2013 IEEE International Conference on Cloud Engineering

The Impact of Mobile Multimedia Applications on Data Center Consolidation

Kiryong Ha*, Padmanabhan Pillai†, Grace Lewis‡, Soumya Simanta‡, Sarah Clinch§, Nigel Davies§ and Mahadev Satyanarayanan*

*Carnegie Mellon University, Email: {krha, satya}@cs.cmu.edu,
†Intel Labs, Email: padmanabhan.s.pillai@intel.com
‡CMU-SEI, Email: {ssimanta, grace}@sei.cmu.edu,
§Lancaster University, Email: seclinch@gmail.com, nigel@comp.lancs.ac.uk

***Abstract*—The convergence of mobile computing and cloud computing enables new multimedia applications that are both resource-intensive and interaction-intensive. For these applications, end-to-end network bandwidth and latency matter greatly when cloud resources are used to augment the computational power and battery life of a mobile device. We first present quantitative evidence that this crucial design consideration to meet interactive performance criteria limits data center consolidation. We then describe an architectural solution that is a seamless extension of today's cloud computing infrastructure.**

***Keywords*-Cloud Computing; Mobile Computing; Data Center Placement; Cloudlets; Virtual Machines; Network Latency; Wireless Networks; Face Recognition; Speech Recognition; Object Recognition; Augmented Reality;**

I. Introduction

The convergence of cloud computing and mobile computing has begun. Apple's *Siri* for the iPhone [1], which performs compute-intensive speech recognition in the cloud, hints at the rich commercial opportunities in this emerging space. Rapid improvements in sensing, display quality, connectivity, and computational capacity of mobile devices will lead to new cloud-enabled mobile applications that embody voice-, image-, motion- and location-based interactivity. Siri is just the leading edge of this disruptive force.

Many of these new applications will be interactive as well as resource-intensive, pushing well beyond the processing, storage, and energy limits of mobile devices. When their use of cloud resources is in the critical path of user interaction, end-to-end operation latencies can be no more than a few tens of milliseconds. Violating this bound results in distraction and annoyance to a mobile user who is already attention-challenged. Such fine-grained cloud usage is different from the coarse-grained usage models and SLA guarantees that dominate cloud computing today.

The central contribution of this paper is the experimental evidence that these new applications force a fundamental change in cloud computing architecture. We describe five example applications of this genre in Section II, and experimentally demonstrate in Section III that even with the rapid improvements predicted for mobile computing hardware, such applications will benefit from cloud resources. The remainder of the paper explores the architectural implications of this class of applications. In the past, *centralization* was the dominant theme of cloud computing. This is reflected in the consolidation of dispersed compute capacity into a few large data centers. For example, Amazon Web Services spans the entire planet with just a handful of data centers located in Oregon, northern California, Virginia, Ireland, Singapore, Tokyo, and São Paolo. The underlying value proposition of cloud computing is that centralization exploits economies of scale to lower the marginal cost of system administration and operations. These economies of scale evaporate if too many data centers have to be maintained and administered.

Aggressive global consolidation of data centers implies large average separation between a mobile device and its cloud. End-to-end communication then involves many network hops and results in high latencies. Section IV quantifies this point using measurements from Amazon EC2. Under these conditions, achieving crisp interactive response for latency-sensitive mobile applications will be a challenge. Limiting consolidation and locating small data centers much closer to mobile devices would solve this problem, but it would sacrifice the key benefit of cloud computing.

How do we achieve the right balance? Can we support latency-sensitive and resource-intensive mobile applications without sacrificing the consolidation benefits of cloud computing? Section V shows how a two-level architecture can reconcile this conflict. The first level of this hierarchy is today's unmodified cloud infrastructure. The second level is new. It consists of dispersed but unmanaged infrastructure with no hard state. Each second-level element is effectively a "second-class data center" with soft state generated locally or cached on demand from the first level. Data center proximity to mobile devices is thus achieved by the second level without limiting the consolidation achievable at the first level. Communication between first and second levels is outside the critical path of interactive mobile applications.

Throughout this paper, the term "cloud computing" refers to transient use of computational cloud resources by mobile clients. Other forms of cloud usage such as processing of large datasets (data-intensive computing) and asynchronous long-running computations (agent-based computing) are outside the scope of this paper.

II. Mobile Multimedia Applications

Beyond today's familiar desktop, laptop, and smartphone applications is a new genre of software to seamlessly augment human perception and cognition. Consider Watson, IBM's question-answering technology that publicly demonstrated its prowess in 2011 [2]. Imagine such a tool being available anywhere and anytime to rapidly respond to urgent questions posed by an attention-challenged mobile user. Such a vision may be within reach in the next decade. Free-form speech recognition, natural language translation, face recognition, object recognition, dynamic action interpretation from video, and body language interpretation are other examples of this genre of futuristic applications. Although a full-fledged cognitive assistance system is out of reach today, we investigate several smaller applications that are building blocks towards this vision. Five such applications are described below.

*A. Face Recognition (*FACE*)*

A most basic and fundamental perception task is the recognition of human faces. The problem has been long studied in the computer vision community, and fast algorithms for detecting human faces in images have been available for some time [3]. Identification of individuals through computer vision is still an area of active research, spurred by applications in security and surveillance tasks. However, such technology is also very useful in mobile devices for personal information management and cognitive assistance. For example, an application that can recognize a face and remind you who it is (by name, contact information, or context in which you last met) can be quite useful to everyone, and invaluable to those with cognitive or visual impairments. Such an application is most useful if it can be used anywhere, and can quickly provide a response to avoid potentially awkward social situations.

The face recognition application studied here detects faces in an image, and attempts to identify the face from a pre-populated database. The application uses a Haar Cascade of classifiers to do the detection, and then uses the Eigenfaces method [4] based on principal component analysis (PCA) to make an identification. The implementation is based on OpenCV [5] image processing and computer vision routines, and runs on a Microsoft Windows environment. Training the classifiers and populating the database are done offline, so our experiments only consider the execution time of the recognition task on a pre-trained system.

*B. Speech Recognition (*SPEECH*)*

Speech as a modality of interaction between human users and computers is a long studied area of research. Most success has been in very specific domains or in applications requiring a very limited vocabulary, such as interactive voice response in phone answering services, and hands-free, in-vehicle control of cell phones. Several recent commercial efforts aim for general purpose information query, device control, and language translation using speech input on mobile devices [1], [6], [7].

The speech recognition application considered here is based on an open-source speech-to-text framework based on Hidden Markov Model (HMM) recognition systems [8]. It takes as input digitized audio of a spoken English sentence, and attempts to extract all of the words in plain text format. This application is single-threaded. Since it is written in Java, it can run on both Linux and Microsoft Windows. For this paper, we ran it on Linux.

