CHAPTER

3 DEVICE/CLOUD COLLABORATION FRAMEWORK FOR INTELLIGENCE APPLICATIONS

Y. Yoon*, D. Ban, S. Han**, D. An†, E. Heo****

**Hongik University, Wausan-ro, Mapo-gu, Seoul, South Korea; **Samsung Electronics, South Korea; †Keimyung University, Dalgubeol-daero, Dalseo-gu, Daegu, South Korea*

3.1 INTRODUCTION

Cloud computing is now an established computing paradigm that offers on-demand computing and storage resources to the users who cannot afford the expenditure on computing equipment and management workforce. This computing paradigm first led to the notable commercial success of Amazon's EC2 [1] and Microsoft's Azure [2]. These companies have adopted the business model of renting out their virtualized resources to the public. More recently, Google and Facebook are now utilizing their data-center-based clouds to internally run machine-learning algorithms based on the large volume of data collected from their users. Google runs a popular proactive service, Google Now, which gives individualized recommendations based on the user's context inferred from the personal data [3]. Facebook leverages the user-uploaded images and social-network data to automatically recognize users and the relationship among them [4,5].

Despite the continuous growth, many organizations have raised the concerns about the cloud computing with respect to performance and privacy. In the following section, we elaborate on those issues.

3.2 BACKGROUND AND RELATED WORK

Public cloud-vendors have attracted institutions who have little incentive to pay the upfront cost of IT infrastructure, especially when the applications and services they host do not require high compute and storage capacity. However, once their applications and services become popular, the demand for higher compute and storage capacity may suddenly soar. This is the point when these institutions may not feel that public cloud is cost-effective anymore, and consider opting for a different cloud computing model.

For example, one may adopt a hybrid cloud computing model that utilizes both the public and the private cloud infrastructure [6]. A typical use case of this model is to do *workload bursting*, that is, offloading tasks from the public cloud to the self-managed private cloud for cost savings [7]. Some institutions may rather resort to embarking on an entirely private cloud infrastructure. For example, in

2015, Samsung Electronics migrated S Voice, its mobile voice-based personal-assistance service, from a proprietary public cloud to its home-grown cloud infrastructure. This migration has led to service performance improvement and reduced management cost. However, Samsung's own cloud infrastructure is small in scale and serves only a special purpose. Therefore, whether it can scale out to host various other services is questionable.

Some organizations have considered forming a community that shares cloud infrastructure and management resources. In this model, the community members can share their cloud resources with one another. In addition, the shared cloud resources can be offered to nonmembers to drive extra revenue. Recently, companies such as IBM and Samsung have teamed up to push for the community cloud-computing initiatives [8]. However, there are many hurdles ahead, such as crafting an optimal policy for sharing compute and management resources among the participants.

Whereas the aforementioned concerns are related to the cost and efficiency issues, many cloud users have also expressed their fears about the security and privacy breach. For example, a number of private photos of celebrities were leaked from Apple's iCloud in 2014 [9]. Since then, Samsung Electronics has had a hard time mobilizing on-device personal data to the cloud for analytics service because many device owners strongly demand privacy protection. These concerns hamper the effort of advancing personalized intelligence services because the collection of private data is critical for the quality of such services.

In the remainder of this chapter, we present a novel cloud-computing framework that improves both the scalability and the privacy-protection mechanism. At a high level, this framework leverages the compute and storage resources on the smart mobile devices. Also, this framework enables security solutions that protect privacy without degrading the quality of applications. Note that we focus on the applications that offer personalized intelligence service. Therefore, we demonstrate how the selected real-world intelligence applications take advantage of the new cloud-computing framework.

3.3 DEVICE/CLOUD COLLABORATION FRAMEWORK

In this section, we present a novel framework that enables collaboration between smart mobile devices and cloud for a more scalable and secure computing mechanism.

3.3.1 POWERFUL SMART MOBILE DEVICES

In 2014, more than 1 billion smartphones were sold worldwide, and more than 2 billion consumers are expected to get a smartphone by 2016 [10]. Smartphones[a] nowadays have enough computing capacity to process various computing tasks. However, the device usage can be constrained by limited battery life and network connectivity. Therefore, we can consider utilizing the highly available cloud resources in addition to the device.

In Fig. 3.1, we have illustrated a high-level layout of our device/cloud collaboration framework. In the following section, we will explain how the framework functions to address the aforementioned concerns on performance and privacy.

[a]We will focus on the smartphones as the representative smart mobile devices. In the remainder of this chapter, the term *device* will actually refer to smartphones.

