Mobile Netw Appl (2013) 18:129–140
DOI 10.1007/s11036-012-0368-0

A Survey of Computation Offloading for Mobile Systems

Karthik Kumar · Jibang Liu · Yung-Hsiang Lu · Bharat Bhargava

Published online: 10 April 2012

Abstract Mobile systems have limited resources, such as battery life, network bandwidth, storage capacity, and processor performance. These restrictions may be alleviated by *computation offloading*: sending heavy computation to resourceful servers and receiving the results from these servers. Many issues related to offloading have been investigated in the past decade. This survey paper provides an overview of the background, techniques, systems, and research areas for offloading computation. We also describe directions for future research.

Keywords mobile cloud computing · computation offloading · survey · energy · performance

1 Introduction

Advancements in computing technology have expanded the usage of computers from desktops and mainframes to a wide range of mobile and embedded applications, including surveillance, environmental sensing, GPS navigation, mobile phones, autonomous robots, etc. Many of these applications run on systems with limited resources. For example, mobile phones are battery-powered. Environmental sensors have small physical sizes, slow processors, and small amounts of storage. Most of these applications use wireless networks and their bandwidths are orders-of-magnitude lower than wired networks. Meanwhile, increasingly complex programs are running on these systems—for example, video processing on mobile phones and object recognition on mobile robots. Thus there is an increasing gap between the demand for complex programs and the availability of limited resources.

Offloading is a solution to augment these mobile systems' capabilities by migrating computation to more resourceful computers (i.e., servers). This is different from the traditional client-server architecture, where a thin client *always* migrates computation to a server. Computation offloading is also different from the migration model used in multiprocessor systems and grid computing, where a process may be migrated for load balancing [62]. The key difference is that computation offloading migrates programs to servers outside of the users' immediate computing environment; process migration for grid computing typically occurs from one computer to another within the same computing environment, i.e., the grid. Offloading is in principle similar to efforts like SETI@home [5], where requests are sent to surrogates for performing computation. The difference is that SETI@home is a large scale distributed computing effort involving several thousands of users, whereas offloading is typically used to augment the computational capability of a resource constrained device for a single user. The terms "cyber foraging" and "surrogate computing" are also used to describe computation offloading. In this paper, we use the above terms interchangeably.

A significant amount of research has been performed on computation offloading: making it feasible, making offloading decisions, and developing offloading infrastructures, as shown in Table 1. Prior to 2000, researchers mostly focused on making offloading feasible. This

K. Kumar (✉) · J. Liu · Y.-H. Lu · B. Bhargava
Purdue University, West Lafayette, IN, USA
e-mail: karthik.mdk@gmail.com

Y.-H. Lu
e-mail: yunglu@purdue.edu

Table 1 Research focuses about offloading in the past 15 years

Years	1996–2000	2001–2005	2006–2010
Feasibility, importance	[20, 38, 40, 54, 61, 63, 79, 81, 82]		
Decision		[8, 13, 26, 29, 30, 41, 44–46, 67, 74]	[35, 52, 56, 80, 83]
Infrastructures		[4, 25, 27]	[28, 32, 37, 58, 60, 66, 70, 71, 77, 84]

was primarily due to limitations in wireless networks, such as low bandwidths. In early 2000s, the focus moved to developing algorithms for making offloading decisions i.e, decide *whether* offloading would benefit mobile users. Improvements in virtualization technology, network bandwidths, and cloud computing infrastructures, have shifted the direction of offloading. These developments have made computation offloading more practical. This paper surveys the development of computation offloading for mobile systems over the last 15 years, and identifies directions for future research.

Offloading may save energy and improve performance on mobile systems. However, this usually depends on many parameters such as the network bandwidths and the amounts of data exchanged through the networks. Many algorithms have been proposed to make offloading decisions to improve performance or save energy [8, 13, 26, 29, 30, 35, 41, 44–46, 52, 56, 67, 74, 80, 83]. The decisions are usually made by analyzing parameters including bandwidths, server speeds, available memory, server loads, and the amounts of data exchanged between servers and mobile systems. The solutions include partitioning programs [13, 15, 35, 44–46, 52, 60, 72, 74, 83] and predicting parametric variations in application behavior and execution environment [26, 30, 36, 67, 80].

Offloading requires access to resourceful computers for short durations through networks, wired or wireless. These servers may use *virtualization* to provide offloading services so that different programs and their data can be isolated and protected. Isolation and protections have motivated research on developing infrastructures for offloading at various granularities [4, 25, 27, 28, 32, 37, 58, 60, 66, 70, 71, 77, 84]. Offloading may be performed at the levels of methods [66], tasks [84], applications [83], or virtual machines [16]. Java RMI, .NET remoting, and RPC (remote procedure call) are several mechanisms enabling offloading at the class and object level. Techniques have been proposed to enable offloading at the virtual-machine level; for example, Chun and Maniatis [16] use cloud computing to enable offloading. Cloud computing allows *elastic* resources and offloading to multiple servers; it is an enabler for computation offloading. Various infrastructures and solutions have been proposed to improve offloading: they deal with various issues such as transparency to users, privacy, security, mobility, etc. All of these infrastructures and solutions address different issues associated with offloading.

The purpose of this paper is to acquaint readers with research on computation offloading for mobile systems. This paper provides an overview of the motivations, techniques, technological enablers, and architectures for computation offloading. It surveys the common approaches used to make offloading decisions, and classifies these approaches based on various factors, including

- why to offload (improve performance or save energy)
- when to decide offloading (static vs dynamic)
- what mobile systems use offloading (laptops, PDAs, robots, sensors)
- types of applications (multimedia, gaming, calculators, text editors, predictors)
- infrastructures for offloading (grid and cloud computing).

This paper serves as a collective reference for the algorithmic mechanisms and the associated infrastructures, and identifies existing barriers and directions for research. The paper is organized as follows: Section 2 describes a brief history of enabling technologies. Section 3 explains two objectives for offloading: reduce execution time and save energy, and describes infrastructures and tools developed to address the challenges of offloading. Section 4 describes why offloading will become increasingly important in the years to come, and Section 5 concludes the paper.

