

Computation Offloading Decisions for Reducing Completion Time

Salvador Melendez and Michael P. McGarry
Department of Electrical and Computer Engineering
University of Texas at El Paso
El Paso, TX 79968

Abstract—We analyze the conditions in which offloading computation reduces completion time. We extend the existing literature by deriving an inequality (Eq. 4) that relates computation offloading system parameters to the bits per instruction ratio of a computational job. This ratio is the inverse of the arithmetic intensity. We then discuss how this inequality can be used to determine the computations that can benefit from offloading as well as the computation offloading systems required to make offloading beneficial for particular computations.

Index Terms—Cloud computing, mobile cloud computing, communication, networks, cloud resource provisioning.

I. INTRODUCTION

Computing as a utility offers computation as a service over a communication network in much the same way that the electrical utility offers electric power as a service over a power distribution network. Several advances have occurred in recent years to make computing as a utility commercially viable. These include advances in computing resource virtualization/isolation, advances in high-bandwidth/low-latency communication, and increasing economies of scale for large-scale computing facilities. Cloud computing [1], [2] is a recent buzz word that encompasses computing as a utility. Many computing scenarios benefit from computing as a utility, such as those where the computing demand is elastic or unpredictable. However, one of the most compelling uses of computing as a utility is to enhance the capabilities of edge computing devices [3], [4] such as smart phones, tablets, wearable computers, smart objects (e.g., smart appliances and furniture), and cyber-physical systems (CPS). Edge computing devices can be empowered by computing as a utility, through remote execution, to execute real-time applications that would not be possible on the edge device itself in the desired time-frame. The act of remote execution on a so called “cloud resource” is often referred to as **computation offloading** [5] or cyber-foraging [6]–[8].

The computing devices that provide the computation offloading service to edge computing devices can exist not only in a remote data center but in locations much closer to the edge. These locations may be within the same room, at an Internet access point, or inside an Internet Service Provider (ISP) point-of-presence (PoP). These various locations provide a tiered computing structure, see Figure 1, whereby cloud computing resources can be found at varying distances from the edge computing devices.

Recent research activities have produced a variety of programming frameworks and systems that make computation offloading possible. Spectra [9] and Chroma [10] were seminal computation offloading systems that were followed up by several others (e.g., Slingshot [11], MAUI [4], Cuckoo [12], CloneCloud [13]). Clearly, mechanisms to facilitate computation offloading have received significant attention from the research community. A recent survey on computation offloading [14] identified two objectives for computation offloading: (1) reduce execution time, and (2) shift energy consumption. The survey also identified two classes of decision to be made: (1) *what* computation to offload, and (2) *where* to offload computation.

The decision regarding *what* to offload is generally referred to as the partitioning problem and many techniques have been proposed and evaluated [15]–[31]. Applications are partitioned into components and a binary decision is made whether to offload a component or not. The data exchanged between the application components is considered when making the decision to offload each component.

Much of the literature related to deciding *where* to offload focuses on the binary decision of whether to offload or not, which is similar to the decision of *what* to offload. Some of the analysis developed is static and provides rules of thumb regarding the conditions in which offloading computation is favorable [32], [33]. The general consensus is that it is beneficial to offload computation if there are large amounts of computation and only a small amount of data that needs to be transmitted over a communication network. Most of the systems mentioned above that facilitate computation offloading (e.g., MAUI, Cuckoo) indeed included a computation offloading decision algorithm that decided whether to offload a computation or not (i.e., the binary decision). For example, MAUI [4] contains a binary decision algorithm that is the solution of a binary integer linear mathematical program with an objective function to minimize energy consumption subject to a particular completion time constraint. The decision algorithm utilizes historical energy consumption and network throughput data and executes on the offload target to avoid burdening the mobile device.

