningful, some factors need to be considered. We classify these factors into four categories: applications' characteristics, mobile devices' characteristics, environment factors, and user's preferences

Fig. 1 General idea of mobile cloud computing

and requirements. These factors can help to decide whether a migration should be conducted and how to proceed task migrations.

Generally, the steps of task migrations are as follows:

1. Looking for available resources and services which mobile devices can offload computing tasks to [18, 19].
2. Monitoring environment context information such as status of surrounding networks and relevant devices, predicting what are needed through some sort of reasoning approaches [20–23].
3. Conducting application partitions at a suitable granularity [24] and making decision for tasks scheduling with appropriate algorithms [25, 26].
4. Remote execution control for applications needs to be conducted. This includes how to interact with mobile clients, such as getting users' input and returning results to mobile ends.

The first and second steps make preparation for task migration which are used to collect useful information. The third step is the core for task migrations, namely decision making of task migrations. Effective decisions are important. The last one makes sure that applications can be correctly executed even after the migrations.

2.2 Task migration architectures of MCC

A task migration architecture means what type of resources mobile devices use to enhance their abilities and the way in which mobile devices are connected to and interact with cloud resources. According to the resources organization, existing task migration architectures can be classified as the following four types:

1. *Public cloud*: Mobile devices offload tasks to public resources comprised of stationary servers located far from the mobile nodes or proximate centralized desktops and laptops accessible via the Internet. The migrated tasks or portions of applications can be executed remotely in the cloud. Some task migration solutions use this architecture to enhance capabilities of mobile devices and get better execution of applications. In MAUI [27], mobile devices offload code to a infrastructure composed of immobile MAUI servers. Smart phones use remote procedure call to communicate with MAUI servers. Clonecloud [5] also uses a computational public cloud architecture to transfer mobile applications. It transforms a single machine execution (e.g., smart phone computation) into a distributed execution (e.g., smart phone plus cloud computation) in which the resource intensive parts of the execution are run in powerful clones in the cloud. Public cloud is highly available, scalable, and resource elastic with powerful computing capabilities while mobile phones are resource constrained. A novel MCC architecture was proposed in [1]. It leverages the collaboration between mobile devices and conventional desktop or laptop computers to empower mobile computing to optimize data distribution and utilization of CC resources. As said in [17], the disparity in capabilities between public cloud and smart phones is high and persistent. Therefore, leveraging public cloud to augment mobile devices can benefit mobile application executions pretty much. But the communication time and cost must be considered for migrations because accesses to the cloud may incur a long latency due to wide area network (WAN) delays.
2. *Cloudlet*: A Cloudlet [28] is one computer or cluster of computers that is well connected to the Internet and available to nearby mobile devices. It is resource rich and proximate. Due to the proximity, the end to end response time of applications executing within it is little (a few milliseconds) and predictable. Cloudlet simplifies the challenge of meeting peak bandwidth demands and gives users better experience especially for realtime mobile applications. But using Cloudlets also faces to some practical issues such as reliability problems. Cloud-Vision [29] proposed a mobile-Cloudlet-cloud architecture named MOCHA to do realtime face recognition. The Cloudlet determines how to partition computation among itself and multiple servers in the cloud to optimize the overall Quality of Service (QoS). The implementation and result demonstrate that Cloudlet is technically feasible and high powered to provide benefits to mobile device face recognition applications considering its higher computational power with minimal latencies .
3. *Cloud of Mobile Network Operator (MNO)*: A mobile network operator or MNO is a provider of wireless communications that owns or controls all of the elements necessary to sell and deliver services to an end user including radio spectrum allocation, wireless network infrastructures, and provisioning of computer systems and marketing [30]. There are many MNOs like China Mobile and AT&T which has provided cloud services [31]. As said in [32], a MNO can be a cloud provider by deploying cloud concepts including SaaS, IaaS. A MNO can be a cloud broker which plays as a service mediator between cloud providers and mobile users, and from the other aspect, a broker across other MNOs. Cloud of MNOs is widely covered, safe, and reliable. Mirror [33] keeps a mirror for each smart phone on a computing infrastructure within 3G networks. Paper [32] proposed a service based arbitrated multi-tier infrastructure for

mobile cloud computing. MNOs are used as arbitrators between front end (mobile users) and back end (cloud service providers) to do resource and service allocation for more efficient executions. A MNO and its authorized dealers compose the infrastructure together with clouds.

4. *Cloud of mobile devices*: In the former three architectures, mobile devices act as thin clients which utilize external resources in cloud computing platforms to process computing intensive or data intensive tasks. But we cannot avoid situations that an access to these platforms may be not available and/or it is too expensive to access them. Cloud of mobile devices is a solution to these situations. Mobile devices play the role of resource provider, those which are in vicinity of users and in a stable mode form a virtual cloud for task offloadings. In Hyrax [6], the author introduced a concept of using mobile devices as resource providers. Similar work has been done in grid computing. Black and Edgar [34] demonstrated the feasibility of using mobile devices as part of a grid. With cloud of mobile devices, users can share resources and execute jobs among the devices to gain better experience. This is very suitable for some scenarios like group tourism. [35] created a cloud among devices in vicinity taking advantage of the pervasiveness of mobile devices.

Table 1 shows a simple comparison of the four architectures. Each architecture has its specific characteristics and can meet different task migration requirements. When doing task migrations, we can choose a suitable mobile cloud architecture according to migration scenario and objectives.

Apparently MCC has been a hot topic to augment mobile devices capabilities. There have been many research efforts on mobile cloud environment, as discussed in [3, 24] and many solutions for task migrations in mobile cloud computing as represented in the following Sect. 3.

3 Decision making for task migration

Section 2 delivers an overview for task migration in MCC including the reasons to do task migrations and architectures existing in MCC. To ensure efficient program executions, a task migration needs an intelligent decision making way to adapt to dynamic changes in mobile cloud computing environments and achieve better execution performance. In this section, we focus on decision making for task migration in MCC and some specific existing approaches of task migrations will be talked about with some detailed descriptions.

There have been many solutions about decision making for task migration in mobile cloud computing. They are all for augmenting mobile devices and making applications better executed, but they are different from each other in decision objectives and decision algorithms. A significant difference is that a migration decision can be static or dynamic. Now in the following is some approaches with the two different decision making ways. We discuss each approach with a detailed description.

