The Case for VM-based Cloudlets in Mobile Computing

Mahadev Satyanarayanan[†], Paramvir Bahl[‡], Ramon Caceres[•], Nigel Davies[°]

[†]Carnegie Mellon University, [‡]Microsoft Research, [•]AT&T Research, [°]Lancaster University

1 Introduction

Mobile computing is at a fork in the road. After two decades of sustained effort by many researchers, we have developed the core concepts, techniques and mechanisms to provide a solid foundation for this still fast-growing area. The vision of "information at my fingertips at any time and place" was only a dream in the mid-1990s. Today, ubiquitous email and Web access is a reality that is experienced by millions of users worldwide through their BlackBerries, iPhones, Windows Mobile, and other mobile devices. Continuing on this road, mobile Web-based services and location-aware advertising opportunities have begun to appear. Large investments are being made in anticipation of major profits.

Yet, by staying on this path, mobile computing ignores its true potential. Awaiting discovery is an entirely new world in which *mobile computing seamlessly augments the cognitive abilities of users* using compute-intensive capabilities such as speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality, planning and decision-making. By thus empowering mobile users, we could transform many areas of human activity. The sidebar speculates on one example of such a transformation.

This paper discusses the technical obstacles to this transformation, and proposes a new system architecture to overcome them. In this architecture, a mobile user exploits virtual machine (VM) technology to rapidly instantiate customized service software on a nearby *cloudlet,* and then uses that service over a wireless LAN. The mobile device typically functions as a thin client with respect to the service. A cloudlet is a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices.

Our strategy of leveraging transiently-customized proximate infrastructure as a mobile device moves with its user through the physical world is called *cloudlet-based resource-rich mobile computing.* Crisp interactive response, which is essential for seamless augmentation of human cognition, is easily achieved in this architecture because of the physical proximity and one-hop network latency of the cloudlet. Using a cloudlet also simplifies meeting the pe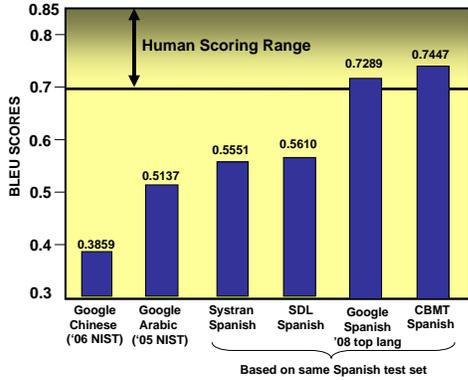ak bandwidth demand of multiple users interactively generating and receiving media such as HD video and high-resolution images. Rapid customization of infrastructure for diverse applications emerges as a critical requirement of this architecture. We present results from a proof-of-concept prototype that suggest that this requirement can indeed be met through VM technology.

We begin in Section 2 with a discussion of resource limitations in mobile hardware, and then explain in Section 3 why cloud computing is limited in its ability to address these limitations. Our proposed solution is presented in Sections 4 and 5, and a proof-of-concept prototype is described in Section 6. We conclude with a summary of the main points in Section 7.

2 Resource-Poor Mobile Hardware

The phrase "resource-rich mobile computing" appears to be an oxymoron at first glance. It has long been recognized that mobile hardware is necessarily resource-poor relative to static client and server hardware [15]. At any given cost and level of technology, considerations of weight, size, battery life, ergonomics, and heat dissipation exact a severe penalty in computational resources such as processor speed, memory size, and disk capacity. From the viewpoint of a user, a mobile device can never be too small, too light or have too long a battery life. While mobile hardware continues to evolve and improve, it will always be resource-poor relative to static hardware. On hardware that people carry or wear for extended periods of time, improving size, weight and battery life are higher priorities than enhancing compute power. This is not just a temporary limitation of current mobile hardware technology, but is intrinsic to mobility. Computation on mobile devices will always be a compromise.

Resource poverty is a major obstacle for many applications with the potential to seamlessly augment human cognition. These applications typically require processing and energy that far outstrips the capabilities of mobile hardware. In the lab, with ample computing resources, the state of the art for applications such as face recognition, speech recognition, and language translation is near-human in performance and quality. For example, as shown in Figure 1(a), Spanish-English translation comparable to human quality was achieved in 2006 on a 100-node computing engine using large online corpora

(a) Machine Translation Quality (Carbonell et al [5])

Year	Computer worse than human	Computer better than human	Indeter- minate	$\frac{Worse}{Better}$
1999	87.5%	4.2%	8.3%	21.0
2001	87.5%	8.3%	4.2%	10.5
2003	45.8%	16.7%	37.5%	2.75
2005	37.5%	33.3%	29.2%	1.13
2006	29.2%	37.5%	33.3%	0.78