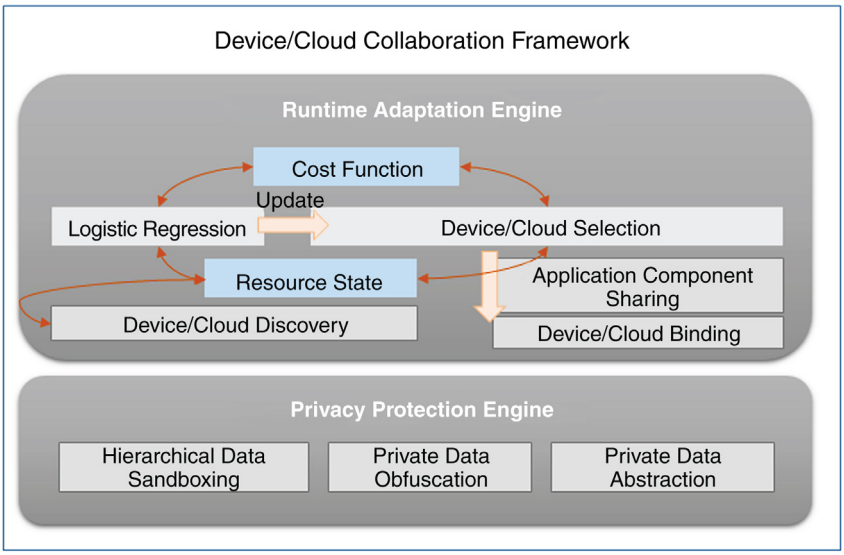

FIGURE 3.1 High-Level Layout of the Device/Cloud Collaboration Framework

3.3.2 RUNTIME ADAPTATION ENGINE

Given a computation task, our framework first faces a problem of choosing the entity to execute the task. Suppose we are given a task of processing a query issued over voice, and assume that we have a lightweight mobile version of a voice-query processing engine that is embedded in a smartphone. This mobile engine can answer the given query without the cost of transferring the voice data to the cloud over the network. However, it will consume the limited battery life of the device, and/or the accuracy of the result may not necessarily be as good as that of the cloud-based query processing engine, which runs on resources with higher compute capacity. If the lightweight voice-query engine returns a poor result, then the user may have to issue the query redundantly to the cloud-based query processing engine with the hope of getting a better result. This may hurt the overall quality of experience (QoE). This calls for a decision mechanism that automatically selects a better *agent* that can execute a given task. With the automatic selection process in place, users do not have to worry about going through extra interaction cycles for determining where to run a job.

Note that the *Runtime Adaptation Engine* (RAE) sits at the core of our framework, as shown in Fig. 3.1. The RAE maintains a list of available devices and cloud to utilize *Device/Cloud Discovery*, and monitors the state of their available resources. The RAE employs a *Logistic Regression* [11] algorithm to learn the most cost-effective policy for distributing tasks among devices and cloud, given the resource state. Here, the definition of the cost function is the weighted sum of the resource state (such as battery life), network, and CPU usage. The policy obtained by running the Logistic Regression is enforced by the *Device/Cloud Selection* module that chooses the most economical compute resources, based on the expected cost-value for a given task.

The mechanism of the RAE is actually an autonomous agent, which can be deployed on each device and cloud. RAEs communicate with each other to transparently share the resource state for determining the ways to distribute a given workload. The cloud-side RAE can also model its own cost function

as a weighted sum of residual CPU cycles and storage space across the entire infrastructure. If the residual capacity falls below particular thresholds, the cloud may have to reject the resource-sharing request coming from the paired devices. This is because running the requested task would be too costly. Specifically, the cloud-side RAE advises the device-side RAE to either execute the task within the device or simply wait for the compute resource on the cloud to be freed up. As mentioned earlier, the RAEs on the devices and the cloud make decisions *autonomously*, without any supporting brokerage system in the middle. However, the device-side RAE has the burden of periodically monitoring the state of the cloud resources. On the other hand, the cloud-side RAE does not have to monitor the resource state of the millions of paired devices, as the cloud makes a relatively simple decision, that is, to either reject or accept a task-execution request. By default, our framework does not consider offloading cloud-initiated tasks to the devices. However, later in Section 3.4, we will show a case where the original cloud-side application components can be customized to run on the mobile devices.

We have focused on the case where the user's task is distributed between a *single* device and cloud. However, a recent study has revealed that more than 60% of the online-service users use at least two devices daily, and about 25% of the users use three devices [12]. This fact motivates us to consider the collaboration among devices, as well as through machine to machine (M2M) communication channels, such as Samsung's *AllShare* Convergence solution [13]. By considering the neighboring devices as well, the overall compute resource availability increases further, and the computation burden on the cloud can be reduced significantly. To support task distribution among the neighboring devices, the device/cloud collaboration framework implements discovery of devices (*Device/Cloud Discovery* in Fig. 3.1). Suppose a device (device-A) wants to collaborate with its neighboring device (device-B), which is not equipped with appropriate application components to process a given task. Then, device-A can transfer the necessary application components to device-B through the *Application Component Sharing* (shown in Fig. 3.1).