As apparent throughout this paper, *many* studies have been conducted on topics related to computation offloading and a comprehensive survey of all studies would be impossible. Hence, this paper does not intend to provide a complete survey of the field. The references are selected based on our limited knowledge of the topics, as well as creating a coherent flow of this paper. Readers must be aware that some important papers may not be included in this survey due to the limited length.

2 Enabling technology

This section describes some enabling technologies for computation offloading. Figure 1 shows how various

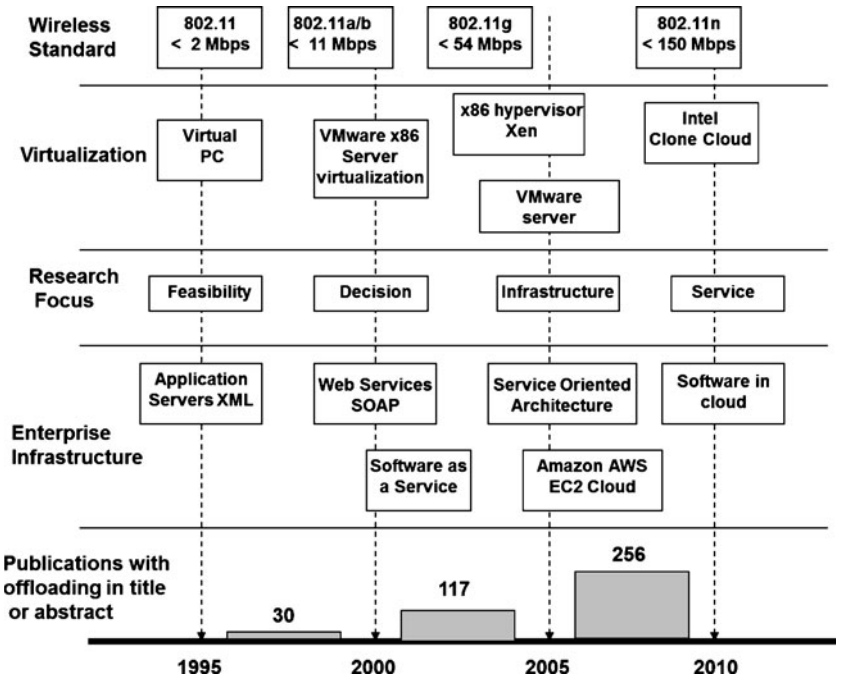

Fig. 1 Enabling technologies for computation offloading. The paper counts are obtained from IEEE Xplore

technological advancements contributed to offloading. The graph shows the number of publications with the terms "offloading" in the title or abstract obtained by searching IEEE Xplore. In the following subsections, we discuss two significant enablers for offloading: (1) wireless networks and mobile agents and (2) virtualization and cloud computing.

2.1 Wireless networks and mobile agents

Until late 1990s, unstable and intermittent wireless network connectivity and low bandwidths were the main problem for mobile systems. The focus was building wireless data networks (in particular WiFi) to facilitate mobility. These improvements spurred many research activities on mobile computing, including mobile agents.

Mobile agents are autonomous programs that can control their movement from machine to machine in a heterogeneous network. Mobile agents introduced the concept of migrating computation from mobile devices. Infrastructures for mobile agents were targeted to achieve platform independence and used technologies like Java and XML [38, 40, 81, 82]. Kotz et al. [40] suggest using mobile agents for accessing Internet resources from a portable device as a solution to poor network connections, variable network addresses and signal quality. They propose Agent TCL: a mobile agent that can be written in Tcl, Java, and Scheme. Wong et al. [81, 82] suggest using Java, mobile agents and XML technologies to compose an enterprise platform that is application-independent, portable and efficient. They propose Concordia: a Java-based infrastructure for mobile agents with design goals of flexible agent mobility, agent collaboration, agent persistence, reliable agent transmission, and agent security. Joseph et al. [38] present a toolkit called Rover for mobile information access: it uses relocatable dynamic objects and queued remote procedure calls to overcome connectivity problems. All these technologies focus on migrating computation for mobile devices, network connectivity, and leveraging Java for developing platform-independent applications.

2.2 Virtualization and cloud computing

Virtualization was first developed in the 1960s by IBM as a way to logically partition large mainframe computers into smaller, independent computing units [23]. This enabled multitasking: the ability to run multiple applications and processes at the same time. Multitasking was necessary at that time because of mainframes' high costs. Virtualization lost popularity during the 1980s and early 1990s when inexpensive x86 desktop computers became popular [68]. Rather than sharing resources centrally in the mainframe model, organizations used the low-cost desktops for their computational needs. However, new problems have emerged including

- Under utilization: Typical deployments have very low utilization of total computing capacity. Users want to run only a few applications per computer

to obtain better response time. As a result, many computers are under-utilized.
- Security: Desktops are often managed by individual users and they have to regularly apply security patches. Otherwise, the computers can become vulnerable.
- Operational costs: The total cost of ownership can grow rapidly for supporting increasing numbers of desktops and laptops, and for upgrading and updating software. Moreover, these computers may waste power as they are often kept on 24 h.

Virtualization has emerged as a solution over the last decade by making it possible to run multiple operating systems and multiple applications on the same computer (or a set of computers) simultaneously, increasing utilization and flexibility [3, 10, 51, 68]. Since different types of virtual machines can be created, users can scale the number of virtual machines based on demand. Due to virtualization, these machines have separation and protection. Cloud computing uses virtualization to offer computing as a service; users can "lease" computing resources based on their requirements. This paper focuses on mobile systems and they can use cloud computing for offloading. An overview of cloud computing, and its potential to influence the future of computing can be found in [6]. Several other articles discussing cloud computing applications, research, and implementations can be found in [11, 12, 31, 55, 73, 78].

3 Offloading decisions

Since offloading migrates computation to a more resourceful computer, it involves making a decision regarding *whether* and *what* computation to migrate. A vast body of research exists on offloading decisions for (1) improving performance and (2) saving energy. Sections 3.1 and 3.2 describe these two purposes for offloading. Section 3.3 provides a taxonomy and surveys existing studies. Section 3.4 describes some of the research areas to improve offloading, and surveys some infrastructures and solutions.