*C. Object and Pose Identification (*OBJECT*)*

A third application is based on a computer vision algorithm originally developed for robotics [9], but modified for use by handicapped users. The computer vision system identifies known objects, and importantly, also recognizes the position and orientation of the objects relative to the user. This information is then used to guide the user in manipulating a particular object.

Here, the application identifies and locates known objects in a scene. The implementation runs on Linux, and makes use of multiple cores. The system extracts key visual elements (SIFT features [10]) from an image, matches these against a database of features from a known set of objects, and finally performs geometric computations to determine the pose of the identified object. For the experiments in this paper, the database is populated with thousands of features extracted from more than 500 images of 13 different objects.

*D. Mobile Augmented Reality (*AUGREAL*)*

The defining property of a mobile augmented reality application is the display of timely and relevant information as an overlay on top of a live view of some scene. For example, it may show street names, restaurant ratings or directional arrows overlaid on the scene captured through a smartphone's camera. Special mobile devices that incorporate cameras and see-through displays in a wearable eyeglasses form factor [11] can be used instead of a smartphone.

AUGREAL uses computer vision to identify actual buildings and landmarks in a scene, and label them precisely in the view [12]. This is akin to an image-based query in Google Goggles [13], but running continuously on a live video stream. AUGREAL extracts a set of features from the scene image, and uses the feature descriptors to find similar-looking entries in a database constructed using features from labeled images of known landmarks and buildings. The database search is kept tractable by spatially indexing the data by geographic locations, and limiting search to a slice of the database relevant to the current GPS coordinates. The prototype application uses a dataset of 1005 labeled images of 200 buildings as the relevant database slice. AUGREAL runs on Microsoft Windows, and makes significant use of

Application	Average request size	Response size
FACE	62 KB	< 60 bytes
SPEECH	243 KB	< 50 bytes
OBJECT	73 KB	< 50 bytes
AUGREAL	26 KB	< 20 bytes
FLUID	16 bytes	25 KB

Figure 1. Average request & response size of each application

	Typical Server		Typical Handheld	
Year	Processor	Speed	Device	Speed
1997	Pentium II	266 MHz	PalmPilot	16 MHz
2002	Itanium	1 GHz	Blackberry 5810	133 MHz
2007	Core 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2011	Xeon X5	32 GHz (2x6 cores)	Samsung Galaxy S2	2.4 GHz (2 cores)

Figure 2. Evolution of Hardware Performance (adapted from Flinn [14])

OpenCV libraries [5], Intel Performance Primitives (IPP) libraries, and multiple processing threads.

*E. Physical Simulation and Rendering (*FLUID*)*

Our final application is used in computer graphics. Using accelerometer readings from a mobile device, it physically models the motion of imaginary fluids with which the user can interact. For example, it can show liquid sloshing around in a container depicted on a smartphone screen, such as a glass of water carried by the user as he walks or runs. The application backend runs a physics simulation, based on the predictive-corrective incompressible smoothed particles hydrodynamics (PCISPH) method [15]. We note that the computational structure of this application is representative of many other interactive applications, particularly "real-time" (i.e., not turn-based) games.

FLUID is implemented as a multithreaded Linux application. To ensure a good interactive experience, the delay between user input and output state change has to be very low, on the order of 100ms. In our experiments, FLUID simulates a 2218 particle system with 20 ms timesteps, generating up to 50 frames per second.

Figure 1 shows average request and response sizes for each application. All applications send requests with input data from the mobile device and receive back computed results based on the inputs. The average request size is tens of kilobyte for a captured image and several hundreds kilobytes for a recorded speech input. The response size is typically less than 100 bytes as the returned results are simple text strings. In FLUID application, however, the requests are streams of sensed motion information using accelerometer data, so each request is just a few bytes. The response data is the state of the simulated world, so unlike the other applications, the responses here are much larger than the requests.

	Dell Latitude 2102	Samsung Galaxy S2
CPU	Intel Atom N550	ARM Cortex-A9
	1.5 GHz per core	1.2 GHz per core
	2 cores (4 threads)	2 cores
RAM	2 GB	1 GB
Storage	320 GB	16 GB
OS	Linux, Windows	Android

Figure 3. Dell Netbook Device Used in Experiments

III. WHY CLOUD RESOURCES ARE NECESSARY

A. Mobile Hardware Performance

Handheld or body-worn mobile devices are always resource-poor relative to server hardware of comparable vintage [16]. Figure 2, adapted from Flinn [14], illustrates the consistent large gap in the processing power of typical server and mobile device hardware over a 15-year period. This stubborn gap reflects a fundamental reality of user preferences: Moore's Law has to be leveraged differently on hardware that people carry or wear for extended periods of time. This is not just a temporary limitation of current mobile hardware technology, but is intrinsic to mobility. The most sought-after features of a mobile device always include light weight, small size, long battery life, comfortable ergonomics, and tolerable heat dissipation. Processor speed, memory size, and disk capacity are secondary.

All the experiments in this paper use a Dell Latitude 2102 as the mobile device. This small netbook machine is more powerful than a typical smartphone today (Figure 3), but it is representative of mobile devices in the near future.

B. Extremes of Resource Demands

At first glance, it may appear that today's smartphones are already powerful enough to support mobile multimedia applications without leveraging cloud resources. Some digital cameras and smartphones support built-in face detection. Android 4.0 APIs support tracking of multiple faces and give detailed information about the location of eyes and mouth [17]. Google's "Voice Actions for Android" performs voice recognition to allow hands-free control of a smartphone [18]. Lowe [19] describes many computer vision applications that run on mobile devices today.

However, upon closer examination, the situation is much more complex and subtle. Consider computer vision, for example. Its computational requirements vary drastically depending on the operational conditions. For example, it is possible to develop (near) frame-rate object recognition

Application	Condition 1	Condition 2	Condition 3
SPEECH	0.057 s	1.04 s	4.08 s
FACE	0.30 s	3.92 s	N/A

Figure 4. Average response time of applications on mobile device under different conditions (see Sect. III-B)

	No Cloud		With Cloud	
Application	median	99%	median	99%
SPEECH	1.22 s	6.69 s	0.23 s	1.25 s
FACE	0.42 s	4.12 s	0.16 s	1.47 s

Figure 5. Response times with and without cloud resources.

(including face recognition [20]) operating on mobile computers *if* we assume restricted operational conditions such as a small number of models (*e.g.*, small number of identities for person recognition), and limited variability in observation conditions (*e.g.*, frontal faces only). The computational demands greatly increase with the generality of the problem formulation. For example, just two simple changes make a huge difference: increasing the number of possible faces from just a few close acquaintances to the entire set of people known to have entered a building, and reducing the constraints on the observation conditions by allowing faces to be at arbitrary viewpoints from the observer.