3.3.3 PRIVACY-PROTECTION SOLUTION

So far, we have focused on the aspects of the device/cloud collaboration framework that are concerned with performance and efficiency. Now, we turn our attention to the privacy-protection problem. Suppose we want to provide a location-based service to the users. To provide this service, location data such as the visited GPS coordinates, the point of interest (POI), and the time of visit have to be collected first. Based on these location data, the mobility pattern and the interest of individual users can be inferred. However, these data contain personal information. Hence, for these data, we can ask the user to decide whether to transfer them to cloud or not when accepting the location-based service. Based on the decision made by the user, our framework can assess the expected service-quality for the users. Suppose a user wants to transfer to the cloud a blurred location log with entries that are simply labeled either <at home> or <not at home>. Given this log with little detail, it is difficult to expect a service provider to offer a useful location-based recommendation. In such a case, our framework warns the user that disallowing the sharing of detailed private information would result in poor service quality.

Our framework employs the technology of protecting privacy by sandboxing hierarchically organized application-data [14]. This technology, implemented in the *Hierarchical Data Sandboxing* module (in Fig. 3.1), supports the user to explicitly specify a group of data in the hierarchy to be shared with the cloud or not. The group of data that is set to be kept only within a device will be protected by sandboxing. Although this approach supports a specification of fine-grained privacy-protection policies, such a

declarative approach would be too cumbersome for many typical users. Thus, we can seek an alternative solution of obfuscating (encrypting) data to be transferred to the neighboring devices or to the cloud (*Data Obfuscation* in Fig. 3.1). A natural solution is to encrypt the data upon transferring to cloud [15]. However, the encrypted data can be revealed through decryption-key theft from the compromised servers on the cloud or by spoofing on the tapped network. For these security threats, the cost of countermeasures, such as secret (eg, decryption keys) sharing across replicated servers, is nonnegligible [16]. Instead, we can have a lighter approach of letting the cloud analyze the obfuscated data without deciphering it, and letting the device revert the obfuscated part of the analysis result generated at the cloud side. For instance, suppose a user wants to receive a location-based recommendation based on the personal log of visited POIs. Both the POI itself and the time of POI visits are first obfuscated on the device side. The mapping between the original data and the encrypted data is kept on the device side. The device sends over the encrypted data to the cloud that does not have a decryption key to decipher the encrypted data. On the cloud side, data analytics such as causal reasoning through sequence mining [17] are conducted, based on the encrypted information (eg, POIs and time of visits). For example, suppose a user has the following entries in the location log, as shown in Table 3.1. Each entry contains the mapping between the original data and the encrypted data.

Note that the cloud has to be aware of the data format, that is, the data contains POI and the time of visits. Assume that the sequence-mining engine on the cloud side infers the frequent mobility pattern that the user visits: `b731d61a5be2b9035a20ebef5aa9bfef` (actually Bryant Park) after the user visits: `24e3b66da54d1e21b177ea3351a0e4c2` (actually Starbucks). The cloud notifies this inference result to the device without knowing the actual private content. Once the device receives this result, it deciphers the content by checking the mapping between the encrypted data and the original data. Given the inferred mobility pattern, the device can invoke a third-party recommendation service to receive a list of recommended activities or events around Bryant Park whenever the user is about to leave Starbucks. Here, we assume a threat model that the third-party recommendation service independently enforces its own security measures to prevent the leakage of private queries, and the device itself is safe from being compromised.

Another approach is abstracting given data to hide the details, as shown in Table 3.2.

Table 3.1 A Mapping Between Original and Encrypted Location Data

Original Data	Encrypted Data
Starbucks on 575 5th Avenue, New York, NY, 11 am, Jan. 15th, 2015	`24e3b66da54d1e21b177ea3351a0e4c2`, `a44cdcecb384fd730553e59eed867e63`
Bryant Park in New York, NY, 1 pm, Jan. 15th, 2015	`b731d61a5be2b9035a20ebef5aa9bfef`, `84d9cfc2f395ce883a41d7ffc1bbcf4e`

Table 3.2 Mapping Between the Original and Abstracted Location Data

Original Data	Abstracted Data
Starbucks on 575 5th Avenue, New York, NY, 11 am, Jan. 15th, 2015	Café in the morning
Bryant Park in New York, NY, 1 pm, Jan. 15th, 2015	Public park in the afternoon

For example, a visit to *Starbucks on 575 5th Avenue, New York, NY at 11 am* can be abstracted as a visit to *a café in the morning*. The benefit of this approach is that it can generate recommendations directly on the cloud upon recognition of the frequent visiting patterns. However, the accuracy of the recommendation can be compromised due to the loss of detailed information.