3.1 Improve performance

Offloading becomes an attractive solution for meeting response time requirements on mobile systems as applications become increasingly complex [9]. Another goal is meeting real-time constraints. For example, a navigating robot may need to recognize an object before it collides with the object; if the robot's processor is too slow, the computation may need to be offloaded [52, 69]. Another application is context-aware computing [34]—where multiple streams of data from different sources like GPS, maps, accelerometers, temperature sensors, etc need to be analyzed together in order to obtain real-time information about a user's context. In many of these scenarios, the limited computing speeds of mobile systems can be enhanced by offloading.

The condition for offloading to improve performance can be formulated below. Without loss of generality, we can divide a program into two parts: one part that must run on the mobile system and the other part that *may* be offloaded. The first part may include user interface and the code that handles peripherals (such as the mobile system's camera). Let s_m be the speed of the mobile system. Suppose w is the amount of computation for the second part. The time to execute the second part on the mobile system is

$$\frac{w}{s_m}. \tag{1}$$

If the second part is offloaded to a server, sending the input data d_i takes $\frac{d_i}{B}$ seconds at bandwidth B. Here we ignore the initial setup time for the network. The program itself may also need to be sent to the server. We assume the size of the program is negligible, or the server may download the program from another site through a high-speed network [17]. Offloading can improve performance when execution, including computation and communication, can be performed faster at the server. Let s_s be the speed of the server. The time to offload and execute the second part is

$$\frac{d_i}{B} + \frac{w}{s_s}. \tag{2}$$

Offloading improves performance when Eq. 1 > Eq. 2:

$$\frac{w}{s_m} > \frac{d_i}{B} + \frac{w}{s_s} \Rightarrow w \times \left(\frac{1}{s_m} - \frac{1}{s_s}\right) > \frac{d_i}{B}. \tag{3}$$

This inequality holds for

- large w: the program requires heavy computation.
- large s_s: the server is fast.
- small d_i: a small amount of data is exchanged.
- large B: the bandwidth is high.

This inequality shows limited effects of the server's speed. If $\frac{w}{s_m} < \frac{d_i}{B}$, even if the server is infinitely fast (i.e., $s_s \to \infty$), offloading cannot improve performance. Hence, only tasks that require heavy computation (large w) with light data exchange (small d_i) should be considered. This requires analyzing programs to identify such tasks. Moreover, if we define $w(\frac{1}{s_m} - \frac{1}{s_s}) - \frac{d_i}{B}$ as the performance gain of offloading, the server's

speed has *diminishing return*: doubling s_s will not double the gain.

3.2 Save energy

Energy is a primary constraint for mobile systems. A survey of 7,000 users across 15 countries showed that "75% of respondents said better battery life is the main feature they want" [1, 2]. Smartphones are no longer used only for voice communication; instead, they are used for acquiring and watching videos, gaming, web surfing, and many other purposes. As a result, these systems will likely consume more power and shorten the battery life. Even though battery technology has been steadily improving, it has not been able to keep up with the rapid growth of power consumption of these mobile systems. Offloading may extend battery life by migrating the energy-intensive parts of the computation to servers [42].

The following analysis explains the conditions when offloading saves energy. Suppose p_m is the power on the mobile system. The energy to perform the task can be obtained by modifying Eq. 1:

$$p_m \times \frac{w}{s_m}. \tag{4}$$

Let p_c be the power required to send data from the mobile system over the network. After sending the data, the system needs to poll the network interface while waiting for the result of the offloaded computation. During this time, the power consumption is p_i. Incorporating these parameters in Eq. 2 gives

$$p_c \times \frac{d_i}{B} + p_i \times \frac{w}{s_s}. \tag{5}$$

Table 2 Offloading techniques for improving performance

Year	Paper	Decision	Contribution	M	V	Gr	Ga	T
2002	[49]	Static	Use a distributed platform to transparently offload portions of program to a service	✓		✓		✓
2003	[26]	Dynamic	Use fuzzy control to trigger adaptive offloading, and select an application's partitioning policy	✓				✓
2004	[30]	Static	Propose a computationally efficient prediction utility for mobile devices			✓		
	[15]	Dynamic	Partition an application into components and automatically select an appropriate adaptation strategy				✓	
	[74]	Dynamic	Find the optimal program partitioning corresponding to different ranges of run-time parameters	✓				
2006	[59]	Static	Propose an adaptive (k+1) partitioning algorithm that divides a given application into 1 unoffloadable part and k offloadable parts	✓				
	[71]	Dynamic	Discover servers by run-time monitoring based on the task requirements and device characteristics				✓	
	[72]	Static	Propose a partitioning system that uses bytecode rewriting to transform input Java applications into distributed applications on distinct Java Virtual Machines.			✓		
	[9]	Dynamic	Show that it is possible to quickly, easily, and effectively retarget computationally-intensive useful applications for cyber foraging	✓	✓	✓		✓
	[66]	Static	Introduce a Java bytecode transformer that replaces heavy methods by remote procedure calls to servers	✓				
2007	[60]	Dynamic	Propose an analytical model to express the performance of offloading systems in mobile wireless environments					✓
2008	[36]	Static	Offload based on the execution history of applications, and adaptive to the current conditions of the environment and device					✓
	[80]	Dynamic	Predict statistically when an offloaded computation will outperform execution on the mobile system	✓				
	[84]	Dynamic	Propose a service that can seamlessly offload some tasks of a mobile application		✓			
2010	[52]	Dynamic	Show how a robot can adaptively partition computation between itself and a server to improve performance		✓			

Different techniques may use static or dynamic decisions, and use a range of applications that can benefit from offloading. The abbreviated columns represent the following categories of applications—*M*: multimedia, *V*: vision and recognition, *Gr*: graphics, *Ga*: gaming, *T*: text processing

Offloading saves energy when Eq. 4 > Eq. 5.

$$p_m \times \frac{w}{s_m} > p_c \times \frac{d_i}{B} + p_i \times \frac{w}{s_s} \tag{6}$$

$$\Rightarrow w \times \left(\frac{p_m}{s_m} - \frac{p_i}{s_s}\right) > p_c \times \frac{d_i}{B} \tag{7}$$

Equations 3 and 7 are very similar. To make offloading save energy, heavy computation (large w) and light communication (small d_i) should be considered. In both equations, we assume that data must be transmitted from the mobile system to the server. Fortunately, this may not be true in many cases. For example, the data (such as photographs and videos) may also reside in servers with high-speed networks (such as Facebook.com and YouTube.com). Instead of transmitting the data from the mobile system to the server, the mobile system needs to provide *links* to the server and the server may download the data directly from the hosting sites. In this case, the bandwidth B can be substantially higher, allowing offloading to improve performance and save energy.