To illustrate the great variability of execution times possible with perception applications, we perform a set of experiments using two of the applications discussed earlier. We run the SPEECH and FACE applications on the mobile platform, and measure the response times for a wide variety of inputs. Figure 4 shows the results. For the speech application, execution times generally increase with the number of words the algorithm recognizes (correctly or otherwise) in an utterance. Conditions 1, 2, 3 for this application correspond to sentences in which no words, 1–5 words, and 6–22 words are recognized, respectively. The response time varies quite dramatically, by almost 2 orders of magnitude, and is acceptable only when the application fails to recognize any words. When short phrases are correctly recognized, the response time is marginal, at just over 1 second, on average. For longer sentences, when the application works at all, it just takes too long. For comparison, Agus et al. [21] report that human subjects recognize short target phrases within 300 to 450 ms, and are able to tell that a sound is a human voice within a mere 4 ms.

In the case of the face recognition application, the best response times occur when there is a single, large, recognizable face in the image. These correspond to Condition 1 in Figure 4. It fares the worst when it searches in vain at smaller and smaller scales for a face in an image without any faces (Condition 2). Unfortunately, response time is close to the latter for images that only contain small faces. At close to 4-second average response time in these conditions, this application is unacceptably slow. For comparison, recent experimental results on human subjects by Ramon et al. [22] show that recognition times under controlled conditions range from 370 milliseconds for the fastest responses on familiar faces to 620 milliseconds for the slowest response on an unfamiliar face. Lewis et al. [23] report that human subjects take less than 700 milliseconds to determine the absence of faces in a scene, even under hostile conditions such as low lighting and deliberately distorted optics.

Such data-dependent and context-dependent tradeoffs apply across the board to virtually all applications of this genre. In continuous use under the widest possible range of operating conditions, providing near real-time responses, and tuned for very low error rates, these applications have ravenous appetites for processing, memory and energy resources. They can easily overwhelm a mobile device.

C. *Improvement from Cloud Computing*

Performance improves considerably when cloud resources are leveraged. Figure 5 shows the median and 99th percentile response times for the SPEECH and FACE experiments of Figure 4 with and without use of cloud resources. For the speech case, we leverage an Amazon EC2 instance. For the face recognition application, we use a private cloud. Although variability in execution times still exists, the absolute response times are significantly improved. These experiments confirm that leveraging cloud resources can improve user experience for mobile multimedia applications.

IV. EFFECTS OF CLOUD LOCATION

In reality, "the cloud" is an abstraction that maps to services in sparsely scattered data centers across the globe. As a user travels, his mobile device experiences high variability in the end-to-end network latency and bandwidth to these data centers. We examine the significance of this variability for mobile multimedia applications. Response time for remote operations is our primary metric. Energy consumed on the mobile device is a secondary metric. Application-specific metrics such as frame rate are also relevant.

A. *Variable Network Quality to the Cloud*

In this paper, we focus on Amazon EC2 services provided by several data centers worldwide. We use the labels "East," "West," "EU," and "Asia" to refer to the data centers located in Virginia, Oregon, Ireland and Singapore. We measured end-to-end latency and bandwidth to these data centers from a WiFi-connected mobile device located on our campuses in Pittsburgh, PA and Lancaster, UK. We also repeated these measurements from off-campus sites with excellent last-mile connectivity in these two cities. Figure 6 and 7 present our measurements, and quantify our intuition that a traveling user will experience highly variable cloud connectivity. There are also some surprises in the data.

One surprise is the amazingly good connectivity to EC2 East from our Pittsburgh, PA campus. From a wired connection, we measured 8 ms ping times and 200 Mbps transfer rates to this site. Such numbers are more typical of LAN

	Ideal	Measured on campus						Measured off campus					
EC2	Latency	BW to/from Cloud (Mbps)			Latency (ms)			BW to/from Cloud (Mbps)			Latency (ms)		
site	(ms)	Day 1	Day 2	Day 3	min.	median.	90%	Day 1	Day 2	Day 3	min.	median.	90%
East	1.8	28 / 34	42 / 34	20 / 15	8.7	9.2	12.4	5.1 / 13.7	5.1 / 14.2	5.1 / 13.4	13.8	17.9	21.3
West	24.2	12 / 14	20 / 18	11 / 2.5	91.6	92.1	95.5	5.0 / 13.9	5.1 / 13.6	4.9 / 13.4	83.9	90.3	93.8
EU	36.8	3.6 / 0.9	13 / 0.4	7.6 / 0.9	98.3	99.3	103.0	4.9 / 13.8	5.0 / 11.8	4.8 / 13.3	110	112	115
Asia	102.5	10 / 0.5	2.4 / 0.2	3.0 / 0.4	255	265	272	4.6 / 9.4	4.6 / 9.2	4.4 / 9.7	266	277	286

Figure 6. Measured Network Quality to Amazon EC2 Sites from Carnegie Mellon University (Pittsburgh, PA) ("Ideal" is at speed of light)

	Ideal	Measured on campus						Measured off campus					
EC2	Latency	BW to/from Cloud (Mbps)			Latency (ms)			BW to/from Cloud (Mbps)			Latency (ms)		
site	(ms)	Day 1	Day 2	Day 3	min.	median.	90%	Day 1	Day 2	Day 3	min.	median.	90%
East	38.5	4.7 / 5.2	4.7 / 5.2	5.6 / 5.5	86.6	89.4	101	0.8 / 3.3	1.9 / 4.7	0.6 / 3.1	97.3	106	123
West	54.3	5.4 / 3.5	5.4 / 3.5	3.6 / 3.6	155	159	208	0.5 / 2.4	1.4 / 2.8	0.7 / 2.6	165	182	201
EU	1.7	6.7 / 10.4	6.7 / 10.4	8.0 / 10.5	16.2	32.7	63.4	1.7 / 9.4	2.5 / 14.5	1.4 / 7.6	31	43.6	64
Asia	73.2	4.7 / 2.6	4.7 / 2.6	6.2 / 2.7	273	279	325	0.3 / 1.6	1.4 / 1.9	0.5 / 1.6	259	272	291

Figure 7. Measured Network Quality to Amazon EC2 Sites from Lancaster University (Lancaster, UK) ("Ideal" is at speed of light)

connections than WAN transfers! We believe that this is due to particularly favorable network routing between our campus and the EC2 East site. This hypothesis is confirmed by the poorer off-campus measurements shown in Figure 6. Thus, our EC2 East on-campus results best serve to indicate what one can expect from a LAN-connected private cloud. Li et al. [24] report that average round trip time (RTT) from 260 global vantage points to their optimal Amazon EC2 instances is 73.68 ms. Therefore, the EC2 West numbers in Figure 6 are more typical of cloud connectivity.

Another surprise is the great range of bandwidths observed, particularly the upload/download asymmetry and the significant variation between experiments. To mitigate this time-varying factor, we scheduled our experiments on weekday nights when conditions were stable and bandwidth consistently high. All experiments in the rest of the paper were run under these conditions on campus in Pittsburgh.