3.1 Static task migration decision approaches

1. *Wishbone*: Wishbone [36] is an application partition system based on a data flow graph of operators. Its partitioning algorithm models the application partition problem as an integer linear program (ILP). It uses an ILP solver to solve the optimal graph partitioning problem to get a partition scheme and then assigns tasks to a central server or a sensor node. In this way it

Table 1 Task migration architectures of MCC

Arch	Chara				
	Composition	Distance with user	Advantages	Disadvantages	Task migration examples
Public cloud	Centralized desktops and laptops, stationary servers	Distant/ proximate	Highly available, scalable, elastic, and powerful	Wide area network (WAN) latency	MAUI [27], Clonecloud [5]
Cloudlet	One computer or cluster of computers	Proximate	Resource rich and proximate	Low reliability	Cloudlet [28], Cloud-Vision [29]
Cloud of MNO	Mobile network operator	Proximate	Widely covered, safe, and reliable	May be expensive	Sami [32], Mirror [33]
Mobile cloud	Mobile devices	Proximate	Convenient	May be not stable	Hyrax [6, 35]

The abbreviated columns represent the following: arch architecture; chara characteristics

can reduce mobile devices' cost as much as possible, which is a linear combination of network communication overhead and CPU load. In Wishbone, the partition approach is only suitable to those applications which can be expressed as a data flow graph.

2. *VM placement/migration*: Network I/O performance would affect application performance significantly. Therefore, [37] proposes a network aware virtual machine placement and migration approach in cloud computing environment. It aims to minimize data transfer time of an application. In the proposed VM migration policy shown in Fig. 2, a VM migration is triggered when the data transfer time crosses a certain threshold due to unstable networks. Then, the next optimized physical machine which has minimal access time is chosen according to the current network condition, and the VM is migrated to this physical machine for better performance. Experiment result shows that average task completion time is decreased with task migrations. A precondition of the proposed approach is that host storage and network bandwidth must be known in advance.

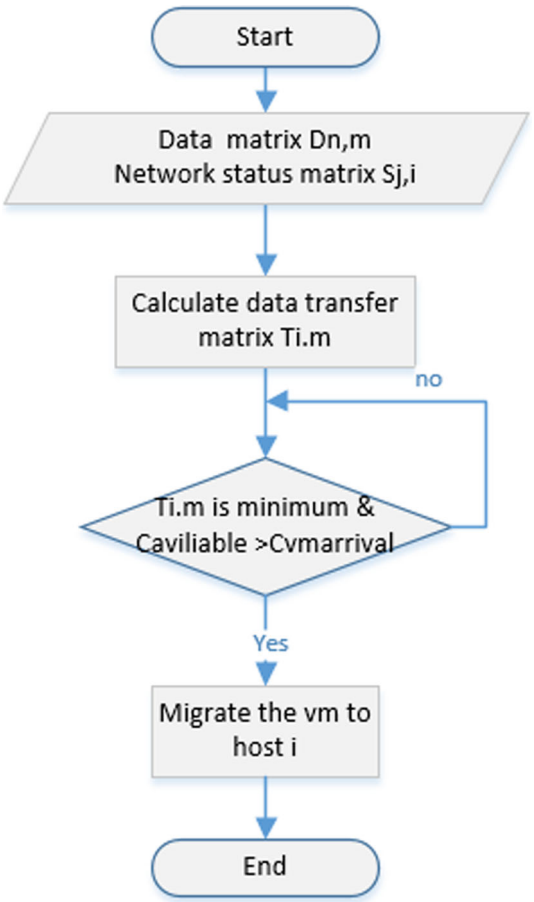

Fig. 2 Network aware VM migration approach

3. *Clonecloud*: Clonecloud [5] is a system dynamically migrating mobile applications to cloud. It uses remote cloud resources and partitions applications at a thread level to obtain a fine granularity and flexibility. A mathematical optimizer chooses migration points that optimize the objective (such as total execution time or mobile device's energy consumption) and makes dynamic profiling to build a cost model. The runtime system chooses which partition to use. After the partition and migration, the application is under distributed execution (parts of the execution are seamlessly offloaded to computational cloud, while rest are still executed locally), and the result will be back to the mobile device to be integrated.
4. *Task-resource Scheduling*: In [38], the author gave a solution to task-resource scheduling problem in cloud computing system. It describes an application work flow as a directed acyclic graph comprised with tasks and their data dependency relationships. The task-resources scheduling problem is to find a scheduling scheme to minimize total execution cost of applications running on the resources provided by cloud service providers. Total execution cost is the sum of computation time and file transmission time. This problem is reduced to a satisfiability problem and then solved with a SAT solver. The overall process is as shown in Fig. 3.
5. *Task Scheduling with DVFS*: A novel algorithm for the MCC task scheduling problem is presented in [39]. The algorithm is designed to minimize total energy consumption of an application in a mobile device under a hard constraint on the application completion time. An application is represented by a directed acyclic task graph $G = (V; E)$. Each node $v_i \in V$ represents a task, and a directed edge $e(v_i; v_j) \in E$ represents the precedence constraint. And the task-precedence requirements are considered and customized. There are three steps when doing task scheduling. In the first step, a minimal delay task scheduling is generated without considering energy consumption of the mobile device. The initial scheduling algorithm takes into account a joint scheduling of tasks on local cores, wireless communication channels, and cloud nodes. And then the task migration algorithm performs energy reduction in the second step by migrating tasks to the cloud or other local cores that can bring great energy reduction without violating the application completion time constraint. In the third step, dynamic voltage and frequency scaling (DVFS) technique is used to further reduce energy consumption by making a suitable choice of the operating frequency for each local core. To avoid high time complexity, the