3.3 Comparison of existing studies

The previous sections describe the objectives for offloading: improving performance and saving energy. In this section, we classify existing studies based on the following criteria:

- When is the offloading decision made? Is it made statically during program development or dynamically during execution?
- How are tasks identified for offloading?
- What applications are offloaded?
- What types of mobile systems benefit from offloading?

Tables 2 and 3 classify different papers based on these criteria. The papers are ordered by years so that readers can see the progression more easily.

3.3.1 Static or dynamic decisions

The offloading decision can be static or dynamic. When the decision is static, the program is partitioned during development. Static partition has the advantage of

Table 3 Offloading techniques for saving energy

Year	Paper	Decision	Contribution	M	V	Gr	Ga	T
2001	[44]	Static	Divide the program into server tasks and client tasks such that the energy consumed at the client is minimized	✓				
2002	[45]	Static	Present a task partition and allocation scheme to divide the distributed multimedia processing between the server and a handheld device	✓				
	[46]	Static	Add security mechanism to offloading and show that despite the overhead of the security mechanism, offloading remains quite effective	✓				
2003	[41]	Dynamic	Examine potential benefits of application specific power management through remote task execution		✓			✓
	[67]	Static	Construct a stochastic model of the client-server system based on Markovian decision processes and formulate power management problem with task migration as an optimization problem					
2004	[13]	Dynamic	Allow offloading of both method execution and bytecode-to-native code compilation when executing a Java application; adaptively decide where to compile and execute a method (locally or remotely), and how to execute it (interpretation or just-in-time compilation)	✓				
2006	[56]	Static	Decide where to deploy software components across the distributed computing resources of autonomous robotic systems; decide how the different systems involved should communicate to best meet overall objectives		✓			
2007	[83]	Static	Execute the program initially on the mobile system with a timeout; if the computation is not completed after the timeout, it is offloaded.		✓			
2008	[70]	Dynamic	Propose a set of mechanisms and policies for running mobile applications across multiple, cooperating machines while actively performing power management to extend mobile system usability lifetimes		✓			
2009	[35]	Dynamic	Analyze conditions when offloading image retrieval is energy efficient	✓				
2010	[17]	Dynamic	Present system with fine grained energy-aware code offload capability	✓	✓		✓	

The techniques may use Static or Dynamic decisions for offloading, and use a wide range of applications that benefit from offloading. The abbreviated columns represent the following categories of applications—*M*: multimedia, *V*: vision and recognition, *Gr*: graphics, *Ga*: gaming, *T*: text processing

low overhead during execution; however, this approach is valid only when the parameters can be accurately predicted in advance. In Eqs. 3 and 7, s_m, p_c, p_i, and p_m can usually be accurately estimated. The server's speed s_s may vary but some cloud vendors can guarantee the minimum level of performance. The other parameters: w, d_i, and B may vary widely due to run-time conditions. A static scheme may predict some of these parameters and decide how the application is offloaded. Prediction algorithms include probabilistic prediction [67], history-based prediction [30, 36], and fuzzy control [26].

In contrast, dynamic decisions can adapt to different run-time conditions, such as fluctuating network bandwidths. Dynamic approaches may also use prediction mechanisms for decision making. For example, the offloading bandwidth B can be monitored and predicted using a Bayesian scheme [80]. Meanwhile, most dynamic decisions incur higher overhead because the program has to monitor the run-time conditions. Even for programs with dynamic decisions, the tasks that may potentially be offloaded are identified during program development. Partitioning a program during execution is undesirable due to the very high overhead for analyzing the program. Figure 2a shows static and dynamic decisions in the papers surveyed.

3.3.2 Program partition

Before making the offloading decision, the offloadable parts of a program have to be identified. This is usually achieved by *partitioning* a program. Various algorithms are used [13, 15, 35, 44–46, 52, 60, 72, 74, 83] to partition the computation between a mobile system and a server. A typical approach represents the program as a graph: the vertices represent the computational components (such as functions) and the edges represent the communication between them [59]. Figure 3 shows an example of dividing a program using graph partition. The program takes input x, y, z and gives output r. In the figure, the computation consists of four functions A, B, C, D. The objective is to decide which of these functions to offload. Each of these functions is a possible candidate for offloading; however it is difficult to independently apply the offloading analysis from Eqs. 3 and 7 for each function. We have to consider the intermediate data that is sent between the functions, given by x_1, y_1, z_1, x_2, and z_2. For example, A generates x_1, y_1, z_1; it sends x_1, y_1 to B and it sends y_1, z_1 to C. Let us assume that the data x_1, y_1, and z_2 are large. As a result, we want to keep the functions exchanging the data on the same system, so that the communication between them is a local function call and no network

