The Case for VM-Based Cloudlets in Mobile Computing

A new vision of mobile computing liberates mobile devices from severe resource constraints by enabling resource-intensive applications to leverage cloud computing free of WAN delays, jitter, congestion, and failures.

Mahadev Satyanarayanan
Carnegie Mellon University

Paramvir Bahl
Microsoft Research

Ramón Cáceres
AT&T Research

Nigel Davies
Lancaster University

Mobile computing is at a fork in the road. After two decades of sustained effort by many researchers, we've finally 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 just a dream in the mid 1990s; today, ubiquitous email and Web access is a reality that millions of users worldwide experience through BlackBerries, iPhones, Windows Mobile, and other mobile devices. On one path of the fork, mobile Web-based services and location-aware advertising opportunities have begun to appear, and companies are making large investments in anticipation of major profits.

Yet, this path also leads mobile computing away from its true potential. Awaiting discovery on the other path is an entirely new world in which mobile computing seamlessly augments users' cognitive abilities via 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 (see the sidebar for an example).

This article discusses the technical obstacles to this transformation and proposes a new architecture for overcoming 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's well-connected to the Internet and 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 cloudlet's physical proximity and one-hop network latency. Using a cloudlet also simplifies the challenge of meeting the peak bandwidth demand of multiple users interactively generating and receiving media such as high-definition video and high-resolution images. Rapid customization of infrastructure for diverse applications emerges as a critical requirement, and our results from a proof-of-concept prototype suggest that VM technology can indeed help meet this requirement.

Resource-Poor Mobile Hardware

The phrase "resource-rich mobile computing" seems like an oxymoron at first glance. Researchers have long recognized that mobile hardware is necessarily resource-poor relative

Published by the IEEE CS ■ 1536-1268/09/$26.00 © 2009 IEEE

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's 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 might 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 combined with software for scene interpretation, facial 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 looks at his long-suffering dog, "take me out" is whispered.

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 facial recognition at real-time speeds. This is only one of many new applications that we can imagine.

to static client and server hardware.[1] At any given cost and level of technology, considerations such as 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 user's viewpoint, a mobile device can never be too small or light or have too long a battery life. Although mobile hardware continues to evolve and improve, it will always be resource-poor relative to static hardware—simply put, for the 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 isn't just a temporary limitation of current technology but is intrinsic to mobility. Computation on mobile devices will thus always involve a compromise.

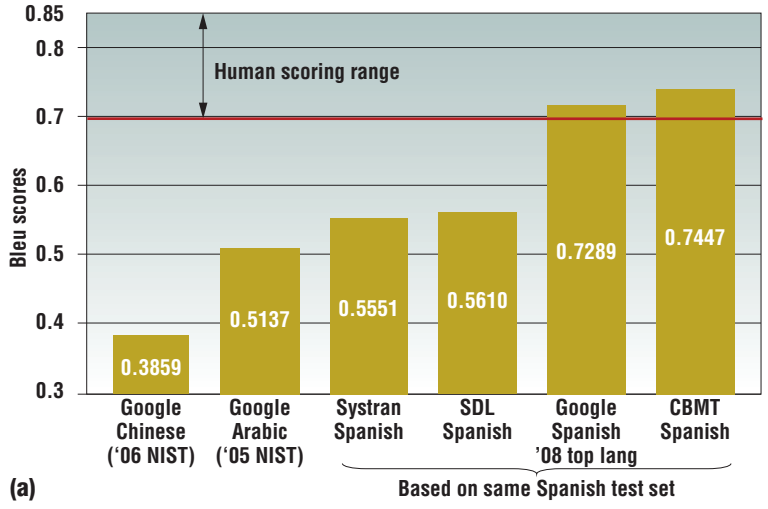
Resource poverty is a major obstacle for many applications with the potential to seamlessly augment human cognition because such applications typically require processing and energy that far outstrips mobile hardware's capabilities. In the lab and with ample computing resources, the state of the art for applications such as face recognition, speech recognition, and language translation is near-human in performance and quality. As Figure 1a shows, for example, researchers achieved Spanish-English translation comparable to human quality in 2006 on a 100-node computing engine by using large online corpora and a context-based machine translation algorithm.[2] For the IBM BLEU metric used in the figure, scores above 0.7 enter the bilingual human translator range and those above 0.8 approach the experienced professional human translator range. Face recognition using computer vision is another area in which rapid progress has occurred over the past decade. Figure 1b, adapted from Andy Adler and Michael Schucker's 2007 comparison of human and automatic face recognition performance,[3] shows that computers and humans are comparable in this task today. Although several technical improvements for practical deployment are still needed in such applications, it doesn't take a giant leap of faith 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.