In this paper we extend the literature that analyzes when offloading reduces completion time. We derive an inequality that compares offloading system parameters to computational job parameters to determine when offloading would reduce

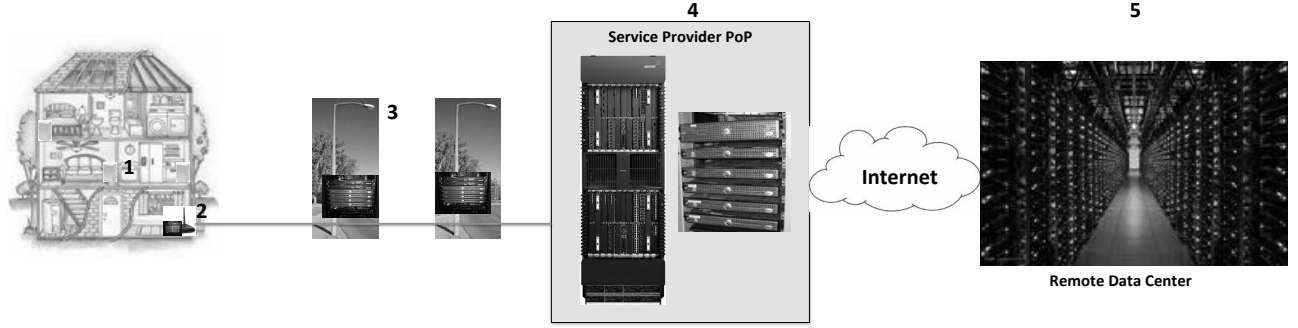


Fig. 1. A five tier computation offloading system: (1) “plug computers” in certain rooms of a home (dual-core ARM systems) plugged into power outlets, (2) server access points that are a combined wireless access point and multiprocessor/multi-core computer, (3) small racks of computers attached to light posts in the community, (4) large racks of computers alongside racks of telecommunications switching equipment in telecom points-of-presence, (5) remote data center (e.g., Amazon Web Services).

completion time. The computational job parameters we use form a ratio that is related to the arithmetic intensity of a computation [34].

A. Overview

In Section II we present the system model that we use in our subsequent analysis. In Section III we present our analysis of the conditions in which computation offloading can reduce execution time. Finally, in Section IV we conclude with a summary of our findings.

II. SYSTEM MODEL

The system under study consists of a client device that produces computational jobs that can either be executed locally or offloaded for execution on one of several cloud resources. The cloud resources are distributed spatially in a network at varying numbers of hops from the client device. If the job is executed locally, the completion time is the time to execute all of the instructions of the job at the execution rate of the local computing device (i.e., computation time). If the job is offloaded, the completion time is the time to execute all of the instructions of the job at the execution rate of the selected cloud resource (i.e., computation time) plus the time to transmit the input and output data through the network (i.e., communication time). In our analysis, we consider the memory access times to be similar between the local computing device and the cloud resources. This is a reasonable assumption given the well documented processor-memory performance gap [35]. We now present our models for computation time and communication time that compose our model of completion time.

A. Computation time

Let C be the size of the computational job (instructions), e be the execution rate of the local computing device (instructions/sec), and E be the execution rate of that particular compute resource (instructions/sec). The computation time for the computational job executed locally is $\xi = \frac{C}{e}$ and $\xi = \frac{C}{E}$ if it is executed on a particular cloud resource.

B. Communication time

We use a per-hop model of communication time that incorporates the generally accepted categories of delay incurred at each hop [36]: (i) processing, (ii) queueing, (iii) transmission, and (iv) propagation delays. Let S be the size of the packet (bits), α be the processing delay at a hop in a network (sec), β be the queueing delay incurred at that hop (sec), γ be the rate of the transmission channel at that hop (bits/sec), l be the length of that hop (meters), and c be the speed of light (meters/sec). Then, the time required to transmit a packet across one hop in the network is $\psi = \alpha + \beta + \frac{S}{\gamma} + \frac{l}{c}$.

Let h be the number of hops along the path that a packet traverses from its source to its destination. Ignoring the negligible processing and propagation delays, we transform the equation above to characterize the end-to-end communication time as $\psi = \sum_{j=1}^h \left(\beta(j) + \frac{S}{\gamma(j)} \right)$.