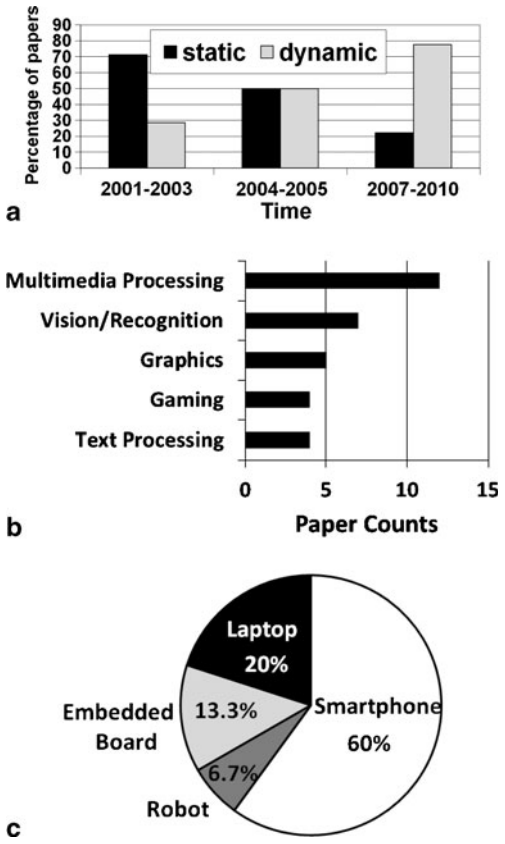

Fig. 2 **a** Types of algorithms used for offloading—in recent years, there are fewer static (i.e. development time) decisions and more dynamic (i.e. execution time) decisions. **b** Most frequently used types of applications for offloading. **c** Percentage breakdown of different types of devices used by the applications

communication is needed. Some of the functions may require heavy computation, and the energy to perform computation on the mobile system must also be considered. The figure shows a possible partition between a mobile system and a server. The optimal decision

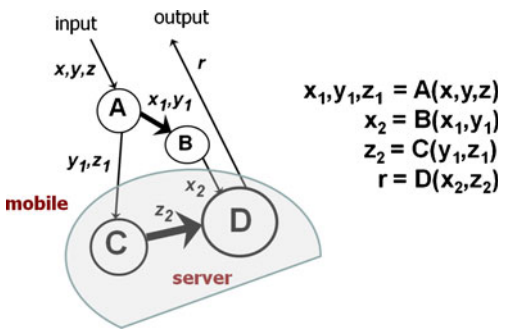

Fig. 3 Expressing a program as a graph and partitioning the graph between a mobile system and a server: x, y, z are inputs to the program and r is the output. The region marked "mobile" is executed on the mobile system, and the region marked "server" is executed on the server. The size of a vertex indicates the amount of computation and the width of an edge indicates the amount of data sent from one vertex to the other

depends on the relative tradeoffs between computation and communication; this decision is made by considering all the vertices simultaneously and is similar to a graph partition problem, known to be NP-Complete, even if all the parameters are known in advance [33]. In other scenarios where the program information is unknown [83], the application may not be partitioned—and the entire application is either executed on the mobile system, or offloaded.

3.3.3 Applications

Applications used for evaluation include text editors [26, 36, 49], multimedia [13, 35, 44–46, 49, 59, 66, 74], vision and recognition [9, 41, 52, 56, 83, 84], and gaming [15, 71]. Text editors transfer relatively small amounts of data—i.e., text, hence small d_i in Eqs. 3 and 7 and perform computations like spell check. Multimedia, vision, and recognition applications transfer large amounts of data in the form of images and videos (large d_i). If all the multimedia is on the mobile system, the offloading decision depends primarily on B: if most of the data is already on a server (such as Youtube), then offloading can be more beneficial. Gaming applications like chess are interesting candidates for offloading, because the amount of computation for the program to win the game depends on the skill level of the user; thus the computation w depends on how the user plays the game. Figure 2a shows the different applications used by various papers. The applications run on a wide range of clients with different computational capabilities, ranging from PDAs, laptops, and robots, as shown in Fig. 2c.

Table 4 Frameworks to alleviate the challenges faced by offloading

Paper	Year	Contributions	(a)	(b)	(c)	(d)
[38]	1995	Provide a set of tools to isolate mobile applications from the limitations of mobile communications using relocatable dynamic objects and queued remote procedure calls	✓	✓		
[53]	1995	Present API for application-aware adaptation with the ability to retrieve and present data at varying degrees of fidelity	✓	✓		
[39]	2000	Provide support for cooperative mobile ambient awareness in heterogeneous wireless and mobile infrastructure	✓			✓
[43]	2000	Propose a load-balancing scheme to minimize the call blocking probability due to lack of computing resources while offloading	✓	✓		
[4]	2002	Propose a uniform means for allowing users to contract context-based services with service providers	✓			
[64]	2004	Present adaptation mechanisms for applications that improve user support while taking resource availability into account	✓			✓
[77]	2004	Propose tool for developer to design the offloading aspects of the application by specifying an offloading layout that is enforced by the runtime during deployment	✓			
[65]	2004	Present adaptation mechanisms driven by resource information and resource contracts that are negotiated between the middleware and the application components	✓			
[25]	2004	Propose a surrogate infrastructure based on virtual machine technology that allows resource-constrained devices to utilize a surrogate's compute, network, and storage resources				✓
[27]	2005	Present a system infrastructure that allows local mobile devices to interact with a grid network.				✓
[24]	2005	Propose middleware-based programmable infrastructure that allows the nodes to download and activate required protocol and service software dynamically	✓			
[28]	2006	Present a service approach that extends pervasive devices with semantic Grid services by using a device-proxy-Grid system infrastructure	✓			
[37]	2006	Propose an automated refactoring process that makes application components should be amenable to partitioning	✓			✓
[19]	2007	Present a data outsourcing access control architecture for preserving privacy			✓	
[58]	2007	Propose mechanism to find a surrogate and the path to the surrogate to achieve a minimum surrogate execution time, given a service's remote execution components	✓			
[18]	2008	Address the privacy issue in the data outsourcing with combined use of access control and cryptography			✓	
[50]	2009	Reduce data leakage when sensitive information is sent to the cloud			✓	
[21, 22]	2009	Construct a fully homomorphic encryption scheme for any operations			✓	
[48]	2010	Use homomorphic encryption to protect images when comparing them on servers			✓	

The abbreviated columns represent the following in Section 3.4: *a* inter-operability, *b* mobility and fault tolerance, *c* privacy and security, and *d* context awareness

3.4 Infrastructures

The previous sections describe the conditions when offloading computation can improve performance, or save energy, or both. Many papers have contributed to the infrastructures to make offloading practically adopted. These infrastructures address various issues such as:

(a) Inter-operability: Different types of resource-constrained devices may interact and connect across different types of networks to one or many servers. For example, devices like the iPhone switch to 3G signal when there is no WiFi network available. Since the 3G radio is typically slower and consumes more power than WiFi, the offloading decision may vary based on the network available. Moreover, offloading may be possible between different systems of different computational capabilities; it is important to hide these interactions from the user [4, 16, 38, 39, 43, 53, 64, 65, 77].