3.4 APPLICATIONS OF DEVICE/CLOUD COLLABORATION

In this section, we show how our framework can be used by the real-world intelligence applications developed specifically at Samsung Electronics. The selected applications offer the following functionalities: *context-aware proactive suggestion, semantic QA caching, and automatic image/speech recognition*. We introduce the interesting practical engineering experiences of adapting the application in order to leverage the framework in the most effective manner.

3.4.1 CONTEXT-AWARE PROACTIVE SUGGESTION

Based on the personal data collected on each mobile device, we have devised Proactive Suggestion (PS), an application that makes context-aware recommendations. In Fig. 3.2, the individual components of the PS are laid out.

Analytics engines of PS produce hierarchical personal data that are interdependent to each other. Raw data such as GPS coordinates, call logs, application usage, and search queries are fed to a *Cooccurrence Analysis* engine, which is responsible for identifying activities that occurred at the same

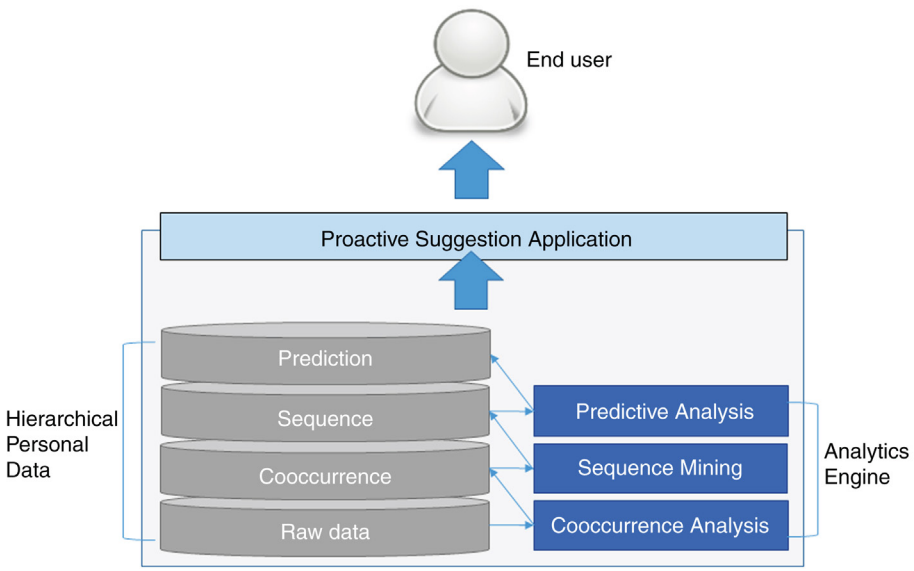

FIGURE 3.2 High-Level Layout of the Core Components for the Proactive Suggestion Application

Analytics engines process personal data to produce contextual data that are used for multilevel recommendations to the end user.

