Cloud-Based Execution to Improve Mobile Application Energy Efficiency

Eli Tilevich and Young-Woo Kwon, *Virginia Tech*

Cloud offloading—a popular energy optimization technique—executes a mobile application's energy-intensive functionality via a cloud-based server. To maximize efficiency, systems must determine the functionality to offload at runtime, which will require innovation in both automated program transformation and systematic runtime adaptation.

The growing complexity of mobile application functionality poses increasing concerns for device battery capacity (K. Pentikousis, "In Search of Energy-Efficient Mobile Networking," *IEEE Comm. Magazine*, vol. 48, no. 1, 2010, pp. 95-103). Recent research has focused on cloud offloading as a possible technique for improving mobile application energy efficiency. The concept is relatively simple: when a mobile application identifies some functionality as "hot" (that is, energy intensive), it transforms to a distributed application and offloads the required data to a cloud server, which executes the hot functionality and then transfers back the results, thereby saving battery power for the mobile device.

However, most offloading schemes fail to consider the energy required for network transfer, which can sometimes negate any energy savings gained from offloading. Because mobile network conditions vary, the decision whether or not offloading will save energy must occur dynamically at runtime and requires an immediate cost-benefit analysis to determine whether the energy saved by reducing local processing exceeds the overhead required for distributed processing. This analysis is far from trivial due to mobile hardware heterogeneity and execution environment volatility, and it must ultimately involve encapsulating the functionality and exposing it via clean programming abstractions. In other words, adaptive cloud offloading requires innovation in both program transformation and runtime adaptation.

CURRENT LIMITATIONS AND OPPORTUNITIES

In a typical mobile application, network communication consumes up to half the total energy budget. Our studies of distributed programming abstractions and energy efficiency reveal that network types and conditions can significantly affect the energy required for remote data transfer (Y.-W. Kwon and E. Tilevich, "The Impact of Distributed Programming Abstractions on Application Energy Consumption," *Information and Software Technology*, vol. 55, no. 9, 2013, pp. 1602-1613). By considering these variables, mobile computing researchers can create adaptive cloud offloading mechanisms that maximize the energy saved by mobile applications running on a variety of devices over dissimilar networks.

0018-9162/14/$31.00 © 2014 IEEE Published by the IEEE Computer Society

However, most current offloading techniques are preset and so not capable of adapting when an application switches between mobile networks with different characteristics. Those that can adapt their offloading mechanisms at runtime have runtime systems custom-tailored for individual research projects and so lack clear programming interfaces; thus, they can't be configured, reused, or ported easily.

To maximize energy savings in heterogeneous and volatile mobile networks, cloud offloading schemes must support adaptive offloading strategies. Doing so without overburdening programmers requires runtime mechanisms that properly encapsulate the offloading logic by including energy profiling and network monitoring. Such runtime functionality should be exposed via intuitive programming abstractions that allow programmers to express the adaptive logic of offloading decisions, including all required parameters.

REDUCING ENERGY CONSUMPTION USING CLOUD OFFLOADING

Here, we present our cloud offloading research over the past several years, focused primarily on advancing various program analysis and transformation techniques and, more recently, on dynamic runtime support.

Cloud offloading in the presence of network disconnections

Our first major work in cloud offloading was a program transformation approach that preserves a mobile application's ability to execute locally, as it was originally written (Y.-W. Kwon and E. Tilevich, "Energy-Efficient and Fault-Tolerant Distributed Mobile Execution," *Proc. 32nd Int'l Conf. Distributed Computing Systems* [ICDCS 11], IEEE CS, 2012, pp. 586-595). This ability is crucial in the event that the mobile network connecting the device to the cloud becomes disconnected. Rather than split an execution into local and remote partitions, our approach efficiently replicates program state to switch between local and remote executions, which can both reduce client energy consumption and tolerate network outages. When the network is operational, energy-intensive functionality is offloaded to the cloud server by transferring only the program state necessary for remote execution. Efficient check-pointing synchronizes the program state between the local and remote executions. In cases when the network becomes disconnected during offloading, the remote execution redirects to the mobile device. Thus, while network outages inhibit optimal energy use, they do not make the application unusable.

To maximize energy savings in heterogeneous and volatile mobile networks, cloud offloading schemes must support adaptive offloading strategies.