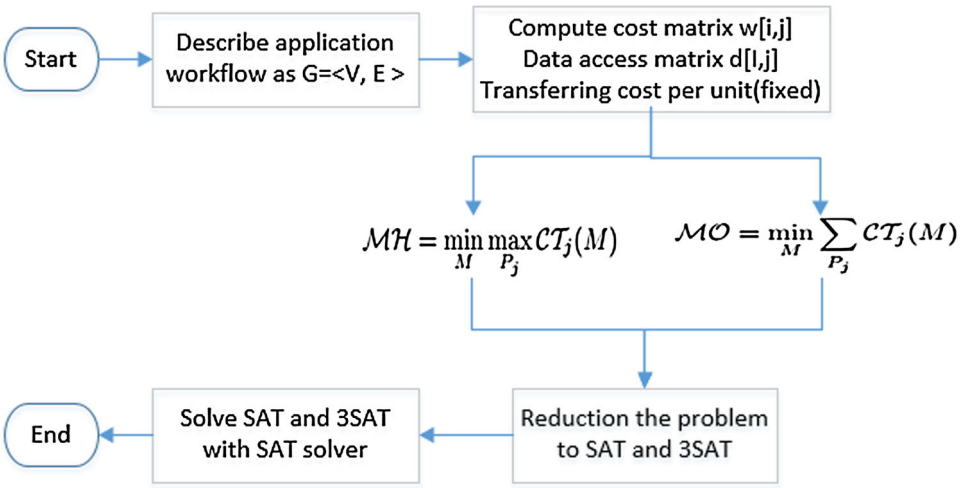

Fig. 3 Using a SAT solver to do task-resource scheduling

author proposed a linear time rescheduling algorithm for the task migration. When it comes to the overall computation complexity of the MCC task scheduling algorithm, it is $O(N^3 \times K)$ where N is the number of tasks and K is the number of cores.

From above descriptions about these static decision making approaches, it is not hard to find that static migration decisions have the advantages of simplicity and low overhead during executions. Compared with them, dynamic decision approaches make more consideration to different runtime conditions and are more adopted by current researches.

3.2 Dynamic task migration decision approaches

6. *Parametric Analysis*: The work of [40] proposes a parametric analysis approach to achieve an optimal program partitioning and scheduling for computation offloading. Applications are divided into tasks according to task control flow graphs (TCFG). And then based on the optimal program partitioning that corresponds to the current values of runtime parameters, the program is transformed into a distributed program which at runtime self-schedules its computation tasks to either a mobile device or a server. The overall process is as shown in Fig. 4. There is a trade-off between computation workload and communication cost when making a partition and migration decision. It obtains program computation workload and communication cost with cost analysis and expresses them as functions of runtime parameters, and then the parametric partitioning algorithm will find the optimal program partitioning corresponding to different ranges of runtime parameters.
7. *$(k + 1)$ Multi-Constraint Partitioning Algorithm*: The paper [41] proposes an adaptive $(k + 1)$ partition algorithm for application offloading. Statically, it partitions an application using a graph cutting algorithm at class level according to the application's features (a dynamic multi-cost graph). According to the partition scheme, the multi-constraint partitioning algorithm obtains a best distribution of the application where one un-offloadable partition will run locally on the mobile device and k offloadable partitions are distributed to k surrogates.
8. *MAUI*: [27] is an architecture where mobile program code can be offloaded to a powerful infrastructure. MAUI decides at runtime which methods should be remotely executed, driven by an optimization engine that achieves the best energy savings possibility under the mobile device's current connectivity constrains. To offload an application, it uses collected samples of CPU utilization and the corresponding smart phone energy consumption to build a

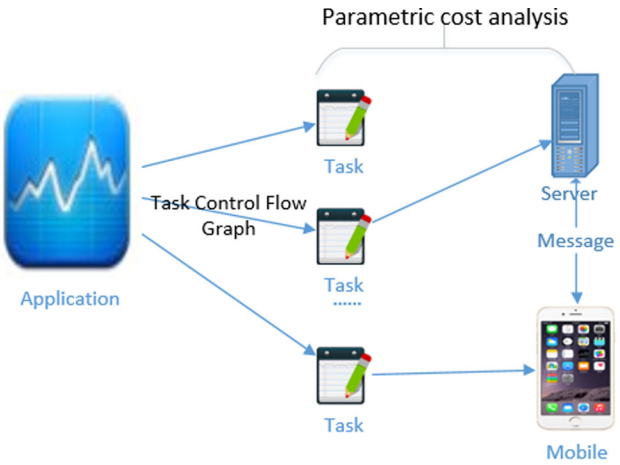

Fig. 4 Parametric analysis for adaptive computation offloading

simple linear model, using the least-square linear regression. Then, it can predict the energy consumption of task executions and optimize the current model using machine learning. Based on the energy consumption model, decision making of application partition and migration is dynamically performed at the method level. High level overview of the MAUI architecture mainly includes two parts, smart phone and MAUI server. Each end has a proxy, a profiler, and a solver. The profilers are used to do device and program profilings, and the MAUI solver uses data collected by the MAUI profiler as input to determine which remotable methods should be executed locally and which should be executed remotely.

9. *Darwin*: In [42], the authors introduced an integrated automation framework called Darwin. It enables workload migrations in a source-target mode. The objectives are to enhance the speed, reduce the cost and inherent risk of performing migrations. During a workload migration, it firstly discovers the source environment, makes analysis to select migration target candidates, and then creates a workload migration map. Darwin configures the target platform(s) environment and application components to carry out the migration at last. As a framework, Darwin is able to architecturally support different variations but still has gaps such as lack of open virtualization format (OVF) integration with the request for migration XML specification.
10. *EOA*: [43] proposes a novel efficient online algorithm (EOA) to solve the migration problem of batch jobs among data centers (DCs). EOA can handle future variabilities and uncertainties of energy sources, and can make fundamental trade-off between energy and bandwidth costs of migrating application data and states. Online job migrations are defined as minimum value problems of the total operation cost (sums of the energy costs at a DC and the bandwidth costs of migrating jobs among DCs) in the cloud. As a computationally efficient algorithm, the complexity of EOA algorithm is $O(logn)$ where n is the total number of servers in the cloud.
11. *Cloud-Vision*: The work [29] introduces an application migration approach to increasing mobiles' performance (response time) by using proximate Cloudlet. A Cloudlet receives processing intensive tasks from mobile devices and then assigns them to itself or remote cloud nodes. It has two schemes for task migrations: equally distribution and greedy style. Apparently in the equally distribution scheme tasks are equally distributed among available cloud servers (or the Cloudlet). In the greedy style scheme, a migration decision is made according to the time of finishing the tasks. The latter approach has better performance, especially when the communication delays of servers are different. But a weak point is that it is difficult to know the communication delays because of the environment's dynamics.
12. *LALTM*: The author of [44] proposed a Java byte code transformation technique for realizing task migrations which aims at minimizing the overhead. It presents how to do migrations for one application focusing on state capturing before a migration and state restoring after execution results return. A migration is triggered by application itself when the execution has reached some special states such as exceptions. The migration mechanism makes use of asynchronous exceptions and byte code instrumentation to perform state capturing which can avoid to poll the migration state and check the migration state repeatedly after every function returns. In the state restoring, a technique, namely Twin Method Hierarchy, is used to minimize the overhead induced by code restorations. The mobile device transfers just enough portion of the stack, methods, and objects to the cloud node. This minimizes the overhead of migrating tasks and allows more flexible migration paths.
13. *EMSO*: [45] presents an energy efficient multisite offloading (EMSO) algorithm that saves energy and reduces time consumption. EMSO divides an application by depth-first search based on a weight object relation graph (WORG). After static analysis to the application, partition and offloading are dynamically performed and can adapt to environment changes. The test result of the system shows that EMSO can significantly reduce energy and time consumption.
14. *MDP migration*: In [46], the authors modeled the service migration problem for Follow Me Cloud (FMC) using a Markov decision process (MDP) and formulated a decision policy to determine whether and where to do service migrations according to the trade-off between migration cost and user perceived quality. The overall process of Markov decision process is as shown in Fig. 5. FMC is a network architecture which enables service mobility across federated data centers (DCs). Following the mobility of a mobile user, the service located in a given DC is migrated at every time when an optimal DC is available. In the MDP model, given a state (current service location), there are a set of actions (migrate or not) which are associated with transitions to other