B. Impact on Response Time

We next evaluate how cloud connectivity affects the applications described in Section II. We consider six cases. The first, labeled "Mobile," runs the application entirely on the mobile device. Cloud connectivity is irrelevant, but the resource constraints of the mobile device dominate. In four other cases, the mobile device performs the resource-intensive part of each operation on one of the four Amazon data centers and blocks until it receives the result.

	Mobile	1WiFi	Cloud (East, West, EU, Asia)
CPU	Atom E5320 1.5 GHz 2 cores, 4 threads	Xeon N550 1.86 GHz 4 cores	X-Large Instance 20 Compute Units 8 virtual cores
RAM	2 GB	4 GB	7 GB
VMM	none	KVM	Xen,VMware

Figure 8. Platform specifications

The sixth case, labeled "1WiFi," corresponds to the theoretical best-case for data center location. With today's deployed wireless technology, this is exactly one WiFi hop away from a mobile device. This can only be approximated today in special situations: e.g., on a WiFi-covered campus, with access points connected to a private data center through a lightly-loaded gigabit LAN backbone. If naively implemented at global scale, 1WiFi would lead to a proliferation of data centers. Section V discusses how the consolidation benefits of cloud computing can be preserved while scaling out the 1WiFi configuration.

Figure 8 compares the characteristics of the compute platforms used in our configurations. For 1WiFi, we create a minimal data center using a six-year old WiFi-connected server. The choice of this near-obsolete machine is deliberate. By comparing it against a fast mobile device and fast EC2 cloud instances, we have deliberately stacked the deck against 1WiFi. Hence, any wins by this strategy in our experiments should be considered quite meaningful.

FACE*:* Figure 9 summarizes the response times measured for FACE under different conditions. Here, we test with 300 images that may have known faces, unknown faces, or no faces at all. Processing on the mobile device alone can provide tolerable response times for the easier images, but is crushed by the heavy-tailed distribution of processing costs. Only 1WiFi can provide fast response (<200ms) most of the time, and a tolerable worst case response time. Hence, 1WiFi is the best approach to running FACE.

SPEECH*:* Results for SPEECH are somewhat different (Figure 10). Here, the application generally requires significant processing for each query, and data transfer costs are modest. This changes the relative performance of the

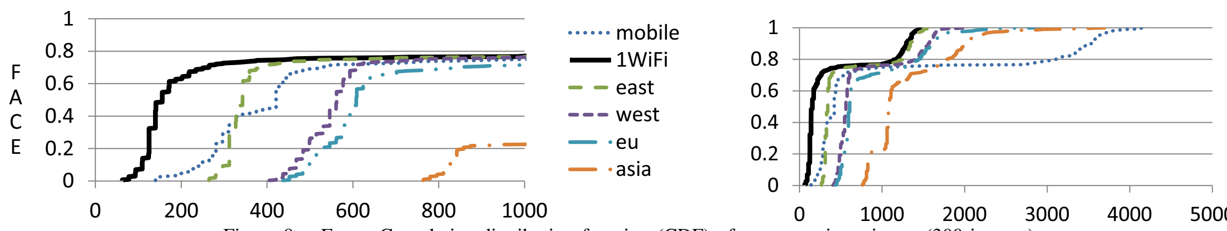


Figure 9. FACE: Cumulative distribution function (CDF) of response times in ms (300 images).

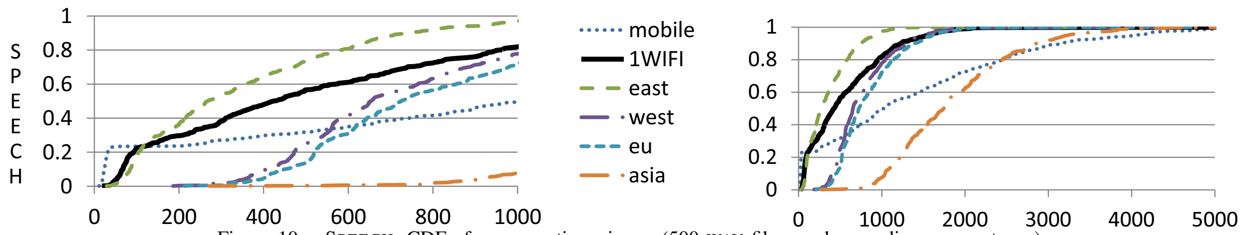


Figure 10. SPEECH: CDF of response times in ms (500 WAV files, each recording one sentence).

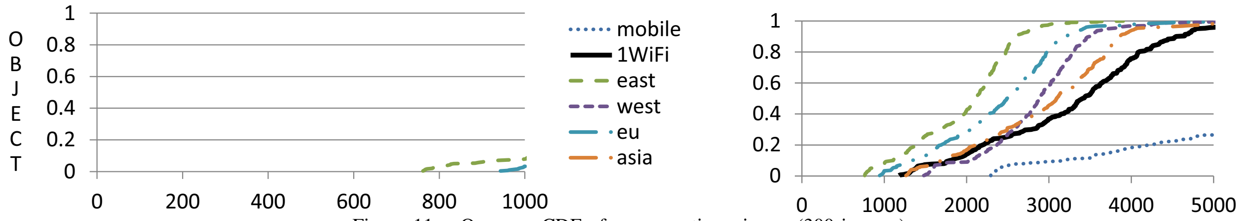


Figure 11. OBJECT: CDF of response times in ms (300 images).

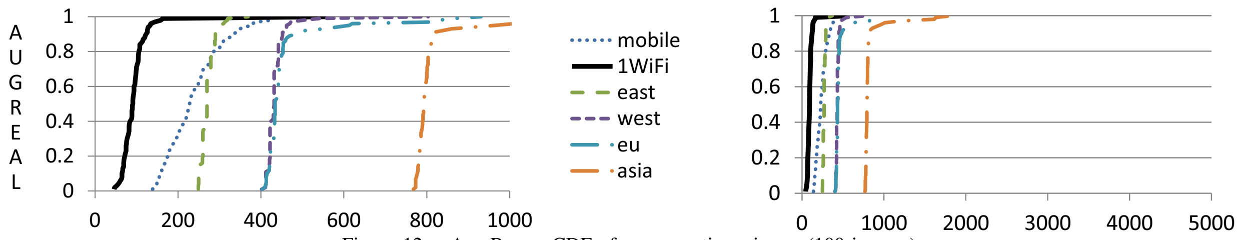


Figure 12. AUGREAL: CDF of response times in ms (100 images).