(b) Face Recognition Quality (Adler and Schuckers [1])

Figure 1: Near-Human Quality of Cognitive Augmentation Applications Today

and a context-based machine translation (CBMT) algorithm [5]. For the IBM BLEU metric used in this figure, scores above 0.7 enter the bilingual human translator range, and those above 0.8 approach the human experienced-professional translator range. Face recognition using computer vision is another area where rapid progress has occurred over the past decade. Figure 1(b), adapted from Adler and Schucker's 2007 comparison of human and automatic face recognition performance [1], shows that computers and humans are comparable on this task today. While several technical improvements for practical deployment are still needed in such applications, no giant leaps of faith are needed to recognize their future potential. The real challenge lies in sustaining their state-of-the-art performance and quality in the wild: under highly variable conditions on lightweight, energy-efficient, resource-impoverished mobile hardware.

3 The Limits of Cloud Computing

An obvious solution to the resource poverty of mobile devices is to leverage *cloud computing*. A mobile device could execute a resource-intensive application on a distant high-performance compute server or compute cluster, and support thin client user interactions with the application over the Internet. Unfortunately, as discussed below, long WAN latencies are a fundamental obstacle.
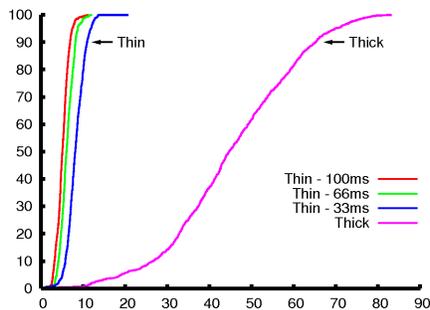
3.1 Why Latency Hurts

WAN delays in the critical path of user interaction can hurt usability by degrading the crispness of system response. Even trivial user-application interactions incur these delays in cloud computing. Humans are acutely sensitive to delay and jitter, and it is very difficult to control these parameters at WAN scale. As latency increases, interactive response suffers. Loosely-coupled tasks such as Web browsing may continue to be usable, but deeply immersive tasks such as augmented reality become jerky or sluggish to the point of distraction. This reduces the user's depth of cognitive engagement.

Lagar-Cavilla et al [10] have shown that latency can negatively impact interactive response in spite of adequate bandwidth. Figure 2(a) compares the measured output frame rate of a visualization application (Quake-Viz) under two different configurations: local machine with hardware graphics acceleration ("Thick"), and remote compute server over a 100 Mb/s network with the output viewed through VNC ("Thin"). A high frame rate provides the illusion of smoothness to an interactive user. Figure 2(a) shows that even a modest latency of 33 ms causes frame rate to drop considerably from the frame rate experienced with a thick client. The VNC protocol strives to keep up by dropping frames, resulting in jerky interaction. Work-conserving thin client protocols, such as X windows, preserve frames but offer sluggish interaction. In both cases, the user experience is considerably poorer than for local interaction. Figure 2(b) reports measured Internet2 latencies between representative end points at planetary scale [10]. The measured figures far exceed the speed-of-light lower bound in the last column.

Independently, Tolia et al [18] have shown that the user-perceived quality of thin client performance is highly variable, and depends on both the degree of interactivity of the application and the end-to-end latency of the network. As Figure 3 illustrates, the usability of a highly interactive task such as photo editing suffers unacceptably even at moderate network latency (100 milliseconds RTT) and very good bandwidth (100 Mbps). This in contrast to tasks such as Web browsing, which are interactively undemanding. Figure 3(b) shows the distribution of response times for individual interactions in a `GIMP` photo editing task. The mapping of response times to subjective impressions of quality shown in Figure 3(a) is based on long-established HCI guidelines that were developed through empirical studies.

3.2 WAN Latency Is Unlikely to Improve

The current trajectory of Internet evolution makes it very unlikely that these fundamental considerations will

2

	Min	Mean	Max	Lower Bound
Berkeley – Canberra	174.0	174.7	176.0	79.9
Berkeley – New York	85.0	85.0	85.0	27.4
Berkeley – Trondheim	197.0	197.0	197.0	55.6
Pittsburgh – Ottawa	44.0	44.1	62.0	4.3
Pittsburgh – Hong Kong	217.0	223.1	393.0	85.9
Pittsburgh – Dublin	115.0	115.7	116.0	42.0
Pittsburgh – Seattle	83.0	83.9	84.0	22.9

(a) Frame Rate CDF at 100 Mbps   (b) Measured Internet2 Round Trip Times (milliseconds)

Figure 2: Impact of Network Latency on a Highly Interactive Visualization Application (Lagar-Cavilla et al [10])

change in the forseeable future. The prime targets of networking improvements today are bandwidth, security, energy efficiency, and manageability. Often, the techniques used for these improvements hurt latency. For example, firewalls and overlay networks both achieve their goals by increasing the software path length traversed by packets. In wireless networks, a common energy-saving technique is to turn on the mobile device's transceiver only for short periods of time to receive and acknowledge packets that have been buffered at a base station. This increases average end-to-end latency for packets, and also increases jitter. Bandwidth, on the other hand, may be hardly affected by these techniques because it is an aggregate rather than instantaneous measure. Although bandwidth will continue to improve over time, latency is unlikely to improve dramatically. In fact, it may worsen.