The Limits of Cloud Computing

An obvious solution to mobile devices' resource poverty 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, long WAN latencies are a fundamental obstacle.

Why Latency Hurts

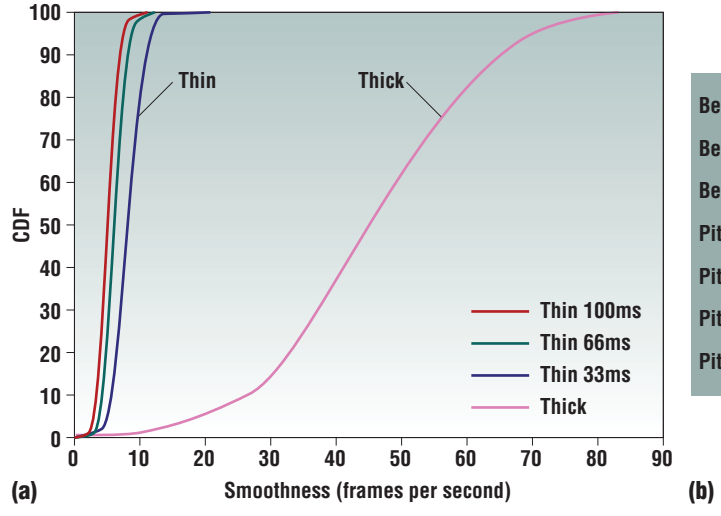
WAN delays in the critical path of user interaction can hurt usability by degrading the crispness of system response. Even trivial user–application interactions incur delays in cloud computing. Humans are acutely sensitive to



(b)

Year	Computer worse than human (%)	Computer better than human (%)	Indeterminate (%)	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

Figure 1. Near-human quality of cognitive augmentation applications today. Machines are much more capable of matching humans in (a) language translation[2] and (b) facial recognition[3] than in the past.



(b)

	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

Network latency hurts interactive performance even with good bandwidth: (a) a highly interactive visualization application's measured output frame rate under two different configurations: "Thick" (a local machine with hardware graphics acceleration) and "Thin" (a remote compute server on a 100 Mb/s network, with round trip times ranging from 33ms to 100ms) (b) measured Internet2 round trip times between representative sites confirm that the 33-100ms range lies well within the range of observed latencies in the real world.

delay and jitter, and it's very difficult to control these parameters at WAN scale: as latency increases, interactive response suffers. Loosely coupled tasks such as Web browsing might 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.

Andrés Lagar-Cavilla and his colleagues[4] showed that latency can negatively impact interactive response in spite of adequate bandwidth. Figure 2a compares the measured output frame rate of a visualization application (Quake-Viz) under two different configurations: local machine with hard-

Resp. time	Subjective impression
<150ms	Crisp
150ms–1s	Noticeable to Annoying
1s–2s	Annoying
2s–5s	Unacceptable
> 5s	Unusable

(a)

RTT	Crisp	Noticable	Annoying	Unacceptable	Unusable
1ms	3,278	40	0	0	0
20ms	3,214	82	4	18	0
66ms	2,710	572	12	3	21
100ms	2,296	973	20	6	23

(b)

Figure 3. Network latency's effect on usability. At 100 Mbps for GIMP on VNC, (a) the mapping of response times and (b) the response time distribution of individual GIMP interactions shows that user experience degrades significantly as network latency increases.

ware graphics acceleration ("thick") and remote compute server over a 100 Mb/s network with the output viewed through the VNC protocol ("thin"). A high frame rate provides the illusion of smoothness to an interactive user. Figure 2a shows that even a modest latency of 33 ms causes the frame rate to drop considerably from that 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 the frames but offer sluggish interaction. In both cases, the user experience is considerably poorer than it is for local interaction. Figure 2b reports measured Internet2 latencies between representative endpoints at planetary scale,[4] with the measured figures far exceeding the speed-of-light lower bound in the last column.