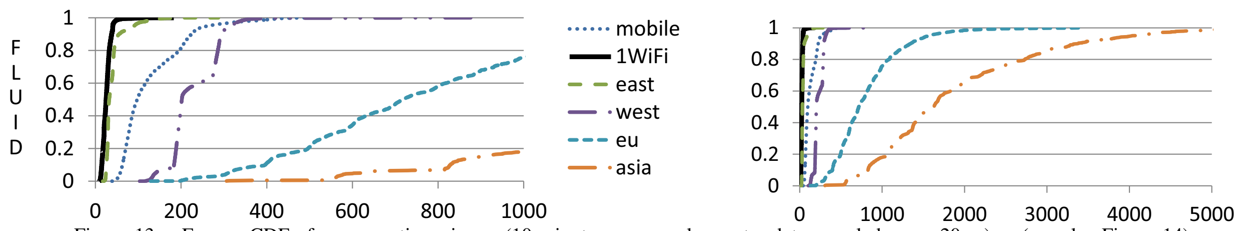


Figure 13. FLUID: CDF of response times in ms (10 minute runs, accelerometer data sampled every 20ms). (see also Figure 14)

	Normalized Simulation Speed	Displayed Frame Rate (FPS)
mobile	0.2	9.2
1WiFi	1.0	49.8
east	1.0	42.8
west	1.0	10.3
eu	1.0	3.6
asia	1.0	1.6

Figure 14. Simulation speed, frame rate for FLUID.

		Mobile	1WiFi	East	West	EU	Asia
FACE	(W)	14.8	12.6	11.4	10.9	11.0	11.0
	(J/query)	16.4	5.4	6.6	8.5	9.5	14.3
SPEECH	(W)	16.1	14.5	14.5	14.4	14.4	14.4
	(J/query)	22.5	8.2	5.3	11.2	12.2	26.9
OBJECT	(W)	16.8	14.5	14.5	14.6	14.5	14.5
	(J/query)	107.0	48.2	28.9	41.5	35.3	43.8
AUGREAL	(W)	13.9	11.7	11.3	11.7	11.3	11.3
	(J/query)	3.3	1.1	3.1	5.1	5.2	9.4
FLUID	(W)	17.0	15.8	15.9	15.8	15.8	15.7
	(J/frame)	1.9	0.3	0.4	1.8	4.6	10.7

Figure 15. Energy consumption on mobile device

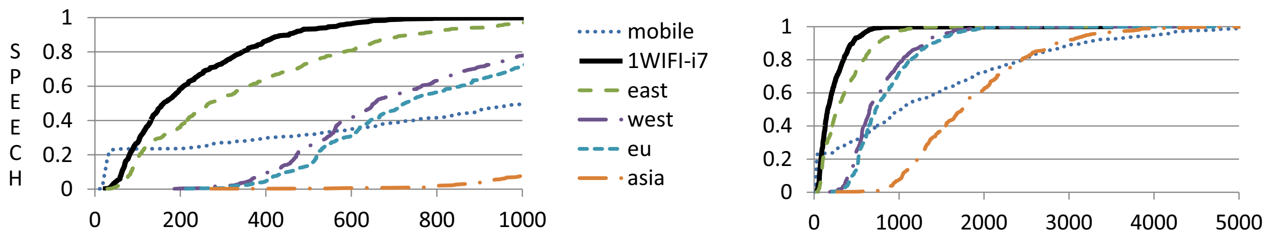


Figure 16. Experiments of Figure 10 repeated with faster 1WiFi machine

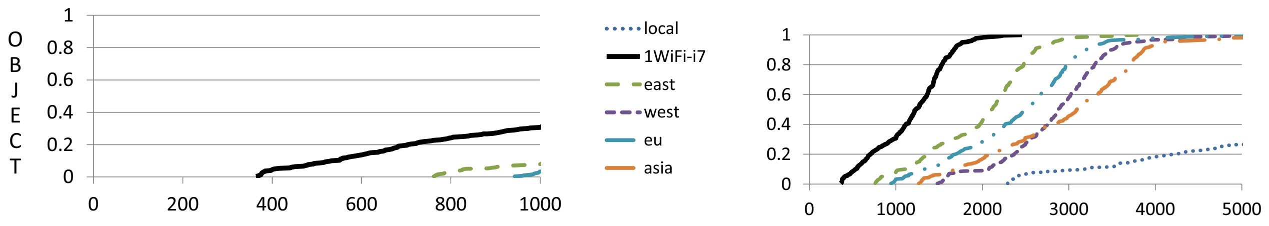


Figure 17. Experiments of Figure 11 repeated with faster 1WiFi machine

strategies significantly. As the response time is dominated by processing time, this favors the more capable but distant servers in the cloud over the weak 1WiFi server. Processing without cloud assistance is out of the question. For SPEECH, using the closest EC2 data center is the winning strategy. To understand the effect of a more powerful 1WiFi machine, we repeated that experiment with an Intel i-3770 desktop. The results shown in Figure 16 confirm that 1WiFi now dominates the alternatives.

OBJECT*:* Compared to the previous two applications, OBJECT requires significantly greater compute resources. Unfortunately, the processing load is so large that none of the approaches yield acceptable interactive response times (Figure 11). This application really needs more resources than our single VM instances or weak 1WiFi server can provide. To bring response times down to reasonable levels for interactive use, we will either need to parallelize the application beyond a single machine/VM boundary and employ a processing cluster, or make use of GPU hardware to accelerate critical routines. Both of these potential solutions are beyond the scope of this paper. Using the faster 1WiFi machine (Intel i-3770) does help significantly (Figure 17).

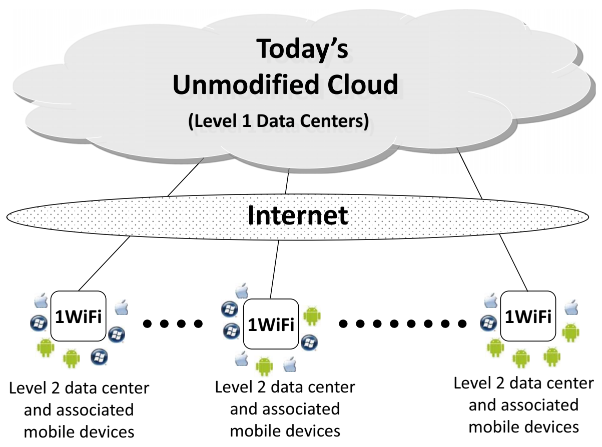
AUGREAL*:* This application employs a low-cost feature extraction algorithm, and an efficient approximate nearest-neighbor algorithm to match features in its database. While these processing costs are modest, data transfer costs are high because of image transmission. Therefore, as shown in Figure 12, none of the EC2 cases is adequate for this application. They generally provide slower response times than execution on the mobile device. 1WiFi, on the other hand, works extremely well for this application, providing very fast response times (around 100ms) needed for crisp interactions. This is clearly the winning strategy for AUGREAL.

FLUID*:* Response time for FLUID is defined as the time between the sensing of a user action (i.e., accelerometer reading), to when that input is reflected in the output. This largely reflects three factors: the execution time of a simulation step, network latency, and data transfer time for a frame from the simulation thread. As seen in Figure 13, local execution on the mobile device produces good response times, since all but the first factor are essentially zero. However, simulation speed and frame rate also need to be considered (Figure 14). The simulation runs asynchronously to the inputs and display, and tries to match simulated time with wall-clock time. Since the mobile device cannot execute the simulation steps fast enough, fluid motions are less than one fifth of realistic speeds. The cloud strategies do not have this issue, but due to bandwidth and network latencies, cannot deliver the results of the simulation fast enough to sustain the full frame rate. Only 1WiFi and East can deliver both good responsiveness and high frame rates.