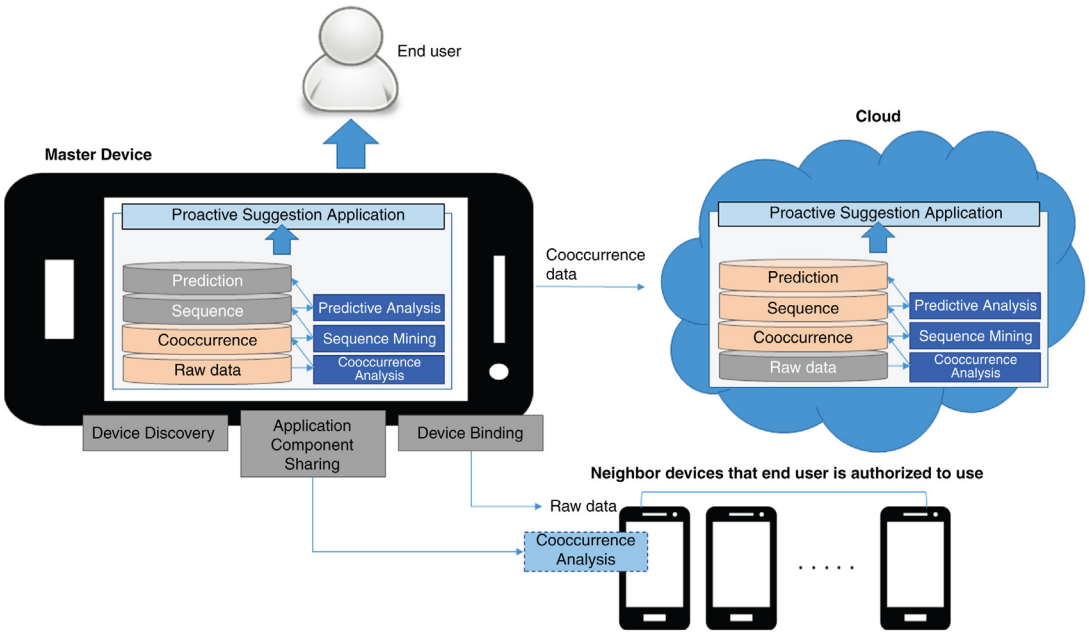

FIGURE 3.3 An Example of Utilizing the Device-Collaboration Framework for the Proactive Suggestion Application

time [18]. For example, the cooccurrence analysis engine may recognize that a user listens to live streaming music *while* walking in the park. Given such cooccurrence data, the *Sequence Mining* engine can infer causal relationships between personal activities that occurred over time [19]. The recognized sequential patterns can be fed into the *Predictive Analysis* engine to assess the probability of a particular activity taking place in a certain context [19].

Fig. 3.3 illustrates how PS implements the device/cloud collaboration framework. The master device can discover neighboring devices that the end user is authorized to use (*Device Discovery*). The master device can send over the data to one of the neighboring devices that has sufficient compute capacity (*Device Binding*). The neighboring device can retrieve an appropriate analytics engine for processing the data sent by the master device (*Application Component Sharing*). In this example, the highlighted pieces of data on the master device are shared between cloud and neighboring devices.

Note that the PS application initially opted for the *Hierarchical Data Sandboxing* for an explicit and declarative privacy-protection method. We could not afford to run an alternative privacy-protection method based on the data obfuscation, due to the limited resources on the device that was already bogged down by the analytics work. However, recall that our framework is flexible enough to allow user-defined cost functions. For example, if the cost of running an analytics operation (eg, the cost of consuming battery life) is excessive, then the *Device/Cloud Selection* module in the framework may decide to transfer the analytics task to the cloud or simply wait for the battery level to rise above the configured thresholds. It turned out that transferring the data over the network consumed as much energy as running the analytics operation within the device. Thus, the *Device/Cloud Selection* module opted for waiting until the battery got charged above the configured level.

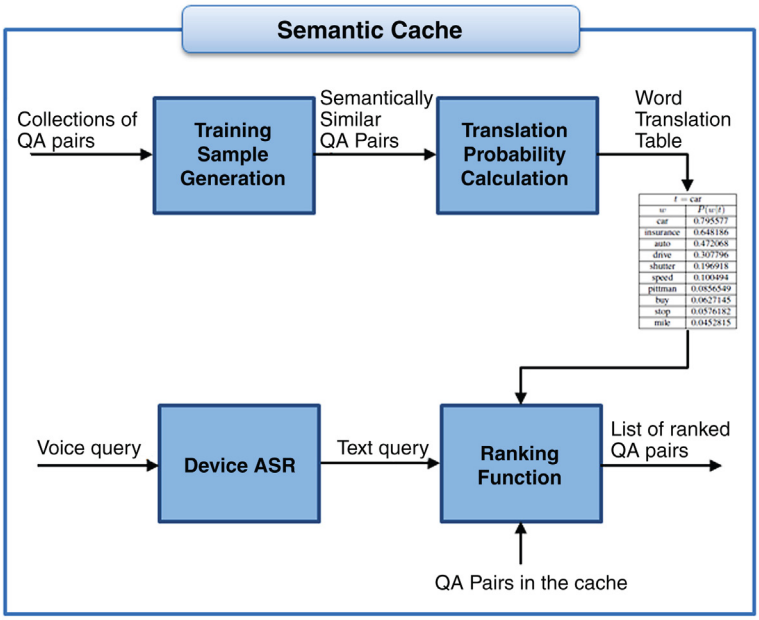

FIGURE 3.4 Illustrations of the Technique to Cluster Semantically Similar QA Pairs for Retrieving an Answer for a Newly Given Query Without Asking the QA Engine on the Cloud Side

3.4.2 SEMANTIC QA CACHE

Semantic QA cache is a mobile application that retrieves answers to a given query from the cache filled with answers to the semantically similar queries issued in the past. Semantic QA cache can be useful when there is no Internet connectivity or when the user is not in favor of transferring private queries to the cloud. Fig. 3.4 illustrates how the semantic QA cache is managed. Semantic QA cache returns a list of similar queries and the associated answers. Semantic QA cache constantly updates ranking function based on the word-translation table as explained in [20]. The ranking function measures the similarity between a newly issued query and the queries measured in the past.