3.3   Bandwidth-induced Delays Also Hurt

While the discussion above has focused on Internet latency and jitter, there is also a very different source of user-perceived delay arising from the transmission of large data items that need to be processed within a tight user-machine interaction loop. For example, executing computer vision algorithms on a high-resolution scene image or on high-definition scene video is a processor-intensive task that is a natural candidate for offloading to a high-performance computing engine. The user-perceived delay in this case is not just the processing time but also includes the time it takes for bulk data transfer across the network. This delay is determined by the bandwidth available in the network.

Wireless LAN bandwidth is typically two orders of magnitude higher than the wireless Internet bandwidth available to a mobile device. For example, the nominal bandwidths of the fastest currently-available wireless LAN (802.11n) and wireless Internet (HSPDA) technologies are 400 Mbps and 2 Mbps respectively. From the viewpoint of user interaction, the difference in transmission delays at these bandwidths can be very significant: 80 milliseconds instead of 16 seconds for a 4MB JPEG image. This is a huge difference for a deeply immersive application. Even if wireless Internet bandwidth improves by one order of magnitude, wireless LAN bandwidths are also poised to improve by a large amount.

4   How Cloudlets Can Help

Can we obtain the benefits of cloud computing without being WAN-limited? Rather than relying on a distant "cloud," the resource poverty of a mobile device can be addressed by using a nearby resource-rich cloudlet. The need for real-time interactive response can be met by low-latency, one-hop, high-bandwidth wireless access to the cloudlet. The mobile device functions as a thin client, with all significant computation occurring in the nearby cloudlet. Physical proximity of the cloudlet is essential: the end-to-end response time of applications executing in the cloudlet needs to be fast (few milliseconds) and predictable. If no cloudlet is available nearby, the mobile device can gracefully degrade to a fallback mode that involves a distant cloud or, in the worst case, solely its own resources. Full functionality and performance can return later, when a nearby cloudlet is discovered.

As Figure 4(a) illustrates, cloudlets are decentralized and widely-dispersed Internet infrastructure whose compute cycles and storage resources can be leveraged by nearby mobile computers. A cloudlet can be viewed as a *"data center in a box."* It is self-managing, requiring little more than power, Internet connectivity, and access control for setup. This simplicity of management corresponds to an appliance model of computing resources, and makes it trivial to deploy on a business premises such as a coffee shop or a doctor's office. Internally, a cloudlet may be viewed as a cluster of multi-core computers, with gigabit internal connectivity and a high-bandwidth wireless LAN. For safe deployment in unmonitored areas, the cloudlet may be packaged in a tamper-resistant or tamper-evident enclosure with third-party remote monitoring of hardware integrity. Figure 4(b) summarizes some of the key differences between cloudlets and clouds. Most importantly, a cloudlet only contains *soft state* such as cache copies of

3

Resp. Time	Subjective Impression
< 150ms	Crisp
150ms - 1s	Noticeable to Annoying
1s - 2s	Annoying
2s - 5s	Unacceptable
> 5s	Unusable

(a) Mapping of Response Times

RTT	Crisp	Noticeable	Annoying	Unaccep.	Unusable
1ms	3278	40	0	0	0
20ms	3214	82	4	18	0
66ms	2710	572	12	3	21
100ms	2296	973	20	6	23

(b) Response Time Distribution of Individual GIMP Interactions

Figure 3: Usability Impact of Network Latency at 100 Mbps for GIMP over VNC (Tolia et al [18])

data or code that is available elsewhere. Loss or destruction of a cloudlet is hence not catastrophic.

5 Transient Cloudlet Customization

We imagine a future in which cloudlet infrastructure is deployed much like Wi-Fi access points today. Indeed, it would be relatively straightforward to integrate cloudlet and Wi-Fi access point hardware into a single easily-deployable entity. A key challenge is to simplify cloudlet management. Widespread deployment of cloudlet infrastructure will not happen unless software management of that infrastructure is trivial — ideally, it should be totally self-managing. Tightly restricting software on cloudlets to simplify management is unattractive because it constrains application innovation and evolution. Instead, an ideal cloudlet would support the widest possible range of mobile users, with minimal constraints on their software.