C. Impact on Energy Usage

Battery life is a key attribute of a mobile device. Executing resource-intensive operations in the cloud can greatly reduce the energy consumed on the mobile device by the processor(s), memory and storage. However, it increases network use and wireless energy consumption. Since peak processor power consumption exceeds wireless power consumption on today's high-end mobile devices, this tradeoff favors cloud processing as computational demands increase. Network latency has recently been shown to increase energy consumption for remote execution by as much as 50%, even if bandwidth and computation are held constant [25], [14]. This is because hardware elements of the mobile device remain in higher-power states for longer periods of time.

Figure 15 summarizes energy consumption on our mobile device for the experiments described in Section IV-B. For each application, the first row shows the power dissipation in watts, averaged over the whole experiment. In all cases,


Today's
Unmodified Cloud
(Level 1 Data Centers)
Internet
1WiFi
1WiFi
1WiFi
Level 2 data center and associated mobile devices
Level 2 data center and associated mobile devices
Level 2 data center and associated mobile devices
Figure 18. Two-level Hierarchical Cloud Architecture

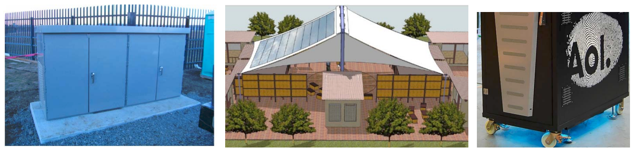
this quantity shows little variation across data centers. Local execution on the mobile device incurs the highest power dissipation. Note that the netbook platform has a high baseline idle power dissipation (around 10W), so the relative improvement in power is likely to be larger on more energy-efficient hardware.

Average power dissipation only tells part of the story. Cloud use also tends to shorten the time to obtain a result. When this is factored in, the energy consumed per query or frame is dramatically improved. These results are shown in the second row for each application in Figure 15. In the best case, the energy consumed per result is reduced by a factor of 3 to 6. The strategies that exhibit the greatest energy efficiency are also the ones that give the best response times.

D. Summary and Discussion

The results of Sections IV-B and IV-C confirm that *logical proximity* to data center is essential for mobile applications that are highly interactive and resource intensive. By "logical proximity" we mean the end-to-end properties of high bandwidth, low latency and low jitter. Physical proximity is only weakly correlated with logical proximity because of the well-known "last mile" problem [26].

1WiFi represents the best attainable logical proximity. Our results show that this extreme case is indeed valuable for many of the applications studied, both in terms of response time and energy efficiency. It is important to keep in mind that these are representative of a new genre of cognitive assistance applications that are inspired by the sensing and user interaction capabilities of mobile devices. Mobile participation in server-based multiplayer games such as Doom 3 is another use case that can benefit from logical proximity [27]. The emergence of such applications can be accelerated by deploying infrastructure that assures mobile users of continuous logical proximity to the cloud. The situation is analogous to the dawn of personal computing, when the dramatic lowering of user interaction latency relative to time-sharing led to entirely new application metaphors such as the spreadsheet and the WYSIWYG editor.


(a) Outdoor (b) Solar Powered (c) Indoor
Figure 19. Unattended Micro Data Centers (Sources: [28], [29])

V. Scaling Out 1WiFi

We thus face contradictory requirements. On the one hand, the 1WiFi property is valuable for mobile computing. On the other hand, it works against consolidation because there have to be many data centers at the edges of the Internet to ensure 1WiFi cloud access everywhere. Consolidation is the essence of cloud computing because dispersion induces diseconomies of scale: the marginal cost of administering machines in a centralized data center is typically lower than when they are spread over smaller data centers. How can we reconcile these contradictory requirements?

A. Concept

We assert that the only practical solution to this problem is a hierarchical organization of data centers, as shown in Figure 18. Level 1 of this hierarchy is today's unmodified cloud infrastructure such as Amazon's EC2 data centers. Level 2 consists of *stateless data centers* at the edges of the Internet, servicing currently-associated mobile devices.

We envision an appliance-like deployment model for Level 2 data centers. They are not actively managed after installation. Instead, soft state (such as virtual machine images and files from a distributed file system) is cached on their local storage from one or more Level 1 data centers. It is the absence of hard state at Level 2 that keeps management overhead low. Consolidation or reconfiguration of Level 1 data centers does not affect the 1WiFi property at Level 2. Adding a new Level 2 data center or replacing an existing one only requires modest setup and configuration. Once configured, a Level 2 data center can dynamically self-provision from Level 1 data centers.

Physical motion of a mobile device may take it far from the Level 2 data center with which it is currently associated. Beyond a certain distance, the 1WiFi property may no longer hold. In that case, a mechanism similar to wireless access point handoff can be executed to seamlessly switch association to a different Level 2 data center.

B. Physical Realization

The hardware technology for Level 2 data centers is already here today for reasons unrelated to mobile computing. For example, Myoonet has pioneered the concept of *micro data centers* for use in developing countries (Figure 19(a) and (b)) [28]. AOL has recently introduced indoor micro-data centers for enterprises (Figure 19(c)) [29]. Today, these micro data centers are being used as Level 1 data centers in private clouds. By removing hard state and adding self-provisioning, they can be repurposed as Level 2 data centers. In the future, one can envision optimized Level 2 data centers for 1WiFi. For example, with modest engineering effort, a WiFi access point could be transformed into a "nano," "pico," or "femto" Level 2 data center by adding processing, memory and storage.

While much innovation and evolution will undoubtedly occur in the form factors and configurations of future Level 2 data centers, we can identify four key attributes that any such implementation must possess:

- *Only soft state:* It does not have any hard state, but only cached state from Level 1. It may also buffer data from a mobile device en route to Level 1.
- *Powerful, well-connected and safe:* It is powerful enough to handle resource-intensive applications from multiple associated mobile devices. Bandwidth between Level 1 and Level 2 is good, typical WAN latency is acceptable, and network failures are rare. Battery life is not a concern. It can be trusted as a computing platform.
- *Close at hand:* It is easily deployable within one Wi-Fi hop of associated mobile devices.
- *Builds on standard cloud technology:* It leverages and reuses Level 1 software infrastructure and standards (e.g. OpenStack [30]) as much as possible.

C. Operating Environment

There is significant overlap in the requirements specifications for Levels 1 and 2. At both levels, there is the need for: (a) strong isolation between untrusted user-level computations; (b) mechanisms for authentication, access control, and metering; (c) dynamic resource allocation for user-level computations; and, (d) the ability to support a very wide range of user-level computations, with minimal restrictions on their process structure, programming languages or operating systems. At Level 1, these requirements are met today using the virtual machine (VM) abstraction. For precisely the same reasons they are so valuable at Level 1, we foresee VMs as central to Level 2.