In Fig. 3.5, we have demonstrated the implementation of the device/cloud collaboration framework by the semantic QA cache. Specifically, we have devised a custom ASR (Automatic Speech Recognition) engine for the mobile device and incorporated the cloud system for Samsung S Voice in the collaboration framework. The cloud system for S Voice consists of a Natural Language Understanding (NLU) module for query understanding, a DM (Dialog Manager) module for query answering, and a powerful ASR engine.

Note that we have adapted the framework to compute the probability of the on-device semantic QA cache to answer a given query correctly. If the probability is high enough, the *Device/Cloud Selection* module will take the risk of looking up the semantic QA cache for an answer. If the cache does not return the right answer and forces the user to ask the cloud again, then our framework will adjust the probability accordingly.

We evaluated the performance benefit of using the device/cloud collaboration framework for semantic QA cache. From the log of our voice-based QA application, we obtained the top-50 frequently issued queries about weather, restaurants, people, and device-specific commands. We selected a random query from the set according to uniformly random distribution (Method 1) and Zipf distribution

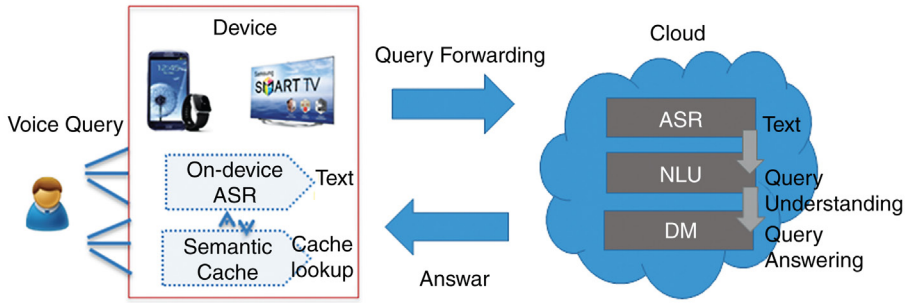

FIGURE 3.5 Semantic QA Cache Implementing the Device/Cloud Collaboration Framework

(Method 2). The latency of getting the response for a query was tested on the cloud-only mode and the device/cloud collaboration mode. In the cloud-only mode, server-version of Google Voice API was used for ASR, and DBpedia and Freebase were used for query answering. In the device/cloud collaboration mode, a custom-made ASR engine and semantic QA cache were used, along with a cloud-based QA service. Leveraging the device/cloud collaboration improved performance for both types of query workloads. The latency was reduced by 56.7 and 69.5% for Method 1 and Method 2, respectively.

3.4.3 IMAGE AND SPEECH RECOGNITION

Automatically recognizing images and speech can greatly enhance a user's experience in using applications. For example, with automatic image recognition, photos taken by a user can be automatically tagged with metadata and catalogued more easily. Similar to Amazon's Firefly [21], we have developed an application called Watch&Go, which lets users obtain detailed information about a product upon taking a photograph. Fig. 3.6 shows the snapshot of Watch&Go that guides users to properly focus on some electronics products, and automatically retrieve information such as type, vendor, model name, and the result of social sentiment about the product.

Practicality of these recognition applications has greatly improved, thanks to the recent advancement of Deep Learning (DL). The DL follows the approach of learning the correlation between the parameters across multiple layers of perceptron [22]. However, DL model training methods usually suffer a slow learning curve compared to the other conventional machine-learning methods. Although it is generally believed that the larger DL model improves the recognition accuracy through a set of well-refined training data, it has been challenging to acquire adequate parameters when we train multiple layers at the same time. The recent appearance of the Restricted Boltzmann Machine (RBM) method, which enables layer-wise and unsupervised training, can relax the aforementioned limitations to some degree. However, the overall computational overhead is still formidable, even for the cloud with abundant compute resources. This performance issue has motivated us to utilize our device/cloud collaboration framework as follows.