Now, consider the transmission of a file that is larger than the maximum allowable packet size. In this case, the file is transformed into a packet train: several maximum-size packets followed by a remainder packet that can be up to the maximum size allowed. As individual packets in the packet train are transmitted through several hops in the network, packet transmission can occur in parallel. To model the communication time considering the effect of parallel packet transmission, we consider: (1) the time to transmit the entire file through the bottleneck transmission channel, and (2) the time to transmit the last packet in the packet train through each hop. Let F be the size of the file (bits) and N be the size of the last packet in the packet train. Then the communication time for the entire file is

$$\psi = \frac{F}{\min\{\gamma(j), \forall j\}} + \sum_{j=1}^h \left(\beta(j) + \frac{N}{\gamma(j)} \right).$$

C. Composite completion time model

Let x be the completion time, I be the size of the input data (bits), and O be the size of the output data (bits). If the computational job is executed locally, the completion time is:

$$x = \frac{C}{e}.$$

If the computational job is executed on a particular cloud resource, the completion time is:

$$\frac{C}{E} + \frac{(I + O)}{\min\{\gamma(j), \forall_j\}} + \sum_{j=1}^h \left(\beta(j) + \frac{N}{\gamma(j)} \right).$$

D. Useful ratios

The following two ratios are useful for our computation offloading analysis.

The computing-to-communication ratio (CCR) is the ratio of the computation time to the communication time. Using the symbols above,

$$CCR = \frac{\xi}{\psi}.$$

The remote-to-local ratio (RLR) is the ratio of the cloud resource execution speed to the local execution speed. Again, using the symbols above,

$$RLR = \frac{E}{e}.$$

III. WHEN TO OFFLOAD COMPUTATION?

To favor remote execution (or computation offloading) for reducing completion time, the following inequality must hold true:

$$\frac{C}{e} > \left[\frac{C}{E} + \frac{(I + O)}{\min\{\gamma(j), \forall_j\}} + \sum_{j=1}^h \left(\beta(j) + \frac{N}{\gamma(j)} \right) \right]. \quad (1)$$

We now manipulate this inequality to derive useful insight into computation offloading system design. To ease manipulation of this inequality, we let H represent the hop-by-hop network delay that is agnostic to the job size or the data size, F be all of the data to be transferred over the network (input and output), and Γ represent the transmission rate of the bottleneck link in the network. We now isolate the RLR on the left hand side of the inequality.

$$\begin{aligned} \frac{C}{e} &> \left[\frac{C}{E} + \left(\frac{F}{\Gamma} + H \right) \right] \\ C \left(\frac{E}{e} - 1 \right) &> E \left(\frac{F}{\Gamma} + H \right) \end{aligned} \quad (2)$$

$$\begin{aligned} \frac{E}{e} &> \frac{\left(\frac{F}{\Gamma} + H \right)}{\frac{C}{E}} + 1; \frac{E}{e} > \frac{\psi}{\xi} + 1 \\ RLR &> \frac{1}{CCR} + 1 \end{aligned} \quad (3)$$

After isolating the RLR on the left hand side, we have the inequality shown in Eq. 3. That inequality shows there is a nearly inverse relationship between RLR and CCR. To visualize the implications of this inequality we tabulate

TABLE I
RLR VALUES REQUIRED FOR OFFLOADING TO BE FAVORABLE FOR SEVERAL CCR VALUES.

CCR	RLR
1e-6	$\approx 1e6$
1e-3	1001
0.01	101
0.1	11
1	2
1e3	1.001
1e6	1.000001

TABLE II
INSTRUCTIONS PER SECOND RATINGS FOR SEVERAL PROCESSORS.

Processor	IPS
MSP430	16×10^6 [37]
A9	3.6×10^9 [38]
Celeron	6.43×10^9 [39]
Core i3	36.8×10^9 [40]
Xeon	136.20×10^9 [39]

the RLR value required to make offloading favorable for various values of the CCR; see Table I. A CCR of 1×10^{-3} requires cloud resources to be more than a thousand times faster than the local computing device. A CCR of 1×10^{-6} requires the cloud resources to be a million times faster! To obtain a sense of practical RLR values, we have compiled the instructions per second (IPS) ratings of two embedded processors that represent the low end (Texas Instrument's MSP430) and the high end (Apple's A9) of the spectrum of the embedded processors deployed in handheld devices. To complete the RLR computation, we have compiled the IPS ratings of a laptop-class processor (Intel's Celeron), desktop-class processor (Intel's Core i3), and a server-class processor (Intel's Xeon). See Table II for these IPS ratings.