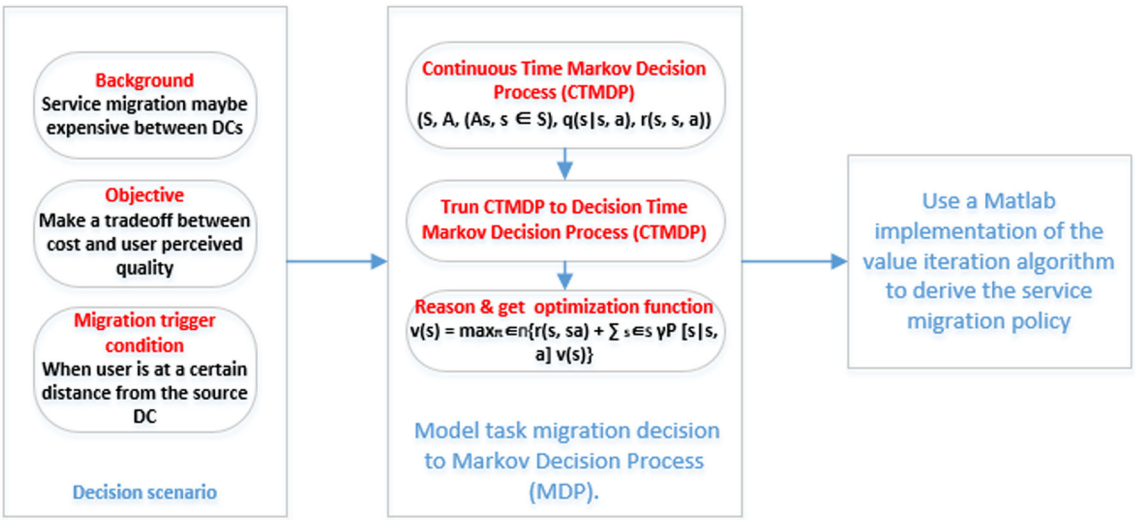

Fig. 5 Markov decision process (MDP) for task migration in Follow Me Cloud

states. For each transition, there is a corresponding reward and cost function. The decision policy is used to find an optimal migration policy which has the max discounted total reward. Numerical results show that the proposed service migration decision mechanism always achieves the maximum expected reward.

15. *OSGi-PC*: OSGi-PC proposed in [47] is based on the de facto Java component standard called OSGi. In OSGi-PC, the work for partitioning application is not yet considered but as it uses a component based approach, the support for application partitioning will not be hard to add. The offloading is done via component migrations across any kind of cloud nodes including the migration from powerful nodes to mobile client nodes.
16. *Mobiles on Cloud Nine*: The paper [48] proposes a solution for task migration in cloud computing systems. Some online migration strategies are developed to make migration decisions according to instantaneous load of the system and estimated execution time of tasks. Following cloud scenarios, migration policies vary from task centric migration where migration decisions are made by the task itself for minimizing its own execution time to cloud wide task migration decided by the cloud provider considering the whole system's performance. When making task migration decisions, three factors are explicitly taken into account: the multi-tenancy effect resulting from interaction of co-located VMs in a server, the cost of migrating source code and input data of the task, and the cost of transferring final result to the user. A migration should be occurred only if it is beneficial for the processing time of a task.
17. *Jade*: Jade [49] is an energy aware computation offloading system for Android mobile devices. It transports computation which contains remotable tasks from an Android device to servers running on the cloud. At runtime, Jade gathers devices' information and tasks' information and uses multi-level scheduling algorithm to offload tasks and balance the workload of servers. Jade was evaluated with two applications (face detection and path finding), and the result showed that it can effectively reduce up to 35 % of average power consumption for mobile devices.
18. *DCOG*: Decentralized computation offloading game (DCOG) [50] is a computing offloading decision mechanism. The authors considered communication and computation aspects of users' tasks and formulated the computation offloading decision-making problem among multiple mobile device users as a decentralized computation offloading game which admits a Nash equilibrium. Within the offloading decision mechanism, the authors used a slotted time structure to do interference measurement and to solve the decision update contention problem cyclically and finally determined whether users' tasks should be executed locally or migrated to the cloud.
19. *DECM*: [51] proposes a dynamic energy aware Cloudlet based mobile cloud computing model (DECM) which uses Cloudlets with dynamic programming to reduce additional energy consumption during wireless communication. DECM manages

and optimizes the cloud based infrastructure usages and services. A dynamic Cloudlet is the main component of DECM to find better connections (connecting to another Cloudlet or cloud or itself) for users. DECM uses recursion to solve service migration problems.

4 Comparisons of decision-making approaches

In Table 2, we summarize the decision-making approaches for task migrations aforementioned from the following aspects: the scenario of task migration problem, the algorithm proposed to make a decision for task migration, the migration granularity, whether the decision making is dynamic or static, what factors are considered when making decisions for task migrations, the complexity of the decision algorithm, and finally the decision mode in which centralized or distributed decision making is adopted.