Independently, Niraj Tolia and his colleagues[5] showed that the user-perceived quality of thin-client performance is highly variable and depends on both the application's degree of interactivity and the network's end-to-end latency. As Figure 3 illustrates, the usability of a highly interactive task such as photo editing suffers unacceptably even at moderate network latency (100 ms round-trip time) and very good bandwidth (100 Mbps). This contrasts with tasks that are interactively undemanding, such as Web browsing. Figure 3b shows the distribution of response times for individual interactions in a GIMP photo editing task. The mapping of response times to the subjective impressions of quality in Figure 3a is based on long-established human-computer interaction guidelines that were developed through empirical studies.

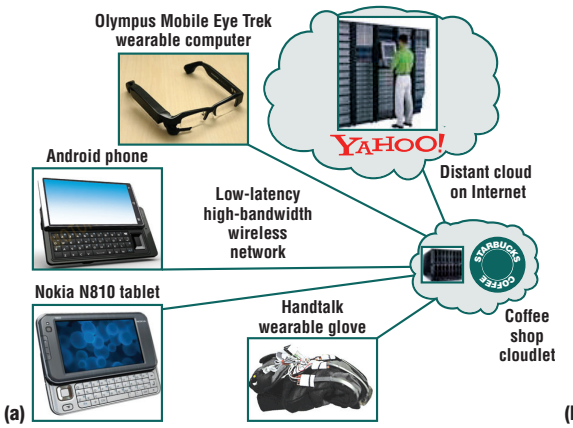
WAN Latency Is Unlikely to Improve

Unfortunately, the current trajectory of Internet evolution makes it very unlikely that these fundamental considerations will change in the foreseeable future. The prime targets of networking improvements today are bandwidth, security, energy efficiency, and manageability, and the techniques used to address them hurt latency. Firewalls and overlay networks, for example, both achieve their goals by increasing the software path length that packets must traverse. 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, which increases average end-to-end packet latency as well as jitter. Bandwidth, on the other hand, might be hardly affected by these techniques because it's an aggregate rather than instantaneous measure. Although bandwidth will continue to improve over time, latency is unlikely to improve dramatically. In fact, it could worsen.

Bandwidth-Induced Delays Also Hurt

Although our discussion so far has focused on Internet latency and jitter, another source of user-perceived delay arises from the transmission of large data items that must be processed within a tight user–machine interaction loop. For example, executing computer vision algorithms on high-resolution images or high-definition video is a processor-intensive task that's a natural candidate for offloading to a high-performance computing engine. The user-perceived delay in this case isn't just the processing time but also includes the time it takes for bulk data transfer across the network. The bandwidth available in the network determines this delay.

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 (High-Speed Downlink Packet Access) technologies are 400 Mbps and 2 Mbps, respectively. From a user interaction viewpoint, the difference in transmission delays at these bandwidths can be very significant: 80 milliseconds instead of 16 seconds for a 4-Mbyte JPEG image, which represents a huge difference for deeply immersive applications. Even if wireless Internet bandwidth improves by one order of



(b)

	Cloudlet	Cloud
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24X 7 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

Figure 4. What is a cloudlet? (a) The cloudlet concept involves proximate computing infrastructure that can be leveraged by mobile devices; it has (b) some key differences with the basic cloud computing concept.

magnitude, wireless LAN bandwidths are also poised to improve by a large amount.

How Cloudlets Can Help

Can we obtain the benefits of cloud computing without being WAN-limited? Rather than relying on a distant "cloud," we might be able to address a mobile device's resource poverty via a nearby resource-rich cloudlet. In this way, we could meet the need for real-time interactive response 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. This cloudlet's physical proximity is essential: the end-to-end response time of applications executing within it must be fast (a 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 the device discovers a nearby cloudlet.