Using the IPS ratings shown in Table II we compute the RLR value for each pair of handheld and laptop/desktop/server class processor. These RLR values are shown in Table III. We see the RLR values range from 1.79 for an A9 processor offloading to a Celeron processor (CCR must be greater than 1.27 to offload) up to 8512.5 for an MSP 430 processor offloading to a Xeon processor (CCR must be greater than 1.1×10^{-4} to offload).

Equation 3 leads one to, at first glance, think that if we increase the RLR value by let's say increasing the execution speed of the remote resource (i.e., increase E) this will make computation offloading beneficial for more applications as characterized by their CCR value. However, if we look back at

TABLE III
CPU RLR VALUES

	Remote			
Local		Celeron	i3	Xeon
MSP430		401.875	2300	8512.5
A9		1.78611	10.222	37.833

the presentation of this inequality specifically in Eq. 2 we see that increasing E will increase both sides of the inequality. Therefore, increasing E does not make offloading favorable for more applications.

To gain clear insight into how system parameters should be manipulated to increase the number of applications that benefit from computation offloading we need an inequality with job parameters on one side of the inequality and computation offloading system parameters on the other side of the inequality.

Starting from the original simplified inequality in Eq. 2 we take steps to move the offloading system parameters to the left hand side.

$$C \left(\frac{1}{e} - \frac{1}{E} \right) > \left(\frac{F}{\Gamma} + H \right); C\Gamma \left(\frac{1}{e} - \frac{1}{E} \right) > (F + H\Gamma)$$

$$\Gamma \left(\frac{1}{e} - \frac{1}{E} \right) > \left(\frac{F}{C} + \frac{H\Gamma}{C} \right); \Gamma \left(\frac{1}{e} - \frac{1}{E} - \frac{H}{C} \right) > \frac{F}{C}$$

If we consider an uncongested network so that H is negligible,

$$\Gamma \left(\frac{1}{e} - \frac{1}{E} \right) > \frac{F}{C}. \quad (4)$$

The ratio of job parameters on the right hand side of this inequality is the inverse of the arithmetic intensity or the bits per instruction of the computational job. This context differs from the common usage of the term arithmetic intensity in that we are referring to bits that would be communicated over the network and not bits that would be read/written from/to the memory system. The left hand side represents the capacity of the offloading system to consume bits and instructions; at the bottleneck link rate Γ and the cloud resource execution rate E compared to the local execution rate e , respectively.

A. $\frac{F}{C}$ values from TACC workload data

We obtained computational workload data from the Texas Advanced Computing Center (TACC) in Austin, TX. The data was provided by the Director of High Performance Computing, Dr. Bill Barth. The trace contains records of 78,176 computation jobs of 1,159 different applications. It provides information such as: 1) job ID, 2) application name, 3) job size, 4) bytes written, 5) bytes read, and 6) execution time. We use this data along with some assumptions about the execution speed of TACC resources to derive the $\frac{F}{C}$ for recognizable applications. Table IV shows the minimum, average, and maximum of the derived values of each data point in the workload data set for recognizable applications.

B. Determining when offloading is beneficial

We can use the inequality of Eq. 4 along with the inverse arithmetic intensity values (i.e., $\frac{F}{C}$) to determine when offloading is beneficial. To assist in this comparison we compute the values of $\left(\frac{1}{e} - \frac{1}{E} \right)$ for various combinations of local and remote processors. See Table V for these values.

TABLE IV
BITS PER INSTRUCTION FOR SEVERAL TACC WORKLOAD APPLICATIONS.