4.1 Decision-making algorithms for task migration in MCC

There are a number of application migration algorithms proposed so far. A common approach is graph based partitioning such as HELVM [41] and operation data flow graph partitioning in [36] which can make better analyses to an application and get a migration scheme easily. Some other partition methods can be used under certain circumstances such as a SAT solver which is used to solve SAT problems transformed from minimization problems [38]. MDP [46] is associated with state transitions of a task migration, and DVFS algorithm [39] is used to reduce energy consumption as described in Table 2. Each decision-making algorithm has complexity different from others which leads to different migration overheads [44].

4.2 Static or dynamic task migration decisions

Some of the algorithms make migration decision statically before an application's execution [36, 39]. Therefore, a program is partitioned during development, and task migration is usually performed only once within static decision approaches. Some migration decisions are made in a dynamic way taking into account the runtime environment and application executions. Task migrations usually are triggered when program executions have reached some special states such as in [37] or an attribute value exceeds a threshold such as in [46].

Static algorithms are relatively simple and lead to low overhead during executions without monitoring computing contexts, but there are limitations considering dynamic situations, such as static decision approaches are valid only when parameters can be accurately predicted in advance. In contrast, dynamic decisions can adapt to different runtime conditions and can be particularly useful since most information of computing environment cannot be known in advance due to the dynamic nature of mobile cloud computing environments (fluctuating network bandwidth, constantly changing memory and CPU). However, dynamic migration decisions may lead to higher runtime overhead.

4.3 Decision mode for task migrations

Task migration mechanisms may be centralized whose decisions are made by a central migration manager or processor considering the whole system's condition or performance [46, 47]. These centralized decision-making approaches are relatively simple but may cause bad performance to one specific computing node or task. Currently, there are some distributed decision approaches proposed such as [44, 48] in which the decision making responsibility belongs to several nodes for better resources utilization. Servers or tasks possibly make autonomous decisions independently or make cooperative decisions, not only considering its own objectives but also that of other nodes.

4.4 Decision factors considered for meaningful migrations

In order to make migrations meaningful, some factors need to be considered when making decisions as shown in Fig. 6.

1. *Applications characteristics*. Only computing intensive or communication intensive applications and applications requiring a large amount of data which is remotely located need to do task migrations. Otherwise, the benefit is not enough to offset the cost of migration produces by communication or latency. Therefore, when deciding whether and where to do migration, we must consider some applications' or tasks' information such as data volume, and computing time [38, 41, 45, 50].
2. *Mobile devices' and cloud's characteristics*. Mobile cloud environment is heterogeneous (in terms of the hardwares, platforms, the services, and networks). We need to know status and features of nodes, such as performance, availabilities, reliabilities, distance, and elasticity and costs. A migration should be performed when there is no enough resources to support applications' execution in mobile or cloud nodes, or in order to have better execution performance. These characteristics include CPU processing capacities of

Table 2 Comparisons of decision-making approaches in MCC

No.	MS	MO	MA	MG	DS	DF	AC	MM
(6) [40]	Different execution environment, input parameters and data files will make difference in applications execution.	Minimize total cost of computing, communication and scheduling	Parametric analysis partition	Task	D	Computing and communication time, data volume	$O(n^{\frac{d}{2}})$	Program itself
(7) [41]	Offloading decision should satisfy multiple constraints imposed by users or mobile resources.	Reduce resource consumption	HELVM (heavy-edge, light-vertex matching)	Class	D	CPU, memory, bandwidth, data communication	$O(n^4)$	Central
(1) [36]	Sensor network applications are data intensive, while the environment is heterogeneous and resource-constrained.	Minimize a combination of network load and CPU consumption	Operation data flow graph partition	Operation	S	Computing and communication time, CPU and memory utilization	NA (not available)	Central
(2) [37]	Network I/O performance would affect data-intensive applications' performance hosted by networked VMs.	Minimize the data access time	Traverse	VM	S	Data storage information, network status	$O(n^2)$	Central
(8) [27]	Offloading should be fine-grained and can maximize energy saving with minimal burden on programmer.	Achieve the best energy savings	NG (not given)	Method	D	CPU, energy consumption, network bandwidth	NA	Central
(9) [42]	Migrating workload to cloud in a smooth and cost effective way, with minimal disruption, is a challenge.	Enhance speed, reduce inherent risk of migration	/	Application	D	CPU utilization, workload, resource information	/	Central
(3) [5]	Migration of mobile applications lacks of flexibility and widespread adaptability.	Optimize execution time or energy consumption	NG	Thread	D	Time and bandwidth	NA	Central
(10) [43]	Judicious job migration approaches are needed to handle the trade-off between energy and bandwidth cost.	Balance energy saving and bandwidth cost in data center operation	EOA (efficient online algorithm)	Job	D	Energy price, resource information, bandwidth	$O(n^2)$	Central
(11) [29]	With effective partition and migration, face recognition can benefit from the cooperation of mobiles and cloud.	Minimize the overall response time (latency, compute time) of application	Fixed/greedy algorithm	Task	D/S	Bandwidth, task execution time	$O(n)/O(n^2)$	Central
(4) [38]	Task-resource management is severely related to the efficiency of cloud computing systems.	Minimize the total cost (execution time, access cost) of servers	SAT solver	Task	S	Execution time, communication cost, data volume	$O(n^2)$	Central
(12) [44]	There is insufficient in current application-level task migration approaches.	Reduce overhead and increase flexibility of migration	/	Stack/method/object	D	Migration overload, time consumption, resource utilization	/	Program itself