As Figure 4a illustrates, cloudlets are decentralized and widely dispersed Internet infrastructure components whose compute cycles and storage resources can be leveraged by nearby mobile computers. Essentially, a cloudlet resembles a "data center in a box": it's 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 resembles a cluster of multicore computers, with gigabit internal connectivity and a high-bandwidth wireless LAN. For safe deployment in unmonitored areas, the cloudlet can contain a tamper-resistant or tamper-evident enclosure with third-party remote monitoring of hardware integrity. Figure 4b summarizes some of the key differences between cloudlets and clouds. Most importantly, a cloudlet only contains soft state such as cache copies of data or code that's available elsewhere. Hence, a cloudlet's loss or destruction isn't catastrophic.

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 won't 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 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 cloudlet infrastructure's permanent host software environment. The interface between the host and guest environments is narrow, stable, and ubiquitous, which ensures the longevity of cloudlet investments and greatly increases the chances of a mobile user finding compatible cloudlets anywhere in the world. The mallea-

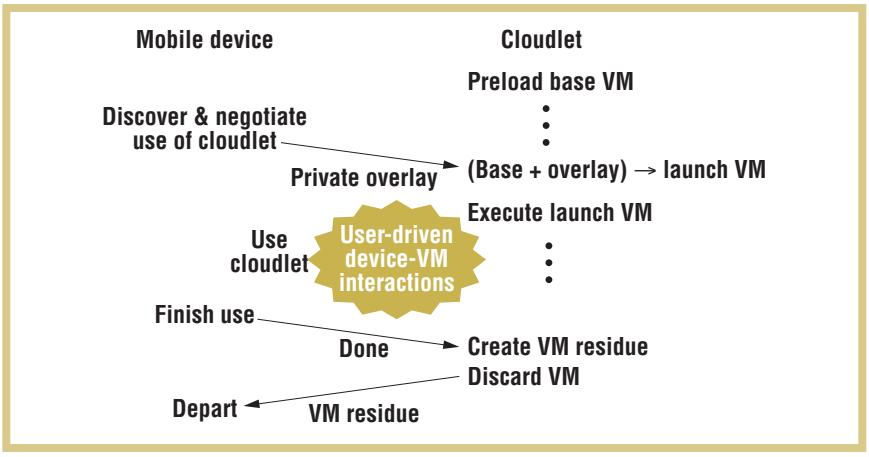


Figure 5. Dynamic virtual machine synthesis timeline. The mobile device transmits the VM overlay to the cloudlet, which applies it to the base VM to generate the launch VM. We anticipate that a relatively small number of base VMs (perhaps a dozen or so releases of Linux and Windows configurations) will be popular worldwide in cloudlets at any given time. Hence, the odds are high that a mobile device will find a compatible base for its overlays even far from home.

ble software interfaces of resource-rich mobile applications are encapsulated within the guest environment and are hence precisely recreated during pre-use cloudlet customization. Consequently, a VM-based approach is less brittle than alternatives such as process migration or software virtualization.[6] It's 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#.

Two different approaches can deliver VM state to infrastructure. One is *VM migration*, in which an already executing VM is suspended, its processor, disk, and memory state are transferred, and finally VM execution is resumed at the destination from the exact point of suspension. We've confirmed this approach's basic feasibility via our work with the Internet Suspend/Resume (ISR) system[7,8] and SoulPad,[9] and by other work such as the Collective[10] and Xen live migration.[11]

The other approach, which is this article's focus, is called *dynamic VM synthesis*. A mobile device delivers a small VM overlay to the cloudlet infrastructure that already possesses the base VM from which this overlay was derived. The infrastructure applies the overlay to the base to derive the launch VM, which starts executing in the precise state in which it was suspended; see Figure 5. 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 Lagar-Cavilla and his colleagues described.[12]

To appreciate its unique attributes, it's useful to contrast dynamic VM synthesis with the alternative approach of assembling a large file from hash-addressed chunks. Researchers have used variants of this alternative in systems such as LBFS,[13] Casper,[14] Shark,[15] the Internet Suspend/Resume system,[16] the Collective,[10] and KeyChain.[17] All these variants have a probabilistic character to them: chunks that aren't available nearby (in the local cache, on portable storage, and so on, depending on the specific variant) must be obtained from the cloud. Thus, bandwidth to the cloud and the hit ratio on chunks are the dominant factors affecting assembly speed. Dynamic VM synthesis differs in two key ways. First, its performance is determined solely by local resources: bandwidth to cloudlet and the cloudlet's compute power. Local hardware upgrades can thus translate directly to faster VM synthesis. Second, WAN failures don't affect synthesis. Even a cloudlet that's totally isolated from the Internet is usable because the mobile device delivers the overlay. In this case, provisioning the cloudlet with base VMs could be done via physical storage media.