Through our framework, the compute-intensive part of DL (ie, the training) is assigned to cloud. Once the learning completes, our framework ports the recognition model to the device for the actual execution of the recognition task. Specifically, we used an *ImageNet-1000* model that was constructed based on a Convolutional Neural Networking (CNN) [23,24] method. With this model, classification of up to 1000 different objects is possible. However, an open-source image-classifier (OpenCV) on an Android device took more than 20 s to classify an object with ImageNet-1000. This was due to the

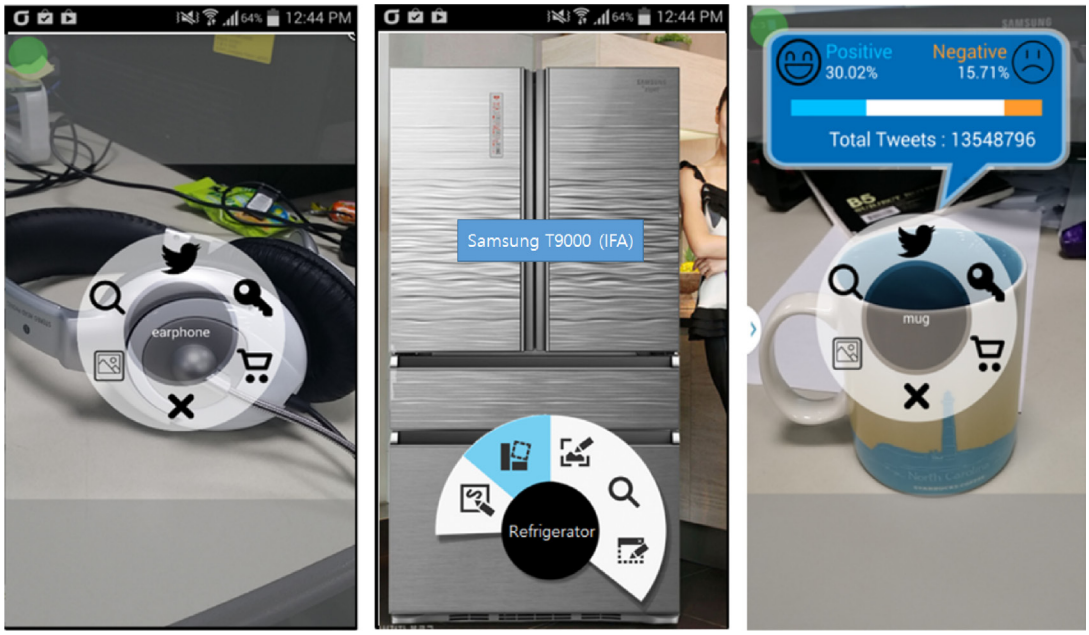

FIGURE 3.6 An Example of Automatically Tagging Recognized Images and Displaying Additional Information Such as Social Sentiment (eg, Positive or Negative Reviews)

inefficient matrix multiplication on the device. We have overcome this problem by parallelizing the matrix multiplication based on OpenCL [25], resulting in the classification latency dropping to an average of 400 ms per object. By utilizing the low-latency on-device image classifier on millions of predeployed mobile devices, we were able to reduce the computational burden on the cloud significantly.

We have achieved similar performance improvement for speech-recognition application with DL through our device/cloud collaboration framework. Specifically, we first extracted 400-h worth of speech data from Fisher Corpus. Contrary to the image-recognition problem, we have employed a Deep Neural Network (DNN) model, which is shown to be effective in constructing an accurate acoustic model [26]. Similar to image recognition, we have assigned the acoustic model-construction task and the classification task to cloud and mobile devices, respectively. Specifically, we ported Kaldi [27] to an Android device in order to process a speech-recognition request based on the constructed acoustic model. The task separation through our device/cloud collaboration framework and the additional acceleration through OpenCL helped us obtain the recognition result within 0.9 RT (Real Time),[b] which is a tolerable delay for the end users.

We could relieve the computational burden on the cloud side further by splitting the learning portion. Lightweight models can be constructed within a mobile device. However, the classification accuracy can be compromised when these models are used. We have observed that the tolerant accuracy level varies between different end-users. Hence, our framework can be adapted to learn the personal tolerance level and determine which model to construct accordingly.

[b]1RT is the criteria which decides the applicability of speech recognition (the lesser value is the better).

3.5 FUTURE WORK

Various ways of componentizing a given intelligence application come with different trade-offs. A more fine-grained componentization, as shown in the PS application, may yield more efficient task distribution among neighboring devices and cloud. Also, fine-grained privacy-protection policies can be applied to these components. However, computing more efficient task distribution incurs additional overhead. It may cost relatively lower overhead to determine execution entities for the coarse-grained. But it can be nontrivial to port a cloud-side coarse-grained component to a resource-limited device when it is deemed necessary. Even if the coarse-grained component is tailored to fit in the device, the quality of the component may degrade. As a future work, we plan to study the effect of different componentization strategies in various application scenarios and guide the developers to pick the best strategy that maximizes collaboration performance.

3.6 CONCLUSIONS

We have presented the benefits of using a collaborative computation framework between devices and cloud through the case studies of selected real-world intelligence applications devised at Samsung Electronics. Applications that implement the device/cloud collaboration framework can yield high performance, such as reduced latency in processing a user's request. In addition, the cost of managing the cloud can be reduced when the compute resources on the millions of smart mobile devices are utilized. Aside from the benefits in terms of cost and performance, the framework helps the application protect privacy of the end users by either processing personal data within a device or analyzing the obfuscated version of the personal data on cloud.