(b) Mobility and Fault Tolerance: Offloading relies on wireless networks and servers; thus it is important to handle failures and to focus on reliable services. Fault tolerance enables the system to continue executing the application in the event of network congestion or failure, or server failure. Studies that have addressed this issue include [38, 43, 53, 57].

(c) Privacy and Security: Privacy is a concern because users' programs and data are sent to servers that are not under the users' control. Security is an issue because a third party may access confidential data. Many studies have been conducted to protect outsourced data [14, 75, 76]. Solutions include steganography [47], homomorphic encryption [21, 22], hardware-based secure execution [7]. Most of these solutions have limitations in their applications: for example, encryption keys may be too large and dramatically increase the amount of data. Also, efficient computation on encrypted data is still a research topic.

(d) Context Awareness: This refers to the device being able to perceive the users state and surroundings and infer context information. This is important because the mechanism of offloading may vary depending on the users' location and context; various studies suggest adaptive mechanisms based on such information [25, 27, 37, 39, 64].

All these issues continue to be active areas of research. Table 4 describes some infrastructures and contributions that alleviate these issues.

4 Offloading in the future

How important will offloading be in the years to come? In this section, we discuss how growth in mobile data and mobile applications, and the computational capabilities of mobile devices will impact offloading in the future.

In the the past few years, two important trends have occurred:

- *sensor deployment*: Sensors are widely deployed for monitoring the environment or for security. These sensors acquire large amounts of data but the sensors have limited computing capabilities.
- *growth in smartphones:* smartphones have become the primary computing platforms for millions of people. Figure 4 shows that the volume of mobile data is forecasted to grow rapidly. Mobile platforms generate large amounts of multimedia data and most of the data are stored on-line on cloud servers.

Sensors and mobile platforms represent an entirely new set of input devices. Consider the possibility when millions of cameras, microphones, GPS, and many other types of sensors are connected. The amounts of data they can produce would be staggering. The information and knowledge that can be extracted would dwarf what we have called *information explosion* today. As the number of connected devices—including mobile phones, tablets, laptops, and sensors—grow, the demand for increased functionalities will continue. In the next few years, we will see pressing needs for personalized management of multimedia data. This would be a natural progression of the Internet. Before the Internet became popular, people already had large amounts of on-line documents stored in their desktop computers or company mainframes. In 1990s, as the Internet became popular, many documents were posted on-line and keyword-based search became necessary. Search engines were an important driving force for the Internet in the late 1990s. The first ten years of

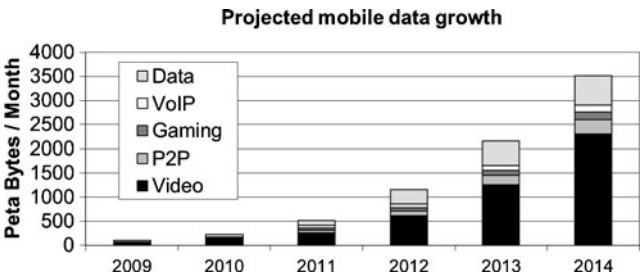

Fig. 4 Mobile data growth projected in the years to come. It is observed that there is a 6 fold increase in mobile data, particularly in multimedia such as videos. Source: CISCO 2010

the 21st century marked the rapid growth of personal multimedia data: images, videos, and audios. As the amounts of multimedia data grow, users need better ways to manage their data than relying on file names, dates, and directories. It would be inconvenient to ask users to describe every image and every video by a set of keywords and then use keyword-based search. This will lead to a rapid growth in recognition and data management technologies on these connected devices. Many of these technologies can provide large speedups with parallelism, since multimedia processing offers many opportunities for both code and data parallelism.

Computing speeds of these connected devices however, will not grow at the same pace as servers' performance. This is due to several constraints, including

- *form factor*: users want devices that are smaller and thinner; yet they also want devices with more computational capability.
- *power consumption*: current battery technology constrains the clock speed of processors since doubling the clock speed approximately octuples the power consumption. It becomes difficult to offer long battery lifetimes with high clock speeds.

These factors indicate that mobile computing speeds will not grow as fast as the growth in data, and applications' computational requirements. Where do these trends intersect? On one hand, we have a massive growth in mobile data in both types and volumes, and in the computational requirements of mobile applications. On the other hand, the computational capabilities of the devices—that acquire and store the data, and provide applications for the user—will be unlikely to grow at the same pace. Offloading computation is a natural solution to this problem.

The economic model for offloading, by renting computation, is provided by virtualization and cloud computing. As the various connected devices become more widespread in their deployment, offloading techniques that can take advantage of cloud computing will become increasingly relevant. Applications on these connected devices will start to be designed such that they have "offloadable" computation—and such design of applications can benefit from the various techniques and solutions surveyed in this paper.

5 Conclusion

This paper surveys and classifies a vast body of research associated with computation offloading for mobile systems. We examine how enablers like mobile agents and virtualization make offloading feasible. We survey different types of algorithms used to partition and offload programs in order to improve performance or save energy. We classify the types of applications that have been used to demonstrate offloading. We list some of the research areas associated with offloading, and describe some infrastructures and solutions that address these research areas. Finally we describe why computation offloading will become increasingly important for resource constrained devices in the future.