App Name	min	ave	max
namd2	1.61x10 ⁻⁷	4.45x10 ⁻⁵	1.01x10 ⁻³
imp_stampede	1.36x10 ⁻⁷	6.60x10 ⁻⁶	9.53x10 ⁻⁴
vasp_ncl	1.43x10 ⁻⁷	6.56x10 ⁻⁶	2.86x10 ⁻⁴
cosmomc	2.34x10 ⁻⁷	5.61x10 ⁻⁶	4.84x10 ⁻⁴
charmm	1.42x10 ⁻⁷	9.32x10 ⁻⁶	7.34x10 ⁻⁵
nwchem	1.01x10 ⁻⁶	7.06x10 ⁻⁵	1.80x10 ⁻⁴
fvcom	3.36x10 ⁻⁶	2.17x10 ⁻⁴	2.27x10 ⁻³
mdrun_mpi	1.66x10 ⁻⁷	8.18x10 ⁻⁶	1.08x10 ⁻⁴
siesta	1.47x10 ⁻⁷	9.81x10 ⁻⁶	5.29x10 ⁻⁵

TABLE V
REMOTE TO LOCAL EXECUTION RATIO $\left(\frac{1}{e} - \frac{1}{E} \right)$

Remote Local	Celeron	i3	Xeon
MSP430	6.23x10 ⁻⁸	6.25x10 ⁻⁸	6.25x10 ⁻⁸
A9	1.22x10 ⁻¹⁰	2.51x10 ⁻¹⁰	2.70x10 ⁻¹⁰

Using this data we can see that if we are offloading from an MSP430 processor to an Intel Celeron processor through a network with a 1 Kbps bottleneck link rate, applications with inverse arithmetic intensities less than 6.23x10⁻⁵ will benefit from offloading with respect to a reduction in completion time. Looking at Table IV we see that a large number of the scientific applications from the TACC workload data will benefit from offloading in this scenario. If we increase the bottleneck link rate to 1 Mbps, then inverse arithmetic intensities less than 6.23x10⁻² benefit from offloading; now all of the scientific applications listed in Table IV benefit from offloading.

If we offload from Apple's A9 to an Intel Celeron through a network with a 1 Kbps bottleneck link rate then applications with inverse arithmetic intensities less than 1.22x10⁻⁷ benefit from offloading. If the bottleneck link rate is increased to 1 Mbps, then inverse arithmetic intensities less than 1.22x10⁻⁴ benefit from offloading.

This style of analysis using our inequality from Eq. 4 can be utilized to determine which computations can benefit from a particular offloading system or what offloading system is required to make offloading beneficial for a particular computation.

IV. CONCLUSION

We have derived an inequality that relates computation offloading system parameters to the arithmetic intensity of a computation (see Eq. 4). This inequality can be used to determine which computations benefit from computation offloading w.r.t. reduction in completion time for a particular computation offloading system. This inequality can also be used to determine the computation offloading system required to permit certain computations to benefit from offloading. See Section III-B for some examples of this type of analysis using our inequality.

Future work should tabulate the arithmetic intensities for various computations and use these values to identify which computations can benefit from practical computation offloading systems.

ACKNOWLEDGMENT

This research was supported by the U.S. Army Research Laboratory (USARL) via the Army High Performance Computing Research Center (AHPARC) through Stanford University, award 60300261-10737-B.