Our solution is *transient customization of cloudlet infrastructure* using hardware virtual machine (VM) technology. The emphasis on "transient" is important: pre-use customization and post-use cleanup ensures that cloudlet infrastructure is restored to its pristine software state after each use, without manual intervention. A VM cleanly encapsulates and separates the transient *guest* software environment from the permanent *host* software environment of the cloudlet infrastructure. The interface between the host and guest environments is narrow, stable, and ubiquitous. This ensures the longevity of cloudlet investments and greatly increases the chances of a mobile user finding compatible cloudlets anywhere in the world. The malleable software interfaces of resource-rich mobile applications are encapsulated within the guest environment and are hence precisely re-created during pre-use customization of cloudlets. As a result, a VM-based approach is less brittle than alternatives such as process migration or software virtualization [13]. It is also less restrictive and more general than language-based virtualization approaches that require applications to be written in a specific language such as Java or C#.

There are two different approaches to delivering VM state to infrastructure. One is a *VM migration* approach in which an already-executing VM is first suspended; its processor, disk and memory state are then transferred; fi-

nally, VM execution is resumed at the destination from the exact point of suspension. The basic feasibility of this approach has been confirmed by our work on the Internet Suspend/Resume (ISR) system [8, 16] and SoulPad [4], and by other work such as the Collective [14] and Xen live migration [6].
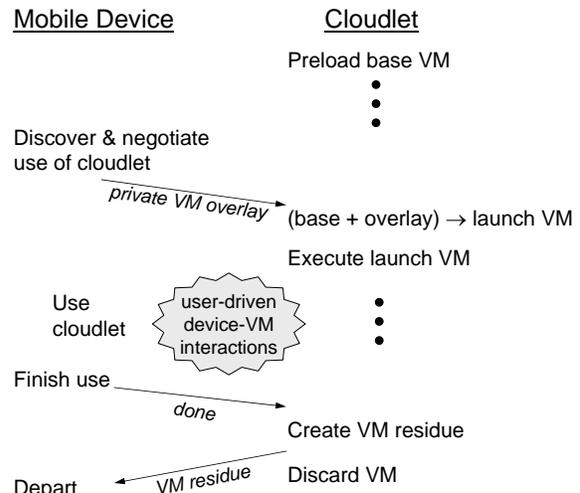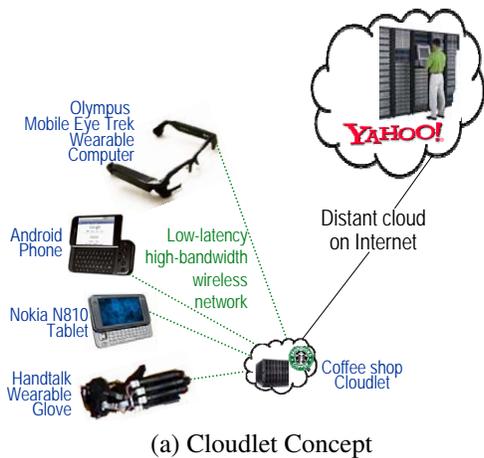

Figure 5: Dynamic VM Synthesis Timeline

The other approach, which is the focus of this paper, is called *dynamic VM synthesis.* A small *VM overlay* is delivered by a mobile device to cloudlet infrastructure that already posseses the base VM from which this overlay was derived. The infrastructure applies the overlay to the base to derive the *launch VM,* which starts execution in the precise state from which the overlay was derived. Figure 5 illustrates the steps of this approach. In a language translation application, for example, the software in the launch VM could be a server that receives captured speech from a mobile device, performs speech recognition and language translation, and returns the output for speech synthesis. If the cloudlet is a cluster, the launch VM could be rapidly cloned to exploit parallelism, as described by Lagar-Cavilla et al [11].

We anticipate that a relatively small number of base VMs (perhaps a dozen or so releases of Linux and Windows configurations) will be popular worldwide in

4

(a) Cloudlet Concept

	Cloudlet	**Cloud**
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24x7 operator
Environment	"Datacenter in a box" at business premises	Machine room with power conditioning and cooling
Ownership	Decentralized ownership by local business	Centralized ownership by Amazon, Yahoo!, etc.
Network	LAN latency/bandwidth	Internet latency/bandwidth
Sharing	Few users at a time	100s-1000s of users at a time

(b) Key Differences: Cloudlet vs. Cloud

Figure 4: What is a Cloudlet?

cloudlets at any given time. Hence, the chances will be high that a mobile device will find a compatible base for its overlays even far from home.