Table 2 continued

No.	MS	MO	MA	MG	DS	DF	AC	MM
(13) [45]	Offloading computation to multiple servers/sites is more promising for mobile devices to save energy.	Minimize the total energy consumption and time cost of mobile devices	EMSO algorithm based on WORG	Object	D	Energy, bandwidth, application feature	$O(n^4)$	Central
(14) [46]	Service migration can bring profit but also may be expensive given the communication between DCs.	Trade-off between cost and user-perceived quality	MDP	Service	D	Migration time consumption, QoS, mobility	NG	Central
(15) [47]	A supporting infrastructure of component/service migrations is needed for small devices.	Enhance memory utilization	OSGi-PC	Component	D	Memory information of servers	$O(n)$	Central
(5) [39]	To get benefit from task migration for augmenting mobile devices, many factors should be considered.	Minimize energy consumption under hard completion time constraint	Initial, rescheduling, and DVFS algorithm	Task	S	Execution time, task precedence, bandwidth, energy	$O(n^3)$	Local/cloud
(16) [48]	Task execution time may vary in cloud because of sharing resources with other co-located tasks.	Minimize execution time of tasks	NG	Task	D	Server load, time, communication cost, mobility, multi-tenancy effect	NA	Server/task-centric/cloud-wide
(17) [49]	Performance of mobiles conflicts with longer battery life.	Reduce energy consumption	Work stealing and HRRN	Task	D	Device status, energy and execution time	NA	Server and client
(18) [50]	Network competition can cause low energy efficiency for migration.	Minimize computation overhead	/	Task	D	Bandwidth, power, CPU, task's size, user preference	$O(n)$	Device-self
(19) [51]	Migration may result in extra energy waste and latency.	Minimize energy consumption	Recursive algorithm	Service	D	Energy consumption and execution time	$O(n^3)$	Central

The papers are sorted by the publication year, and the abbreviated columns represent the following: *MS* migration scenario, *MO* migration objective, *MA* migration algorithm, *MG* migration granularity, *DS* dynamic or static, *DF* decision factors, *AC* algorithm complexity, *MM* migration mode (central/distributed)

both mobile devices and cloud [36, 42, 49], mobile device energy information and mobility [39, 46], and load and resource utilization such as memory and storage on cloud [43, 47, 48].

3. *Network environment*. Mobile cloud computing is developed depending on the utilization of networks. There is no doubt that communication technologies and network capabilities have significant impacts on task migrations, especially wireless networks between computing nodes. As said in [37], network I/O performance would affect the overall application performance significantly, especially on data intensive applications which need frequent data communication. Consequently, network environment information is of particular importance to task migrations. Moreover, network condition is varied with time, so it usually needs realtime monitoring or prediction. So many researches have brought the network factor into decision making for task migrations such as in [5, 29, 46, 48].
4. *Users' preferences and requirements*. Users should have rights to decide whether to do migrations, and migrations should satisfy users' requirements such as the demand to finish application executions within a certain time. Users' preferences and requirements are probably the most important factors in decision making of task migrations [46, 50].

Mobile cloud computing environment is complex, and it may be difficult to obtain all these information when making task migration decisions. As a result, getting

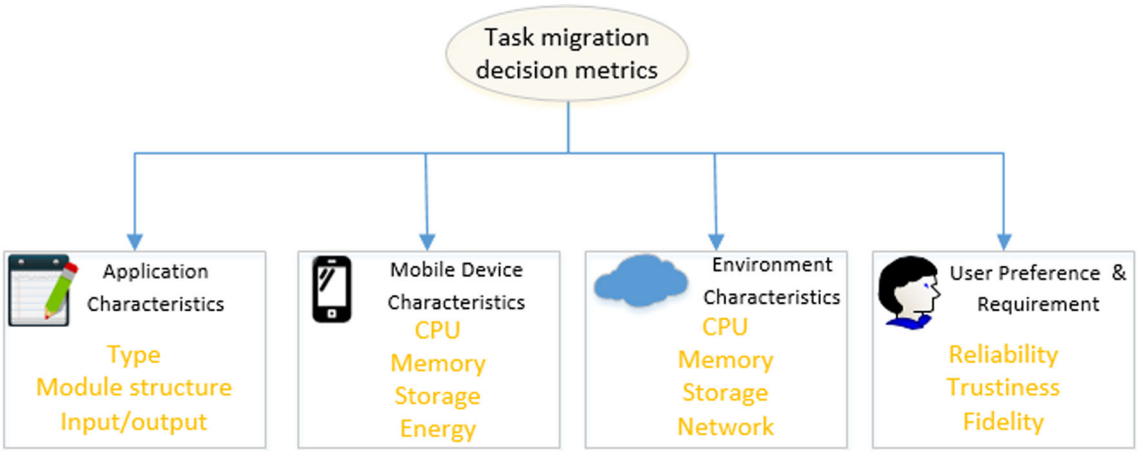

Fig. 6 Decision factors considered for meaningful migrations

optimal solutions may be not possible in real cloud scenarios. Instead, suboptimal solutions can be tried choosing a part of decision factors like most of proposed solutions do.

4.5 Migration granularity

In terms of migration granularities, a finer granularity level for instance thread level or object and method level outsources computational load at a refined level, which requires intensive monitoring and synchronization mechanisms on mobile devices and cloud nodes at runtime [4, 5]. Furthermore, a finer granularity level has the issue of ensuring consistency in distributed executions of mobile applications. Higher level of granularities such as module level [52] and entire application level [28] may be more simple and have lighter overhead but may result in increased data transmission and potentially have high security threats.

- Thread level: a thread level partition and migration[4, 5].
- Method level: partitioning occurs at a method level and intensive methods are migrated to remote servers [27].
- Object level: an entire object is migrated to a remote environment [44, 45].
- Class level: objects of the same class are offloaded to the same server [28].
- Module level: an entire module or bundle is migrated to remote processing [52].
- Task level: a application is a collection of tasks and some tasks are migrated [4, 5].
- Entire application: the whole application is offloaded to remote servers [28, 44].

4.6 Migration motivationa and objectives

Application or task migration mechanisms aim at augmenting processing potentials of resources constrained mobile devices in MCC and makes better executions for user tasks. For doing this, diverse objectives are considered: saving energy [27, 53] and processing time [40, 48], balancing migration incomes with overhead [46], and maximizing utilization of system resources [41, 47]. They are all for obtaining best execution performance and satisfying user requirements under various scenarios.

5 Challenges of decision making for task migration in MCC

Although there have been research efforts for task migration in MCC, it is still in early stage and there is not yet a matured approach to making this really work efficiently and effectively in a large scale deployment environment. The main challenges are:

- *Comprehensive context awareness* should be achieved in order to facilitate decision making for task migrations. It is known that context awareness is particular important when it comes to mobile devices [54]. Location, weather conditions, security, and other relevant contexts should be managed besides battery, power consumption, bandwidth, application data volume, and server workload. Currently, the existing approaches on decision making for task migration consider limited contexts such as power consumption and bandwidth. However, this decision process can be complicated considering weather conditions, trustwor-

thiness, security, and privacy of cloud nodes. This calls for a dedicated context awareness framework which supports task migrations.