Because transferring large data volumes across the network can consume more energy than is saved by offloading, our second contribution was a program analysis technique that reduces the amount of transferred state. This technique leverages static program analysis to determine whether the program state changed during an offloading operation. The analysis uses programmer annotations as input to specify which methods are expected to be energy intensive. Based on this input, the analysis identifies the state to be transferred for each offloading scenario. Finally, the application's binary bytecode is rewritten to run the resulting applications accordingly, either on the mobile device or in the cloud.

Adaptive multitarget cloud offloading

Building on our prior research, we next focused on increasing energy savings by enhancing the runtime system to make offloading decisions in accordance with the mobile device's hardware setup and the network's characteristics, and to make these decisions dynamically at runtime and adjust them continuously in response to fluctuations in the mobile execution environment (Y.-W. Kwon and E. Tilevich, "Reducing The Energy Consumption of Mobile Applications behind the Scenes," *Proc. 29th IEEE Int'l Conf. Software Maintenance* [ICDCS 13], IEEE CS, 2013, pp. 170-179). The programmer annotates suspected energy hotspots, and, after validating programmer input, the mobile application transforms into a distributed application with local and remote parts determined at runtime, as required by the execution environment. An elaborate checkpointing mechanism makes postponing distribution until runtime possible.

The runtime system controls the adaptivity. It manages network connections between client and server, estimates the mobile device's energy consumption, identifies the offloading strategy to follow, synchronizes the transferred check-pointed state, and provides resilience in the case of network disconnections. In essence, the runtime system continuously monitors the energy that each offloading candidate program component consumes, at the level of its constituent subcomponents. Those subcomponents whose cloud-based execution would save the most energy for the current execution environment network are offloaded.

ADAPTIVE RUNTIME FOR EFFICIENT AND SYSTEMATIC CLOUD OFFLOADING

Systems design that provides powerful runtime adaptivity to support advanced cloud offloading mechanisms will require innovation in a number of areas. Because distributed programming mechanisms define the patterns distributed applications use to transmit data across networks, they significantly impact the energy mobile applications consume. However, existing distributed runtime systems can't adapt execution patterns flexibly to reduce energy consumption when mobile applications switch between dissimilar networks. Maximizing energy savings requires tailoring runtime execution to account for specific applications' business logic; to support this kind of customization, runtime systems need programming abstractions that can express energy optimization parameters and triggering strategies.

Programming abstractions

Runtime programming abstractions can minimize energy consumption via advanced, dynamic optimizations. Enabling these requires innovation in expressiveness and runtime support. Such abstractions let programmers write several cloud offloading plans that can be used under different runtime conditions (accounting for network latency, bandwidth, and volatility, for example). At execution, the runtime system automatically selects an appropriate offloading plan and switches dynamically among plans in response to changes in the execution environment. Runtime systems should be realized as ready-made, reusable software components to reduce the maintenance burden on mobile application developers.

Runtime monitoring and energy profiling

Runtime adaptation requires careful monitoring of runtime changes in the execution environment without imposing undue performance overhead. In our current work, we are exploring how runtime systems can unobtrusively gather execution parameters, including network and hardware parameters such as delay, network connection type, and CPU frequency. Using these parameters makes it possible to build a powerful adaptive system that can predict how much energy a given cloud offloading operation will consume. The runtime system can correlate previously obtained device- and execution-specific values, using network delay, connectivity type, CPU frequency, and voltage values to compute future energy consumption.

Custom-tailored runtime adaptation

In addition to automated adaptations, mobile application programmers can implement custom optimization strategies that exploit the application's business logic. Integrating these custom-tailored adaptations with the runtime system can further improve cloud offloading efficiency. A typical custom-tailored optimization can comprise well-known energy optimizations for network communication, including data compression, reducing the offloaded data's size, and selecting the easiest-to-reach remote server. These energy optimizations should be selectable on per-application and per-user bases.

Runtime adaptation provides a highly promising avenue for improving cloud offloading effectiveness. Just as important, this improved efficiency does not need to come at the cost of systematic software development practices. We hope these insights can help influence the research agenda for optimizing future mobile application energy efficiency in the cloud.

Acknowledgment

The National Science Foundation supported this research through grant CCF-1116565.

Eli Tilevich *is an associate professor in Virginia Tech's Department of Computer Science. Contact him at tilevich@cs.vt.edu.*

Young-Woo Kwon *is a PhD candidate in Virginia Tech's Department of Computer Science. Contact him at ywkwon@cs.vt.edu.*

Editor: Kirk Cameron, Department of Computer Science, Virginia Tech; greenit@computer.org