It is useful to contrast dynamic VM synthesis with the alternative approach of assembling a large file from hash-addressed chunks. Variants of the alternative approach have been used in systems such as LBFS [12], Casper [19], Shark [3], the Internet Suspend/Resume system [9], the Collective [14], and KeyChain [2]. All of these variants have a probabilistic character to them: chunks that are not available nearby (in the local cache, on portable storage, and so on, depending on the specific variant) have to be obtained from the cloud. The bandwidth to the cloud and the hit ratio on chunks are the dominant factors affecting speed of assembly. Dynamic VM synthesis differs in two key ways. First, its performance is determined solely by local resources: bandwidth to cloudlet and compute power of the cloudlet. Local hardware upgrades can thus translate directly to faster VM synthesis. Second, WAN failures do not affect synthesis. Even a cloudlet that is totally isolated from the Internet can be used, since the overlay is delivered from the mobile device. In this case, the provisioning of the cloudlet with base VMs could be done through physical storage media.

6 Feasibility of Dynamic VM Synthesis

We have built a proof-of-concept prototype called *Kimberley* to explore the feasibility of dynamic VM synthesis. The mobile device in this prototype is a Nokia N810 Internet tablet running Maemo 4.0 Linux. Cloudlet infrastructure is represented by a standard desktop running Ubuntu Linux. We briefly describe the prototype and experimental results from it below. Wolbach et al provide more details in a 2008 workshop paper [20].

6.1 VM Overlay Creation

Kimberley uses VirtualBox, a hosted VMM for Linux. A tool called `kimberlize` is used to create VM overlays, using `baseVM`, `install-script`, and `resume-script` as inputs. `baseVM` is a VM in which a minimally-configured guest OS has been installed. There are no constraints on the choice of guest OS, except that it must be compatible with `install-script` and `resume-script`. The tool first launches `baseVM`, and then executes `install-script` in the guest OS. The result is a VM that has been configured for use by the mobile device. Next, the tool executes `resume-script` in the guest OS. This launches the desired application, and brings it to a state that is ready for user interaction. This VM, called `launchVM`, is now suspended. It can be resumed rapidly at runtime without the delays of guest reboot or application launch. After creating `launchVM`, `kimberlize` differences its memory and disk images with those of `baseVM` to obtain the VM overlay. The final step is to compress and encrypt this overlay.

6.2 Binding to Cloudlet Infrastructure

Figure 6 shows the key runtime components of Kimberley. The controller of the transient binding between mobile device and cloudlet is a user-level process called *Kimberley Control Manager (KCM)*. An instance of KCM runs on the device and on the cloudlet, and they together abstract service discovery and network management from the rest of Kimberley. KCM supports the browsing and publishing of services using the Avahi mechanism in Linux.

The first step in the binding sequence is the establishment of a secure TCP tunnel using SSL between KCM instances on a device and a cloudlet. This tunnel is then used by the rest of the binding sequence, which typically involves user authentication and optional billing interac-
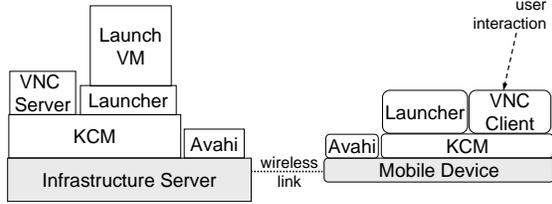
Figure 6: Runtime Binding in Kimberley

Application	Compressed VM Overlay Size (MB)	Uncompressed VM Overlay Size (MB)	Install Package Size (MB)
AbiWord	119.5	364.2	10.0
GIMP	141.0	404.7	16.0
Gnumeric	165.3	519.8	16.0
Kpresenter	149.4	426.8	9.1
PathFind	196.6	437.0	36.8
SnapFind	63.7	222.0	8.8
Null	5.9	24.8	0.0

Table 1: VM Overlay Sizes for 8 GB Virtual Disk

Figure 7: VM Synthesis Time at 100 Mbps (seconds)

tion. Kimberley supports the Simple Authentication and Security Layer (SASL) framework, which provides an extensible interface for integrating diverse authentication mechanisms. After successful authentication, the cloudlet KCM executes a `dekimberlize` command. This fetches the VM overlay from the mobile device or a Web site, decrypts and decompresses it, and applies the overlay to the base VM. The suspended VM is then launched, and is ready to provide services to the mobile device.

6.3 Speed of VM Synthesis

Table 1 shows that VM overlay size is 100–200 MB for a sample collection of Linux applications. This is an order of magnitude smaller than the full VM size of over 8GB. The row labelled *Null* in Table 1 shows that the intrinsic overhead of Kimberley is modest.

For use in cloudlets, rapid VM synthesis is important. Mobile users who rely on cloudlet services will find extended delays for service initiation at a new location to be unacceptable. In addition, cloudlet handoffs should be as rapid, invisible and seamless as Wi-Fi access point handoffs are today — a good potential use of VM migration after initial VM synthesis.