- *Deep context awareness for big data computing environment* is needed. Big data is arising from various applications, and it is becoming more difficult to obtain the so-called deep contexts for big data computing due to complexities of big data [56]. Deep learning has been receiving great popularity from both academics and industries due to its excellent performance in many practical problems [55], such as image and voice recognition. Deep learning may be used to help achieve deep level awareness of contexts which can be called in-depth context awareness that can help to make better decisions [56].
- *Unified standards* for interacting with both mobile clients and cloud servers. Currently, every approach is developing in-house closed framework for supporting task migrations in a unique way which does not satisfy interoperability. There arises the requirement for unifying these work including framework, communication protocol, and decision-making process. Unification helps to adopt a standard way to promote MCC's industrialization and commercialization.
- *Large-scale deployment and testing* of task migrations which can be globally optimized. This is to make sure task migrations can consistently work in for example a big smart city application. The current researches are mainly tested in a small scale in laboratory. In view of this situation, we need to have a larger scale of testing for globally optimized task migrations, considering possibly conflicting optimization parameters. This also calls for more efforts for the research of light weight but powerful decision mechanisms to achieve this.
- *Adaptive and intelligent decision algorithms* to make more adaptive migration decisions. Current task migration solutions in mobile cloud computing use traditional algorithms to make decisions as discussed in Sect. 3 which are limited in terms of efficiency and performance. Task migration decisions may need to combine with the latest development of artificial intelligence. Decision making can also use interdisciplinary theories and approaches such as auction or game theory in economic theories to break through the limitation of existing algorithms and obtain better migration performance.
- *Distributed decision mechanisms* are needed in order to make migration more flexible and accurate. Currently, the majority of work is using a centralized way to make decisions, which may be a bottleneck when there are emergent situations or a global decision is not possible. Centralized decision making needs a powerful central decision component to consider the whole system and to communicate with all the machines and tasks. This inevitably results in heavy network cost. Distributed decision mechanisms allow tasks or machines to make migration decisions according to their own situations. Thus, it is more fine grained, flexible, and accurate. Tasks and machines can get benefit from distributed task migration decisions.

6 Conclusion

Mobile cloud computing is an important computing paradigm today. A decade of efforts by many researchers have developed core concepts, techniques, and mechanisms to provide a solid foundation for progress in this area. Task migration is the main approach in MCC which offloads tasks in small nodes to other powerful cloud nodes. Currently, there are a number of approaches to decision making for task migrations. In this paper, we present a relatively comprehensive survey on these task migration approaches and analyze their solutions to decision making. They differ from each other in application migration algorithms, granularities, decision-making factors, and decision modes (centric overall decision or distributed decision at a server level or task level). Even though we can benefit from task migrations, there are still challenges that need future efforts, which include in-deep context awareness for big data environments, distributed migration decision mechanisms and algorithms, unified stands for interactions, and other issues.

Acknowledgments This research was supported by the International S&T Cooperation Program of China (ISTCP, 2013DFA10980).

References

1. Hung PP, Bui TA, Morales MAG et al (2014) Optimal collaboration of thinthick clients and resource allocation in cloud computing. Pers Ubiquitous Comput 18(3):563–572
2. Kristensen MD (2010) Empowering mobile devices through cyber foraging. Ph. D. Dissertation, Aarhus University
3. Dinh HT, Lee C, Niyato D et al (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mobile Comput 13(18):1587–1611
4. Dou A, Kalogeraki V, Gunopulos D et al (2010) Misco: a mapreduce framework for mobile systems. In: Proceedings of the 3rd international conference on pervasive technologies related to assistive environments. ACM, p 32
5. Chun BG, Ihm S, Maniatis P et al (2011) Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on computer systems. ACM, pp 301–314
6. Marinelli EE (2009) Hyrax: cloud computing on mobile devices using MapReduce. Carnegie-Mellon University, Pittsburgh school of computer science
7. March V, Gu Y, Leonardi E et al (2011) Cloud: towards a new paradigm of rich mobile applications. Proc Comput Sci 5:618–624