REFERENCES

- [1] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, Sep 2009.
- [2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr 2010.
- [3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.
- [4] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Jun 2010, pp. 49–62.
- [5] M. Powell and B. Miller, "Process migration in demos/mp," *ACM SIGOPS Operating Systems Review*, vol. 17, no. 5, pp. 110–119, Oct 1983.
- [6] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H. Yang, "The case for cyber foraging," in *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*, Sep 2002, pp. 87–92.
- [7] M. Kristensen and N. Bouvin, "Scheduling and development support in the scavenger cyber foraging system," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 677–692, Dec 2010.
- [8] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, pp. 1232–1243, Oct 2012.
- [9] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jul 2002, pp. 217–226.
- [10] R. Balan, M. Satyanarayanan, S. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *ACM International Conference on Mobile Systems, Applications and Services (MobiSys)*, May 2003, pp. 273–286.
- [11] Y. Su and J. Flinn, "Slingshot: deploying stateful services in wireless hotspots," in *ACM International Conference on Mobile Systems, Applications and Services (MobiSys)*, Jun 2005, pp. 79–92.
- [12] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," *Springer Mobile Computing, Applications, and Services*, vol. 76, pp. 59–79, Oct 2012.
- [13] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *ACM Conference on Computer Systems (EuroSys)*, Apr 2011, pp. 301–314.
- [14] K. Jumar, J. Liu, Y. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Springer Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb 2013.
- [15] Z. Li, C. Wang, and R. Xu, "Task allocation for distributed multimedia processing on wirelessly networked handheld devices," in *IEEE-IEE Vehicle Navigation and Information Systems Conference*, Oct 1993, p. ??
- [16] —, "Computation offloading to save energy on handheld devices: A partition scheme," in *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, Nov 2001, pp. 238–246.
- [17] Z. Li and R. Xu, "Energy impact of secure computation on a handheld device," in *IEEE International Workshop on Workload Characterization (WWC)*, Nov 2002, pp. 109–117.
- [18] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading inference for delivering applications in pervasive computing environments," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2003, pp. 107–114.
- [19] H. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, "Roam, a seamless application framework," *Elsevier Journal of Systems and Software*, vol. 69, no. 3, pp. 209–226, Jan 2004.
- [20] C. Wang and Z. Li, "A computation offloading scheme on handheld devices," *Journal of Parallel and Distributed Computing*, vol. 64, no. 6, pp. 740–746, Jun 2004.
- [21] —, "Parametric analysis for adaptive computation offloading," in *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, Jun 2004, pp. 119–130.
- [22] S. Ou, K. Yang, and A. Liotta, "An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2006, pp. 116–125.
- [23] S. Ou, K. Yang, A. Liotta, and L. Hu, "Performance analysis of offloading systems in mobile wireless environments," in *IEEE International Conference on Communications (ICC)*, June 2007, pp. 1821–1826.
- [24] Y. Hong, K. Kumar, and Y. Lu, "Energy efficient content-based image retrieval for mobile systems," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2009, pp. 1673–1676.
- [25] B. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS)*, Jun 2010, pp. 7:1–7:5.
- [26] M. Hassan, W. Zhao, and J. Yang, "Provisioning web services from resource constrained mobile devices," in *IEEE International Conference on Cloud Computing (CLOUD)*, July 2010, pp. 490–497.
- [27] Y. Nimmagadda, K. Kumar, Y. Lu, and C. Lee, "Real-time moving object recognition and tracking using computation offloading," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2010, pp. 2449–2455.
- [28] K. Sinha and M. Kulkarni, "Techniques for fine-grained, multi-site computation offloading," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2011, pp. 184–194.
- [29] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun 2012.
- [30] Y. Zhang, H. Liu, and X. Fu, "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing," in *IEEE International Conference on Cloud Networking (CLOUDNET)*, Nov 2012, pp. 80–86.
- [31] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, Mar 2013.
- [32] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer*, vol. 43, no. 4, pp. 51–56, Apr 2010.
- [33] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *IEEE International Conference on Communications Workshops (ICC)*, June 2013, pp. 728–732.
- [34] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, Apr 2009.
- [35] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Morgan Kaufmann, 2012.
- [36] J. Kurose and K. Ross, *Computer Networking: A Top Down Approach*, 6th ed. Pearson, 2012.
- [37] T. Instruments, "Texas Instruments MSP430," <http://www.ti.com/lit/ds/symmlink/msp430g2553.pdf>, 2013.
- [38] ARM, "Cortex-A8 Processor," <http://www.arm.com/products/processors/cortex-a/cortex-a8.php>, 2015.
- [39] U. of California, "CPU performance," https://setiathome.berkeley.edu/cpu_list.php, 2015.
- [40] Intel, "Intel Core i3-2300 Mobile Processor Series," http://download.intel.com/support/processors/corei3/sb/core_i3-2300_m.pdf, 2013.