Feasibility of Dynamic VM Synthesis

To explore the feasibility of dynamic VM synthesis, we have built a proof-of-concept prototype called Kimberley. 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 here; more details appear elsewhere.[18]

VM Overlay Creation

Kimberley uses VirtualBox, a hosted virtual machine manager (VMM) for Linux. A tool called `kimberlize` creates the VM overlays, using `baseVM`, `install-script`, and `resume-script` as inputs. `baseVM` is a VM with a minimally configured guest operating system (OS) 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's configured for mobile device use. Next, the tool executes `resume-script` in the guest OS to launch the desired application and bring it to a state that's 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

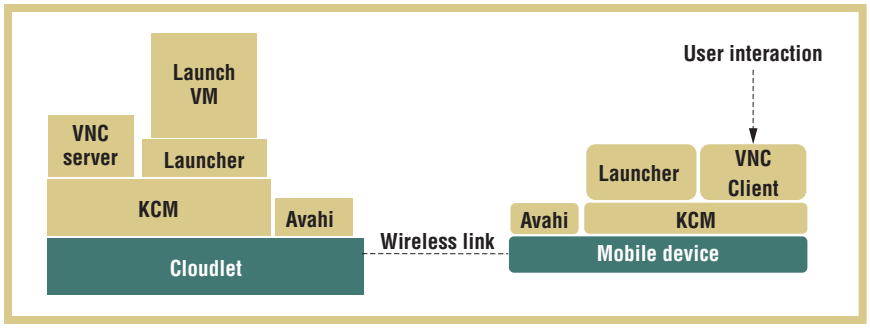


Figure 6. Runtime binding in Kimberley. The KCMs on the cloudlet and mobile device coordinate the binding process.

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.

Binding to Cloudlet Infrastructure

Figure 6 shows Kimberley's key runtime components. 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 together they abstract service discovery and network management from the rest of Kimberley. KCM supports service browsing and publishing using the Avahi mechanism in Linux.

The first step in the binding sequence is the establishment of a secure TCP tunnel using Secure Sockets Layer (SSL) between KCM instances on a device and a cloudlet. The rest of the binding sequence, which typically involves user authentication and optional billing interaction, then uses this tunnel for secure communication. 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, which 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 ready to provide services to the mobile device.

Speed of VM Synthesis

Table 1 shows that VM overlay size is 100 to 200 Mbytes for a sample collection of Linux applications, which is an order of magnitude smaller than the full VM size of more than 8 Gbytes. The row labeled "Null" shows that Kimberley's intrinsic overhead 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 cloudlet receives the VM overlay 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 because Kimberley is an unoptimized initial prototype, with many performance optimizations still possible.

Improving Performance

Synthesizing a VM in 60 to 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 are a desirable and practically achievable goal, requiring about a factor of five improvement. As Figure 7 shows, the

TABLE 1
Virtual machine overlay sizes for an 8-Gbyte virtual disk

Application	Compressed VM overlay size (Mbytes)	Uncompressed VM overlay size (Mbytes)	Install package size (Mbytes)
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