References

1. Battery Life Concerns Mobile Users (2005). http://www.cnn.com. Accessed 23 Sept 2005
2. Mobile Phone Users Demand Decent Batteries (2005). http://www.eetimes.com. Accessed 22 Sept 2005
3. Adams K, Agesen O (2006) A comparison of software and hardware techniques for x86 virtualization. In: International conference on architectural support for programming languages and operating systems, pp 2–13
4. Anagnostou ME, Juhola A, Sykas ED (2002) Context Aware services as a step to pervasive computing. In: Lobster workshop on location based services for accelerating the European-wide deployment of services for the mobile user and worker, pp 4–5
5. Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D (2002) SETI@ home: an experiment in public-resource computing. Commun ACM 45(11):56–61
6. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) A view of cloud computing. Commun ACM 53(4): 50–58
7. Bajikar S (2002) Trusted platform module (tpm) based security on notebook pcs-white paper. White Paper, Mobile Platforms Group–Intel Corporation, 20
8. Balan RK (2004) Powerful change part 2: reducing the power demands of mobile devices. IEEE Pervasive Comput 3(2):71–73
9. Balan RK (2006) Simplifying cyber foraging. PhD thesis, School of Computer Science, Carnegie Mellon University
10. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. In: ACM symposium on operating systems principles, pp 164–177
11. Buyya R, Yeo CS, Venugopal S (2008) Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities. In: IEEE conference on high performance computing and communications, pp 5–13
12. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gen Comput Syst 25(6):599–616
13. Chen G, Kang B-T, Kandemir M, Vijaykrishnan N, Irwin MJ, Chandramouli R (2004) Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. IEEE Trans Parallel Distrib Syst 15(9): 795–809
14. Chow R, Golle P, Jakobsson M, Shi E, Staddon J, Masuoka R, Molina J (2009) Controlling data in the cloud: outsourcing computation without outsourcing control. In: ACM workshop on cloud computing security, pp 85–90

15. Chu H, Song H, Wong C, Kurakake S, Katagiri M (2004) Roam, a seamless application framework. J Syst Softw 69(3):209–226
16. Chun BG, Maniatis P (2009) Augmented smartphone applications through clone cloud execution. In: Conference on hot topics in operating systems, USENIX Association, pp 8–12
17. Cuervo E, Balasubramanian A, Cho D, Wolman A, Saroiu S, Chandra R, Bahl P (2010) MAUI: making smartphones last longer with code offload. In: International conference on mobile systems, applications, and services, pp 49–62
18. di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Pelosi G, Samarati P (2008) Preserving confidentiality of security policies in data outsourcing. In: ACM workshop on privacy in the electronic society, pp 75–84
19. di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Samarati P (2007) A data outsourcing architecture combining cryptography and access control. In: ACM workshop on computer security architecture, pp 63–69
20. Forman GH, Zahorjan J (1994) The challenges of mobile computing. Computer 27(4):38–47
21. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: ACM symposium on theory of computing, pp 169–178
22. Craig Gentry (2010) Computing arbitrary functions of encrypted data. Commun ACM 53(3):97–105
23. Goldberg RP (1974) Survey of virtual machine research. IEEE Comput 7(6):34–45
24. Gouveris S, Sivavakeesar S, Pavlou G, Malatras A (2005) Programmable middleware for the dynamic deployment of services and protocols in ad hoc networks. In: International symposium on integrated network management, pp 3–16
25. Goyal S, Carter J (2004) A lightweight secure cyber foraging infrastructure for resource-constrained devices. In: Mobile computing systems and applications, pp 184–195
26. Gu X, Nahrstedt K, Messer A, Greenberg I, Milojicic D (2003) Adaptive offloading inference for delivering applications in pervasive computing environments. In: IEEE international conference on pervasive computing and communications, pp 107–114
27. Guan T, Zaluska E, Roure D (2005) A grid service infrastructure for mobile devices. In: IEEE international conference on semantics, knowledge, and grid, pp 42–48
28. Guan T, Zaluska E, Roure D (2006) Extending pervasive devices with the semantic grid: a service infrastructure approach. In: IEEE conference on computer and information technology pp 113–118
29. Gurun S, Krintz C (2003) Addressing the energy crisis in mobile computing with developing power aware software. Technical Report, Department of Computer Science, University of California, Santa Barbara
30. Gurun S, Krintz C, Wolski R (2004) NWSLite: a lightweight prediction utility for mobile devices. In: International conference on mobile systems, applications, and services, pp 2–11
31. Hayes B (2008) Cloud computing. Commun ACM 51(7):9–11
32. Hemmes J, Poellabauer C, Thain D (2007) On-demand transient data storage and backup in mobile systems. In: IEEE military communications conference, pp 1–7
33. Hendrickson B, Leland R (1995) A multilevel algorithm for partitioning graphs. In: ACM/IEEE conference on supercomputing, pp 28–41
34. Hong JI, Landay JA (2011) An infrastructure approach to context-aware computing. Int J Hum-Comput Int 16(2): 287–303
35. Hong YJ, Kumar K, Lu YH (2009) Energy efficient content-based image retrieval for mobile systems. In: International symposium on circuits and systems, pp 1673–1676
36. Huerta-Canepa G, Lee D (2008) An adaptable application offloading scheme based on application behavior. In: International conference on advanced information networking and applications - workshops, pp 387–392
37. Jamwal V, Iyer S (2006) Automated refactoring of objects for application partitioning. In: Asia-Pacific software engineering conference, pp 671–678
38. Joseph AD, de Lespinasse AF, Tauber JA, Gifford DK, Kaashoek MF (1995) Rover: a toolkit for mobile information access. In: ACM symposium on operating systems principles, pp 156–171
39. Kanter T (2000) Cooperative mobile ambient awareness. Licentiate Thesis, Department of Teleinformatics, Royal Institute of Technology (KTH)
40. Kotz D, Gray R, Nog S, Rus D, Chawla S, Cybenko G (1997) Agent Tcl: targeting the needs of mobile computers. IEEE Internet Comput 1(4):58–67
41. Kremer U, Hicks J, Rehg J (2003) A compilation framework for power and energy management on mobile computers. In: Proceedings of the 14th international conference on Languages and compilers for parallel computing. Cumberland Falls, KY, USA, pp 115–131
42. Kumar K, Lu YH (2010) Cloud computing for mobile users: can offloading computation save energy? IEEE Comput 43(4):51–56
43. Lele AM, Nandy SK, Epema DHJ (2000) Harmony—an architecture for providing quality of service in mobile computing environments. J Interconnect Netw 1(3): 247–266
44. Li Z, Wang C, Xu R (2001) Computation offloading to save energy on handheld devices: a partition scheme. In: International conference on compilers, architecture, and synthesis for embedded systems, pp 238–246
45. Li Z, Wang C, Xu R (2002) Task allocation for distributed multimedia processing on wirelessly networked handheld devices. In: Parallel and distributed processing symposium, pp 79–84
46. Li Z, Xu R (2002) Energy impact of secure computation on a handheld device. In: IEEE international workshop on workload characterization, pp 109–117
47. Liu J, Kumar K, Lu YH (2010) Tradeoff between energy savings and privacy protection in computation offloading. In: ACM/IEEE international symposium on low power electronics and design, pp 213–218
48. Liu J, Lu Y-H (2010) Energy savings in privacy-preserving computation offloading with protection by homomorphic encryption. In: HotPower
49. Messer A, Greenberg I, Bernadat P, Milojicic D, Chen D, Giuli T, Gu X (2002) Towards a distributed platform for resource-constrained devices. In: International conference on distributed computing systems, vol 22, pp 43–51
50. Mowbray M, Pearson S (2009) A client-based privacy manager for cloud computing. In: International ICST conference on communication system software and middleware, pp 1–8
51. Neiger G, Santoni A, Leung F, Rodgers D, Uhlig R (2006) Intel virtualization technology: hardware support for efficient processor virtualization. Intel Technol J 10(3): 167–177
52. Nimmagadda Y, Kumar K, Lu Y-H, Lee CSG (2010) Real-time moving object recognition and tracking using computation offloading. In: IEEE international conference on intelligent robots and systems, pp 2449–2455
53. Noble BD, Price M, Mahadev Satyanarayanan M (1995) A programming interface for application-aware adaptation in