Figure 7 presents the measured VM synthesis time in Kimberley for six Linux applications, when the VM overlay is delivered to the cloudlet at 100 Mbps. The times range from under a minute to just over a minute and a half. These figures are likely to improve over time since Kimberley is an unoptimized initial prototype, with many performance optimizations possible.
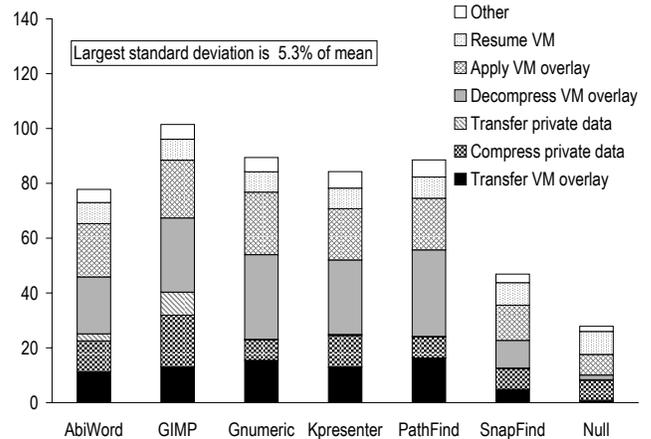
6.4 Improving Performance

Synthesizing a VM in 60 – 90 seconds is acceptable for an unoptimized proof-of-concept prototype, but significant improvement is needed for real-world deployment. Exploring these performance improvements is part of our future work. We conjecture that synthesis times in the small tens of seconds is a desirable and practically achievable goal, requiring about a factor of five improvement. As shown in Figure 7, the major contributors to VM synthesis time are (a) overlay transmission and (b) decompressing and applying the overlay on the cloudlet.

Overlay transmission time can be improved by using a higher-bandwidth short-range wireless network. Relative to the 100 Mbps network used in our experiments, wireless LAN bandwidths are poised to improve through a number of new wireless technologies that are on the brink of commercial relevance. Examples of such technologies include: 802.11n (300-600 Mbps), Ultra-wideband (UWB) (100-480 Mbps), and 60 GHz radio (1-5 Gbps). We anticipate significant development effort in translating these nominal bandwidth improvements into true end-to-end improvements, especially since one of the end points is a mobile device that is not optimized for high performance. However, this is a challenge that has been successfully met in the past with each major improvement in networking technology. We are confident of eventual success, although the path to getting there may be convoluted.

To reduce decompression and overlay application times, one can exploit parallelism. Since these operations are performed on the cloudlet, not the mobile device, there is ample opportunity to take advantage of multi-core computing resources. For example, partitioning the VM image into four parts and generating four (smaller) overlays will allow a four-core cloudlet to synthesize the parts in parallel to achieve close to a 4X speedup. The overall de-

compression and overlay application workload is embarrasingly parallel, allowing higher degrees of parallelism to be exploited quite easily. In addition, it may be possible to pipeline this step with overlay transmission.

Another approach is to use caching, speculative synthesis, and prefetching techniques to eliminate VM synthesis delay. Temporal locality of user mobility patterns suggests that persistent caching of launch VMs may be valuable in eliminating the need for VM synthesis on return visits by a user to a cloudlet. Other users may also benefit, if they use the same launch VM. An idle cloudlet and a mobile device could cooperate in speculative VM synthesis if there is a strong hint of a visit to that cloudlet in the near future. The hints needed for speculative synthesis may be obtained from high-level user information such as location tracking, context information, online calendar schedules, and history-based sources. The cost of erroneous speculation can be kept acceptable by executing the synthesis at low priority.

Synthesis can be applied recursively to generate a family of overlays. Creating a launch VM would then involve pipelined application of these overlays, with intermediate results cached for reuse. Earlier stages of the pipeline would tend to involve larger overlays that are more widely used across applications and are hence more likely to be found in a persistent cache. Conceptually, we seek a "wavelet"-like decomposition of VM state into a sequence of overlays that decrease in size but increase in specificity. A tradeoff is that each overlay introduces some delay in pre-use customization of infrastructure. The cost of generating overlays is not a factor, since it occurs offline.

6.5 Deployment Challenges

Many practical considerations need to be addressed before the vision described in this paper becomes reality. One obvious question pertains to the business model for cloudlet deployment. Is deployment driven bottom-up by business owners installing cloudlets for the benefit of their customers, much as they install comfortable furniture today? Or is it driven top-down by service providers who share profits with the retail businesses on whose premises cloudlets are deployed? In the latter case, how should pricing plans be structured to attract users while leaving room for profit? These are only two examples of many possible business models, and it is difficult to predict at this early stage which of them will prove to be successful.