8. Lu Y, Li S, Shen H (2011) Virtualized screen: a third element for cloud–mobile convergence. MultiMed IEEE 18(2):4–11
9. Lomotey RK, Deters R (2014) Using a cloud-centric middleware to enable mobile hosting of Web services: mHealth use case. Pers Ubiquitous Comput 18(5):1085–1098
10. Shiraz M, Gani A (2014) A lightweight active service migration framework for computational offloading in mobile cloud computing. J Supercomput 68(2):978–995
11. Kakadia D, Saripalli P, Varma V (2013) MECCA: mobile, efficient cloud computing workload adoption framework using scheduler customization and workload migration decisions. In: Proceedings of the first international workshop on Mobile cloud computing & networking. ACM, pp 41–46
12. Gu X, Nahrstedt K, Messer A et al (2004) Adaptive offloading for pervasive computing. Pervasive Comput IEEE 3(3):66–73
13. Abolfazli S, Sanaei Z, Ahmed E et al (2014) Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. Commun Surv Tutor IEEE 16(1):337–368
14. http://www.mobilecloudcomputingforum.com
15. AEPONA (2010) Mobile cloud computing solution brief. White Paper
16. Liu L, Moulic R, Shea D (2010) Cloud service portal for mobile device management. In: 2010 IEEE 7th international conference on e-Business engineering (ICEBE). IEEE, pp 474–478
17. Chun BG, Maniatis P (2009) Augmented smartphone applications through clone cloud execution. HotOS 9:8–11
18. Kallonen T, Porras J (2006) Use of distributed resources in mobile environment. In: International conference on software in telecommunications and computer networks, 2006. SoftCOM 2006. IEEE, pp 281–285
19. Ververidis CN, Polyzos GC (2008) Service discovery for mobile ad hoc networks: a survey of issues and techniques. Commun Surv Tutor IEEE 10(3):30–45
20. Preuveneers D, Berbers Y (2005) Adaptive context management using a component-based approach. In: Distributed applications and interoperable systems. Springer, Berlin
21. Miraoui M, Tadj C, Fattahi J et al (2011) Dynamic context-aware and limited resources-aware service adaptation for pervasive computing. Adv Softw Eng 2011:7
22. Makris P, Skoutas DN, Skianis C (2013) A survey on context-aware mobile and wireless networking: on networking and computing environments' integration. Commun Surv Tutor IEEE 15(1):362–386
23. Yuan B, Herbert J (2014) Context-aware hybrid reasoning framework for pervasive healthcare. Pers Ubiquitous Comput 18(4):865–881
24. Kumar K, Liu J, Lu YH et al (2013) A survey of computation offloading for mobile systems. Mobile Netw Appl 18(1):129–140
25. Balan RK, Satyanarayanan M, Park SY et al (2003) Tactics-based remote execution for mobile computing. In: Proceedings of the 1st international conference on Mobile systems, applications and services. ACM, pp 273–286
26. Frey S, Hasselbring W (2011) The cloudmig approach: model-based migration of software systems to cloud-optimized applications. Int J Adv Softw 4(3 and 4):342–353
27. Cuervo E, Balasubramanian A, Cho D et al (2010) MAUI: making smart phones last longer with code offload. In: Proceedings of the 8th international conference on Mobile systems, applications, and services. ACM, pp 49–62
28. Satyanarayanan M, Bahl P, Caceres R et al (2009) The case for vm-based cloudlets in mobile computing. Pervasive Comput IEEE 8(4):14–23
29. Soyata T, Muraleedharan R, Funai C et al (2012) Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: 2012 IEEE symposium on computers and communications (ISCC). IEEE, pp 59–66
30. https://en.wikipedia.org/wiki/Mobile_network_operator
31. AT&T cloud services. [Online]. http://www.business.att.com/enterprise/Family/hosting-services/cloud/
32. Sanaei Z, Abolfazli S, Gani A et al (2012) SAMI: Service-based arbitrated multi-tier infrastructure for Mobile Cloud Computing. In: 2012 1st IEEE international conference on communications in china workshops (ICCC). IEEE, pp 14–19
33. Zhao B, Xu Z, Chi C et al (2010) Mirroring smart phones for good: a feasibility study. In: Mobile and ubiquitous systems: computing, networking, and services. Springer, Heidelberg, pp 26–38
34. Black M, Edgar W (2009) Exploring mobile devices as Grid resources: using an x86 virtual machine to run BOINC on an iPhone. In: 2009 10th IEEE/ACM international conference on grid computing. IEEE, pp 9–16
35. Huerta-Canepa G, Lee D (2010) A virtual cloud computing provider for mobile devices. In: Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond. ACM, p 6
36. Newton R, Toledo S, Girod L et al (2009) Wishbone: profile-based partitioning for sensornet applications. NSDI 9:395–408
37. Piao JT, Yan J (2010) A network-aware virtual machine placement and migration approach in cloud computing. In: 2010 9th international conference on grid and cooperative computing (GCC). IEEE, pp 87–92
38. Gorbenko A, Popov V (2012) Task-resource scheduling problem. Int J Autom Comput 9(4):429–441
39. Lin X, Wang Y, Xie Q et al (2015) Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. Services Comput IEEE Trans 8(2):175–186
40. Wang C, Li Z (2004) Parametric analysis for adaptive computation offloading. In: ACM SIGPLAN notices, vol 39, no. 6. ACM, pp 119–130
41. Ou S, Yang K, Liotta A (2006) An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In: Fourth annual IEEE international conference on pervasive computing and communications, PerCom 2006, vol 10. IEEE, p 125
42. Ward C, Aravamudan N, Bhattacharya K et al (2010) Workload migration into clouds challenges, experiences, opportunities. In: 2010 IEEE 3rd international conference on cloud computing (CLOUD). IEEE, pp 164–171
43. Buchbinder N, Jain N, Menache I (2011) Online job-migration for reducing the electricity bill in the cloud. In: NETWORKING 2011. Springer, Berlin, pp 172–185
44. Ma RKK, Wang CL (2012) Lightweight application-level task migration for mobile cloud computing. In: 2012 IEEE 26th international conference on advanced information networking and applications (AINA). IEEE, pp 550–557
45. Niu R, Song W, Liu Y (2013) An Energy-efficient multisite offloading algorithm for mobile devices. Int J Distrib Sensor Netw. 2013:9 doi:10.1155/2013/518518
46. Ksentini A, Taleb T, Chen M (2014) A Markov decision process-based service migration procedure for follow me cloud. In: 2014 IEEE international conference on communications (ICC). IEEE, pp 1350–1354
47. Zhang WS, Chen LC, Liu X et al (2014) An OSGi-based flexible and adaptive pervasive cloud infrastructure. Sci China Inf Sci 57(3):1–11
48. Gkatzikis L, Koutsopoulos I (2014) Mobiles on cloud nine: efficient task migration policies for cloud computing systems. In: 2014 IEEE 3rd international conference on cloud networking (CloudNet). IEEE, pp 204–210
49. Qian H, Andresen D (2015) An energy-saving task scheduler for mobile devices. In: 2015 IEEE/ACIS 14th international conference on computer and information science (ICIS). IEEE, pp 423–430

50. Chen X (2015) Decentralized computation offloading game for mobile cloud computing. IEEE Trans Parallel Distrib Syst 26(4):974–983
51. Gai K, Qiu M, Zhao H et al (2016) Dynamic energy-aware Cloudlet-based mobile cloud computing model for green computing. J Netw Comput Appl 59:46–54
52. Chun BG, Maniatis P (2010) Dynamically partitioning applications between weak devices and clouds. In: Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond. ACM, 2010, p 7
53. Hung SH, Shih CS, Shieh JP et al (2012) Executing mobile applications on the cloud: framework and issues. Comput Math Appl 63(2):573–587
54. Grønli TM, Ghinea G, Younas M (2014) Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. Pers Ubiquitous Comput 18(4):883–894
55. Zhang K, Chen X (2014) Large-scale deep belief nets with mapreduce. IEEE Access 2:395–403
56. Zhang Weishan, Duan Pengcheng, Li Zhongwei, Lu Qinghua, Gong Wenjuan, Yang Su (2015) A deep awareness framework for pervasive video cloud. IEEE Access 3:2227–2237