mobile computing. School of Computer Science, Carnegie Melon University
54. Noble BD, Satyanarayanan M (1999) Experience with adaptive mobile applications in Odyssey. Mobile Netw Appl 4(4):245–254
55. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2009) The eucalyptus open-source cloud-computing system. In: IEEE/ACM international symposium on cluster computing and the grid, pp 124–131
56. O'Hara KJ, Nathuji R, Raj H, Schwan K, Balch T (2006) Autopower: toward energy-aware software systems for distributed mobile robots. In: IEEE international conference on robotics and automation, pp 2757–2762
57. Ou S, Wu Y, Yang K, Zhou B (2008) Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments. In: IEEE international conference on communications, pp 1856–1860
58. Ou S, Yang K, Hu L (2007) Cross: a combined routing and surrogate selection algorithm for pervasive service offloading in mobile ad hoc environments. In: IEEE global telecommunications conference, pp 720–725
59. Ou S, Yang K, Liotta A (2006) An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In: IEEE international conference on pervasive computing and communications, pp 116–125
60. Ou S, Yang K, Liotta A, Hu L (2007) Performance analysis of offloading systems in mobile wireless environments. In: IEEE international conference on communications, pp 1821–1806
61. Perkins CE (1996) Handling multimedia data for mobile computers. In: Computer software and applications conference, pp 147–148
62. Powell ML, Miller BP (1983) Process migration in demos/mp. ACM SIGOPS Oper Syst Rev 17(5):110–119
63. Qi M (1997) Resource conservation in a mobile transaction system. IEEE T Comput 46:3:299–311
64. Rigole P, Berbers Y, Holvoet T (2004) Component-based adaptive tasks guided by resource contracts. In: Workshop on component-oriented approaches to context-aware systems, pp 1–5
65. Rigole P, Berbers Y, Holvoet T (2004) Mobile adaptive tasks guided by resource contracts. In: Workshop on middleware for pervasive and ad-hoc computing, pp 117–120
66. Rim H, Kim S, Kim Y, Han H (2006) Transparent method offloading for slim execution. In: International symposium on wireless pervasive computing, pp 1–6
67. Rong P, Pedram M (2003) Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach. In: Conference on design automation, pp 906–911
68. Rosenblum M, Garfinkel T (2005) Virtual machine monitors: current technology and future trends. IEEE Comput 38(5):39–47
69. Se S, Barfoot T, Jasiobedzki P (2005) Visual motion estimation and terrain modeling for planetary rovers. In: International symposium on artificial intelligence for robotics and automation in space, pp 603–700
70. Seshasayee B, Nathuji R, Schwan K (2007) Energy aware mobile service overlays: cooperative dynamic power management in distributive systems. In: International conference on automatic computing, pp 6–12
71. Sivavakeesar S, Gonzalez OF, Pavlou G (2006) Service discovery strategies in ubiquitous communication environments. IEEE Commun Mag 44(9):106–113
72. Tilevich E, Smaragdakis Y (2006) J-orchestra: automatic Java application partitioning. In: European conference on object-oriented programming, pp 1–3
73. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2008) A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 39(1):50–55
74. Wang C, Li Z (2004) Parametric analysis for adaptive computation offloading. In: ACM SIGPLAN conference on programming language design and implementation, pp 119–130
75. Wang Q, Wang C, Li J, Ren K, Lou W (2010) Enabling public verifiability and data dynamics for storage security in cloud computing. In: European symposium on research in computer security, pp 355–370
76. Wang W, Li Z, Owens R, Bhargava B (2009) Secure and efficient access to outsourced data. In: ACM workshop on cloud computing security, pp 55–66
77. Weinsberg Y, Dolev D, Wyckoff P, Anker T (2007) Accelerating distributed computing applications using a network offloading framework. In: IEEE international parallel and distributed processing symposium, pp 1–10
78. Weiss A (2007) Computing in the clouds. NetWorker 11(4):16–25
79. White JE (1997) Mobile agents. In: Software agents (MIT Press), pp 437–472
80. Wolski R, Gurun S, Krintz C, Nurmi D (2008) Using bandwidth data to make computation offloading decisions. In: IEEE international symposium on parallel and distributed processing, pp 1–8
81. Wong D, Paciorek N, Moore D (1999) Java-based mobile agents. Commun ACM 42(3):92–102
82. Wong D, Paciorek N, Walsh T, DiCelie J, Young M, Peet B (1997) Concordia: an infrastructure for collaborating mobile agents. In: International workshop on mobile agents, pp 86–97
83. Xian C, Lu Y-H, Li Z (2007) Adaptive computation offloading for energy conservation on battery-powered systems. In: International conference on parallel and distributed systems, pp 1–8
84. Yang K, Ou S, Chen H-H (2008) On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications. IEEE communications magazine 46(1):56–63