A different set of deployment questions pertain to sizing of cloudlets. How much processing, storage, and networking capacity should a cloudlet possess? How do these resource requirements depend on the specific applications supported? How do they vary over time in the short term and long term, taking into account natural clustering of users? How do cloudlet resource demands vary across individual users and groups of users?

How sparse can cloudlet infrastructure be, yet provide a satisfactory user experience? What management policies should cloudlet providers use to maximize user experience while minimizing cost?

Trust and security issues are also major factors in cloudlet deployment. The thick VM boundary insulates a cloudlet from software executed by careless or malicious users. However, a user's confidence in the safety of cloudlet infrastructure rests on more fragile assumptions. For example, a malicious VMM could subtly distort the execution of language translation within a VM, and thus sabotage an important business transaction without the user being aware of the damage. One approach to coping with this problem is *trust establishment,* where the user performs some pre-use action to check the host software on a cloudlet. Examples of this approach have been described by Garriss et al [7] and Surie et al [17]. A different approach is *reputation-based trust,* in which the user verifies the identity of the cloudlet service provider and then relies on legal, business or other external considerations to infer trust. The first approach is more defensive and robust, but also more cumbersome. The second approach is more fragile, but also more usable because it is fast and minimally intrusive. A useful everyday metaphor is drinking water from a faucet. One can boil the water before drinking (trust establishment), or one can infer safety because one lives in a first-world country (reputation-based trust). Time will tell which of these approaches proves more viable in real-world deployments.

7   Conclusion

Resource poverty is a fundamental constraint that severely limits the class of applications that can be run on mobile devices. This constraint is not just a temporary limitation of current technology, but is intrinsic to mobility. In this paper, we put forth a vision of mobile computing that breaks free of this fundamental constraint. In this vision, mobile users seamlessly utilize nearby computers to obtain the resource benefits of cloud computing without incurring WAN delays and jitter. Rather than relying on a distant "cloud," a mobile user instantiates a "cloudlet" on nearby infrastructure and uses it via a wireless LAN. Crisp interactive response for immersive applications that augment human cognition is then much easier to achieve because of the proximity of the cloudlet. We confirm that a critical untested aspect of this vision, namely rapid customization of cloudlet infrastructure, is achievable through dynamic VM synthesis. While much remains to be done, the concepts and ideas introduced here open the door to a new world of mobile computing in which seamless cognitive assistance of users occurs in diverse ways at any time and place.

Acknowledgements

References

[1] ADLER, A., AND SCHUCKERS, M. E. Comparing Human and Automatic Face Recognition Performance. *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics 37*, 5 (October 2007).

[2] ANNAMALAI, M., BIRRELL, A., FETTERLY, D., AND WOBBER, . Implementing Portable Desktops: a New Option and Comparisons. Tech. Rep. MSR-TR-2006-151, Microsoft Research, October 2006.

[3] ANNAPUREDDY, S., FREEDMAN, M. J., AND MAZIÈRES, D. Shark: Scaling File Servers via Cooperative Caching. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)* (Boston, MA, May 2005).

[4] CACERES, R., CARTER, C., NARAYANASWAMI, C., AND RAGHUNATH, M. Reincarnating PCs with Portable SoulPads. In *Proc. of the 3rd Intl. Conference on Mobile Systems, Applications, and Services* (Seattle, WA, June 2005).

[5] CARBONELL, J., KLEIN, S., MILLER, D., STEINBAUM, M., GRASSIANY, T., AND FREY, J. Context-based Machine Translation. In *Proc. of the 7th Conf. of the Assoc. for Machine Translation in the Americas* (Cambridge, MA, August 2006).

[6] CLARK, C., FRASER, K., HAND, S., HANSEN, J. G., JUL, E., LIMPACH, C., PRATT, I., AND WARFIELD, A. Live Migration of Virtual Machines. In *Proc. of the 2nd USENIX Symposium on Networked Systems Design and Impl. (NSDI)* (Boston, MA, May 2005).

[7] GARRISS, S., CACERES, R., BERGER, S., SAILER, R., VAN DOORN, L., AND X.ZHANG. Trustworthy and Personalized Computing on Public Kiosks. In *Proc. of Mobisys 2008* (Breckenridge, CO, June 2008).

[8] KOZUCH, M., SATYANARAYANAN, M. Internet Suspend/Resume. In *Proc. of the Fourth IEEE Workshop on Mobile Computing Systems and Applications* (Calicoon, NY, June 2002).

[9] KOZUCH, M., SATYANARAYANAN, M., BRESSOUD, T., HELFRICH, C., SINNAMOHIDEEN, S. Seamless Mobile Computing on Fixed Infrastructure. *IEEE Computer 37*, 7 (July 2004).