A rich ecosystem of VM-based mechanisms, policies and practices already exists for Level 1, but some changes may be needed for Level 2. For example, cooling and power are major concerns at Level 1 but are less important important at Level 2 because data centers are much smaller and ease of deployment is the dominant concern.

Trust is a differentiator between the two levels. A Level 1 data center is effectively a small fort, with careful attention paid to physical security of the perimeter. Tampering of hardware within Level 1 is assumed to be impossible. Mechanisms such as TPM-based attestation are therefore not often used at this level. In contrast, a Level 2 data center has weak perimeter security even if it is located in a locked closet or above the ceiling. Hence, tamper-resistant and tamper-evident enclosures, remote surveillance, and TPM-based attestation will all be more important at Level 2.

The speed of provisioning is another major differentiator between Level 1 and Level 2. Today, Level 1 data centers are optimized for launching VM images that already exist in their storage tier. They do not provide fast options for instantiating a new custom image. One must either launch an existing image and laboriously modify it, or suffer the long, tedious upload of the custom image over a WAN. In contrast, Level 2 data centers need to be much more agile in their provisioning. Their association with mobile devices is highly dynamic, with considerable churn due to user mobility. A user from far away may unexpectedly show up at a Level 2 data center (e.g., if he just got off an international flight) and try to use it for an application such as a personalized language translator. For that user, the provisioning delay before he is able to use the application impacts usability.

We see at least three different approaches to rapid provisioning at Level 2. One approach is to exploit higher-level hints of user mobility (e.g., derived from online schedules, travel information, real-time tracking, explicit user guidance, etc.) to pre-provision Level 2 data centers. A second approach is to launch a VM instance at Level 2 without provisioning delay, and then demand page the VM state as execution proceeds. This reduces startup delay at the cost of unpredictable delays during execution. The feasibility of this approach has been shown in the Internet Suspend/Resume system [31] and other similar systems. A third approach is to synthesize the desired VM state from a pre-cached base VM and a relatively small dynamically-transmitted overlay [32], [33], [34]. Exploring the tradeoffs and optimizations in this space will be important future research, but the feasibility of dynamic provisioning is not in doubt.

Unique to Level 2 is the problem of dynamic discovery by mobile clients, as a precursor to association. One approach is manual selection, using a mechanism similar to what is already in use today for choosing WiFi networks based on their SSIDs. More sophisticated solutions could also be built, leveraging existing low-level service discovery mechanisms such as UPnP, Bluetooth Service Discovery, Avahi, and Jini.

VI. Discussion

A. When Level 1 is Unreachable

The hierarchical organization of Figure 18 was derived solely from considerations of performance and consolidation. As a bonus, it also improves availability. Once a Level 2 data center has been provisioned for an associated mobile device, WAN network failures or Level 1 data center failures are no longer disruptive. This achieves disconnected operation, a concept originally developed for distributed file systems [35]. Simanta et al [36], [37] explore the tradeoffs between performance and availability in the methods used to dynamically provision Level 2 from Level 1.

The improved availability of the two-level architecture applies even to mobile applications that are not latency-sensitive. Any mobile application that uses the cloud for remote execution can benefit. Although not widely discussed today, the economic advantages of data center consolidation come at the cost of reduced autonomy and vulnerability to cloud failure. These are not hypothetical worries, as shown by the day-long outage of Siri in 2011 [38], [39], the multi-hour weather-related outage of Amazon's data center in Virginia in June 2012 [40], and the extended Christmas Eve 2012 outage of Netflix's video streaming service due to an Amazon failure [41]. As users become reliant on mobile multimedia applications, they will face inconvenience and frustration when a cloud service for a critical application is unavailable. These concerns are especially significant in domains such as military operations and disaster recovery.

B. Data Placement

Although we have mainly considered the computational aspects of a distributed offload infrastructure, appropriately handling data placement can be important for many applications. If an application requires a relatively small data set for its operation, then the application VM can wholly contain the needed data, and we can trivially migrate and offload the executable and data as a single unit. At the other extreme, for an application that uses a very large data set in an unpredictable manner, there is little one can do to effectively place data. Rather, one must fetch data as needed from Level 1 datacenters, or if this is too expensive, restrict the application to running in the central datacenter. Most applications will likely fall between these extremes – they may have substantial data sets, but will typically use only a subset, and exhibit locality in access. In many cases, we can predict the likely data required from context, and hoard [35] this data at the Level 2 datacenters. For example, in a map application, physical location is highly correlated with accessed map data, so the geographical region around the Level 2 data center can be locally cached. Similarly, for a face recognition database, the faces of people who reside in the local region are likely to be most important. Automatic caching capabilities of distributed file systems like the Coda File System [42] or the OceanStore project [43] can exploit this locality, mediate the distribution and modification of data between Level 1 and Level 2 data centers, as well as provide resiliency in face of failures. Unfortunately, most cloud-sourced data repositories today (e.g., GoogleMaps, Flickr, etc.) do not provide a distributed file system interface. For these, we can imagine a solution that instantiates local proxies on the Level 2 datacenters that provide intelligent caching of data from these repositories. Applications and mobile devices associated with a Level 2 data center can direct their requests to these proxies rather than directly to the cloud to benefit from local data hoarding.

VII. Related Work

This is the first work to rigorously explore the impact of mobile multimedia applications on data center consolidation. The concept of "cyber foraging" by mobile devices (i.e., leveraging nearby resources) was first articulated in 2001 [44]. Flinn [14] describes the extensive work on this topic since then. A 2009 position paper [33] argued that end-to-end latency was the critical determinant of whether public clouds were adequate for deeply immersive applications. It introduced the concept of "cloudlets," which correspond to Level 2 data centers in this paper. However, that work offered no experimental evidence to support its claim. While cloudlets were shown to be *sufficient* by construction, they were not shown to be *necessary.* One way to view this paper is that it provides the empirical evidence that cloudlets are not a luxury, but indeed a necessity in the face of real world network connectivity to public cloud infrastructure.

Recent work by others corroborates the conclusions of this paper. Clinch et al. [45] explore the need for logical proximity when using a large static display from a mobile device. Their results are consistent with the findings reported here. Soyata et al. [46] use Monte Carlo simulation to explore how a face recognition algorithm should be partitioned across multiple back-end computation engines. They conclude that a cloudlet-based strategy is optimal, and provide experimental validation.

VIII. Conclusion

The convergence of mobile computing and cloud computing enables new multimedia applications that are both resource-intensive and interaction-intensive. For these applications, end-to-end network bandwidth and latency matter greatly when cloud resources are used to augment the computational power and battery life of a mobile device. In this paper, we have presented quantitative evidence that latency considerations limit data center consolidation. We have shown how this challenge can be addressed by a two-level hierarchical structure that seamlessly extends today's cloud computing infrastructure.