ACKNOWLEDGMENTS

This chapter is derived from the authors' work from The Grand Bleu Project that was conducted by Intelligence Solution Team (IST) at Samsung Electronics in 2014. We would like to thank Dr Kilsu Eo, who directed IST and initiated The Grand Bleu Project. We are also grateful for the guidance provided by the project leaders, Dr Honguk Woo and Dr Sangho Shin.

REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2). https://aws.amazon.com/ec2
[2] Microsoft Azure: Cloud Computing Platform & Services. https://azure.microsoft.com
[3] Google Now. https://www.google.com/landing/now
[4] Wang C, Raina R, Fong D, Zhou D, Han J, Badros G. Learning relevance from a heterogeneous social network and its application in online targeting. In: Proceedings of the thirty-fourth international ACM SIGIR conference on research and development in information retrieval. Beijing, China, July 24–28, 2011.
[5] Backstrom L, Sun E, Marlow C. In: Proceedings of the nineteenth international conference on World Wide Web, Raleigh, NC, USA, April 26–30, 2010.
[6] Sotomayor B, Montero R, Llorente I, Foster I. Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 2009;13(5):14–22.

[7] Bossche R, Vanmechelen K, Broeckhove J. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: Proceedings of the 2010 IEEE third international conference on cloud computing, Miami, FL, USA, July 5–10, 2010.
[8] Briscoe G, Marinos A. Digital ecosystems in the clouds: towards community cloud computing. In: Proceedings of the third IEEE international conference on digital ecosystems and technologies. Istanbul, Turkey, June 1–3, 2009.
[9] 2014 Celebrity Photo Hack. https://en.wikipedia.org/wiki/2014_celebrity_photo_hack
[10] emarketer.com. http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694
[11] Cox DR. The regression analysis of binary sequences. J R Stat Soc Ser B Methodol 1958;20(2):215–42.
[12] Econsultancy.com. https://econsultancy.com/blog/64464-more-than-40-of-online-adults-are-multi-device-users-stats/
[13] Samsung AllShare Play. http://www.samsung.com/us/2012-allshare-play/
[14] Lee S, Wong E, Goel D, Dahlin M, Shmatikov V. PiBox: a platform for privacy preserving apps. In: Proceedings of the 2013 tenth USENIX conference on networked system design and implementation, Lambard, IL, USA, April 2–5, 2013.
[15] Pearson S, Shen Y, Mowbray M. A privacy manager for cloud computing. In: Proceedings of the first international conference on cloud computing. Bangalore, India, September 21–25, 2009.
[16] Itani W, Kayssi A, Chehab A. Privacy as a service: privacy-aware data storage and processing in cloud computing architectures. In: Proceedings of the eighth IEEE international conference on dependable, autonomic and secure computing. Chengdu, China, December 12–14, 2009.
[17] Agrawal R, Srikant R. Mining sequential patterns. In: Proceedings of the eleventh international conference on data engineering. Taipei, Taiwan, March 6–10, 1995.
[18] Srinivasan V, Moghaddam S, Mukherji A, Rachuri KK, Xu C, Tapia EM. MobileMiner: mining your frequent patterns on your phone. In: Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing. Seattle, USA, September 13–17, 2014.
[19] Mukherji A, Srinivasan V, Welbourne E. Adding intelligence to your mobile device via on-device sequential pattern mining. In: Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: adjunct publication. Seattle, USA, September 13–17, 2014.
[20] Xue X, Jeon J, Croft W. Retrieval models for question and answer archives. In: Proceedings of the thirty-first annual international ACM SIGIR conference on research and development in information retrieval. Singapore, July 20–24, 2008.
[21] Amazon Firefly. https://developer.amazon.com/public/solutions/devices/fire-phone/docs/understanding-firefly
[22] Bengio Y, Goodfellow I, Courville A. Deep learning. USA: MIT Press; 2015.
[23] Hinton G. A practical guide to training restricted Boltzmann machine. Technical Report UTML TR 2010-003, Dept. of Computer Science, Univ. of Toronto; 2010.
[24] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks. In: Proceedings of the twenty-sixth annual conference on neural information processing systems (NIPS), Lake Tahoe, NV, USA, December 3–8, 2012.
[25] OpenCL (Open Computing Language). https://www.khronos.org/opencl/
[26] Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks. In: Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP). Vancouver, BC, Canada, May 26–31, 2013.
[27] Kaldi Project. http://kaldi-asr.org/