[10] LAGAR-CAVILLA, H. A., TOLIA, N., DE LARA, E., SATYANARAYANAN, M., AND O'HALLARON, D. Interactive Resource-Intensive Applications Made Easy. In *Proceedings of Middleware 2007: ACM/IFIP/USENIX 8th International Middlewae Conference* (Newport Beach, CA, November 2007).

[11] LAGAR-CAVILLA, H. A., WHITNEY, J., SCANNELL, A., PATCHIN, P., RUMBLE, S., DE LARA, E., BRUDNO, M., AND SATYANARAYANAN, M. SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In *Proceedings of EuroSys 2009* (Nuremberg, Germany, March 2009).

[12] MUTHITACHAROEN, A., CHEN, B., MAZIERES, D. A Low-Bandwidth Network File System. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles* (Banff, Canada, Oct. 2001).

[13] OSMAN, S., SUBHRAVATI, D., SU., G., NIEH, J. The Design and Implementation of Zap: A System for Migrating Computing Environments. In *Proc. of the 5th Symposium on Operating Systems Design and Impl.* (Boston, MA, Dec. 2002).

[14] SAPUNTZAKIS, C., CHANDRA, R., PFAFF, B., CHOW, J., LAM, M., ROSENBLUM, M. Optimizing the Migration of Virtual Computers. In *Proc. of the 5th Symposium on Operating Systems Design and Impl.* (Boston, MA, Dec. 2002).

[15] SATYANARAYANAN, M. Fundamental Challenges in Mobile Computing. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (1996).

[16] SATYANARAYANAN, M., GILBERT, B., TOUPS, M., TOLIA, N., SURIE, A., O'HALLARON, D. R., WOLBACH, A., HARKES, J., PERRIG, A., FARBER, D. J., KOZUCH, M. A., HELFRICH, C. J., NATH, P., AND LAGAR-CAVILLA, H. A. Pervasive Personal Computing in an Internet Suspend/Resume System. *IEEE Internet Computing 11*, 2 (2007).

[17] SURIE, A., PERRIG, A., SATYANARAYANAN, M., AND FARBER, D. Rapid Trust Establishment for Pervasive Personal Computing. *IEEE Pervasive Computing 6*, 4 (2007).

[18] TOLIA, N., ANDERSEN, D., SATYANARAYANAN, M. Quantifying Interactive Experience on Thin Clients. *IEEE Computer 39*, 3 (Mar. 2006).

[19] TOLIA, N., KOZUCH, M., SATYANARAYANAN, M., KARP, B., BRESSOUD, T., PERRIG, A. Opportunistic Use of Content-Addressable Storage for Distributed File Systems. In *Proceedings of the 2003 USENIX Annual Technical Conference* (San Antonio, TX, June 2003).

[20] WOLBACH, A., HARKES, J., CHELLAPPA, S., AND SATYANARAYANAN, M. Transient Customization of Mobile Computing Infrastructure. In *Proc. of the MobiVirt 2008 Workshop on Virtualization in Mobile Computing* (Breckenridge, CO, June 2008).

Sidebar
Help for the Mentally Challenged

Imagine a future in which there are extensive deployments of dense cloudlet infrastructure based on open standards, much like Wi-Fi access points today. What kind of new applications can we envision in such a world?

Ron has recently been diagnosed with Alzheimer's disease. Due to the sharp decline in his mental acuity, he is often unable to remember the names of friends and relatives. He also frequently forgets to do simple daily tasks. He faces an uncertain future that is clouded by a lack of close family nearby, and limited financial resources for professional caregivers. Even modest improvements in his cognitive ability would greatly improve his quality of life, while also reducing the attention demanded from caregivers. This would allow him to live independently in dignity and comfort for many more years, before he has to move to a nursing home.

Fortunately, a new experimental technology may provide Ron with cognitive assistance. At the heart of this technology is a lightweight wearable computer with a head-up display in the form of eyeglasses. Built into the eyeglass frame are a camera for scene capture, and earphones for audio feedback. These hardware components offer the essentials of an augmented-reality system to aid cognition when they are combined with software for scene interpretation, face recognition, context awareness and voice synthesis. When Ron looks at a person for a few seconds, that person's name is whispered in his ear along with additional cues to guide Ron's greeting and interactions; when he looks at his thirsty houseplant, "water me" is whispered; when he look at his long-suffering dog, "take me out" is whispered. Ron's magic glasses travel with him, transforming his surroundings into a helpful smart environment.

In this example, low-latency, high-bandwidth wireless access to cloudlet resources is an essential ingredient for the "magic glasses" to be able to execute computer vision algorithms for scene analysis and face recognition at real-time speeds. This is only one of many new applications that we can imagine.