IX. Acknowledgements

This research was supported by the National Science Foundation (NSF) under grant numbers CNS-0833882 and IIS-1065336, by an Intel Science and Technology Center grant, and by the Department of Defense (DoD) under Contract No. FA8721-05-C-0003 for the operation of the Software Engineering Institute (SEI), a federally funded research and development center. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily represent the views of the NSF, Intel, DoD, SEI, or Carnegie Mellon University. This material has been approved for public release and unlimited distribution except as restricted by copyright.

References

[1] Apple, "iPhone 4S - Ask Siri to help you get things done," http://www.apple.com/iphone/features/siri.html.

[2] C. Thompson, "What is I.B.M.'s Watson?" *New York Times Magazine*, June 2011, http://www.nytimes.com/2010/06/20/magazine/20Computer-t.html.

[3] P. Viola and M. Jones, "Robust Real-time Object Detection," in *International Journal of Computer Vision*, 2001.
[4] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
[5] OpenCV, "OpenCV Wiki," http://opencv.willowgarage.com/wiki/.
[6] Jibbigo, "Jibbigo Voice Translator Apps," http://www.jibbigo.com.
[7] Vlingo, "Voice to Text Applications Powered by Intelligent Voice Recognition," http://www.vlingo.com.
[8] Sphinx-4, "Sphinx-4: A Speech Recognizer Written Entirely in the Java Programming Language," http://cmusphinx.sourceforge.net/sphinx4/.
[9] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. M. Vandeweghe, "Herb: a home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, January 2010.
[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
[11] Gizmag, "Lumus glasses let you watch video, and the real world," 2012, http://www.gizmag.com/lumus-see-through-video-glasses/20840/.
[12] G. Takacs, M. E. Choubassi, Y. Wu, and I. Kozintsev, "3D mobile augmented reality in urban scenes," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, July 2011.
[13] Google, "Google Goggles," http://http://www.google.com/mobile/goggles.
[14] J. Flinn, *Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload*. Morgan & Claypool Publishers, 2012.
[15] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.
[16] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, Ottawa, Canada, 1996.
[17] R. Paul, "First look: Android 4.0 SDK opens up face recognition APIs," October 2010, http://arstechnica.com/gadgets/news/2011/10/first-look-android-40-sdk-opens-up-face\-recognition-apis.ars.
[18] Google, "Voice Actions for Android," 2011, http://www.google.com/mobile/voice-actions/.
[19] D. Lowe, "The Computer Vision Industry ," 2010, http://people.cs.ubc.ca/~lowe/vision.html.
[20] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe, "FRVT 2006 and ICE 2006 Large-Scale Results," National Institute of Standards and Technology, Tech. Rep. NISTIR 7408, March 2007.
[21] T. Agus, C. Suied, S. Thorpe, and D. Pressnitzer, "Characteristics of human voice processing," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, June 2010.
[22] M. Ramon, S. Caharel, and B. Rossion, "The speed of recognition of personally familiar faces," *Perception*, vol. 40, no. 4, pp. 437–449, 2011.
[23] M. B. Lewis and A. J. Edmonds, "Face detection: Mapping human performance," *Perception*, vol. 32, pp. 903–920, 2003.
[24] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th annual conference on Internet measurement*. ACM, 2010, pp. 1–14.
[25] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, June 2010.
[26] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescape, "Broadband Internet Performance: A View From the Gateway," in *Proceedings of ACM SIGCOMM 2011*, Toronto, ON, August 2011.
[27] S. K. Barker and P. Shenoy, "Empirical Evaluation of Latency-sensitive Application Performance in the Cloud," in *Proceedings of ACM Multimedia Systems*, Phoenix, AZ, February 2010.
[28] Myoonet, "Unique Scalable Data Centers," December 2011, http://www.myoonet.com/unique.html.
[29] R. Miller, "AOL Brings Micro Data Center Indoors, Adds Wheels," http://www.datacenterknowledge.com/archives/2012/08/13/aol-brings-micro-data-center-indoors-adds-wheels, August 2012.
[30] O. Stack, "Open source software for building private and public cloud," January 2013, http://www.openstack.org/.
[31] M. Kozuch, M. Satyanarayanan, T. Bressoud, and Y. Ke, "Efficient State Transfer for Internet Suspend/Resume," Intel Research Pittburgh, Tech. Rep. IRP-TR-02-03, May 2002.
[32] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient Customization of Mobile Computing Infrastructure," in *Proc. of the MobiVirt 2008 Workshop on Virtualization in Mobile Computing*, Breckenridge, CO, June 2008.
[33] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, Oct-Dec 2009.
[34] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-Time Provisioning for Cyber Foraging," CMU School of Computer Science, Tech. Rep. CMU-CS-12-148, December 2012.
[35] J. J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System," *ACM Transactions on Computer Systems*, vol. 10, no. 1, February 1992.
[36] S. Simanta, G. Lewis, E. Morris, K. Ha, and M. Satyanarayanan, "A Reference Architecture for Mobile Code Offload in Hostile Environments," in *Proceedings of WICSA/EICSA 2012: Joint 10th Working IEEE/IFIP Conference on Software Architecture and 6th European Conference on Software Architecture*, Helsinki, Finland, August 2012.
[37] S. Simanta, K. Ha, G. Lewis, E. Morris, and M. Satyanarayanan, "A Reference Architecture for Mobile Code Offload in Hostile Environments," in *Fourth International Conference on Mobile Computing, Applications and Services*, Seattle, WA, October 2012.
[38] S. J. Purewal, "Siri Goes Down For a Day; Apple Says Network Outages Are Possible," *PCWorld*, November 2011.
[39] A. Savvas, "Firms will flee cloud if lessons from Siri and RIM outages not learned," *CFO World*, November 2011.
[40] R. McMillan, "(Real) Storm Crushes Amazon Cloud, Knocks out Netflix, Pinterest, Instagram," *Wired*, June 2012.
[41] D. Kucera, "Amazon apologizes for Christmas Eve outage affecting Netflix," December 2012, http://articles.washingtonpost.com/2012-12-31/business/36103607_1_amazon-web-services-christmas-eve-outage-largest-online-retailer.
[42] M. Satyanarayanan, "The Evolution of Coda," *ACM Transactions on Computer Systems*, vol. 20, no. 2, May 2002.
[43] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: the oceanstore prototype," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, 2003, pp. 1–14.
[44] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, vol. 8, August 2001.
[45] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How Close is Close Enough? Understanding the Role of Cloudlets in Supporting Display Appropriation by Mobile Users," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom 2012)*, Lugano, Switzerland, March 2012.
[46] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-Vision: Real-Time face recognition using a Mobile-Cloudlet-Cloud acceleration architecture," in *17th IEEE Symp. on Computers and Communications*, Cappadocia, Turkey, July 2012.