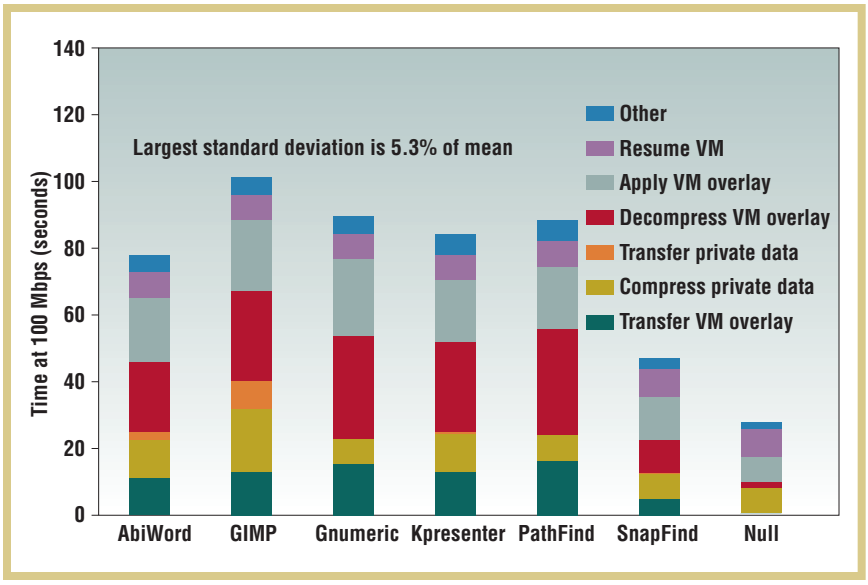


Figure 7. Virtual machine synthesis time at 100 Mbps (seconds). The dominant components are overlay decompression and application, accounting for more than half the time in most cases.

two major contributors to VM synthesis time are overlay transmission and decompressing/applying the overlay on the cloudlet.

We can improve overlay transmission time 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 several new wireless technologies on the brink of commercial relevance—examples include 802.11n (300 to 600 Mbps), ultra-wideband (UWB; 100 to 480 Mbps), and 60-GHz radio (1 to 5 Gbps). We anticipate significant development effort in translating these nominal bandwidth improvements into true end-to-end improvements, especially because one of the endpoints is a mobile device that isn't optimized for high performance. However, this challenge has been successfully met in the past with each major improvement in networking technology. We're confident of eventual success, although the path to getting there might be challenging.

To reduce decompression and overlay application times, we can exploit parallelism. Because these operations are performed on the cloudlet instead of the mobile device, there's ample opportunity to take advantage of multicore computing resources. For example, partitioning the VM image into four parts and generating four (smaller) overlays would allow a four-core cloudlet to synthesize the parts in parallel to achieve close to a 4X speedup. The overall decompression and overlay application workload is embarrassingly parallel, allowing higher degrees of parallelism to be exploited quite easily. In addition, it might be possible to pipeline this step with overlay transmission. We could also use specialized hardware to accelerate decompression and overlay application.

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 might be valuable in eliminating the need for VM synthesis on a user's return visits to a cloudlet. Other users might also benefit if they use the same launch VM. An idle cloudlet and a mobile device could also cooperate in speculative VM synthesis if there's a strong hint of a visit to that cloudlet in the near future. We could obtain such hints from high-level user information such as location tracking, context information, online calendar schedules, and history-based sources. We can keep the cost of erroneous speculation acceptable by executing the synthesis at low priority.

Finally, we can apply synthesis 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 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 trade-off is that each overlay introduces some delay in pre-use infrastructure customization. The cost of generating overlays isn't a factor because it occurs offline.

Deployment Challenges

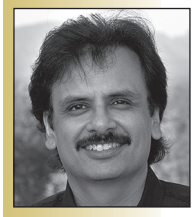
Many practical considerations must be addressed before the vision described in this article becomes reality. One obvious question pertains to the business model for cloudlet deployment: Is deployment driven bottom-up by business owners installing cloudlets for the benefit of their customers, much as 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, which pricing plans will attract users but still leave room for profit? These are only two examples of many possible business models, and it's difficult to predict at this early stage which of them will prove to be successful.

A different set of deployment questions pertain to cloudlet sizing: How much processing, storage, and networking capacity should a cloudlet possess?

the AUTHORS



Mahadev Satyanarayanan is the Carnegie Group Professor of Computer Science at Carnegie Mellon University. His research interests include mobile computing, pervasive computing, and distributed systems. Satyanarayanan has a PhD in computer science from Carnegie Mellon University. He's a fellow of the ACM and the IEEE and the founding editor in chief of this magazine. Contact him at satya@cs.cmu.edu.



Victor Bahl is a principal researcher and founding manager of the Networking Research Group at Microsoft Research. His research interests span a variety of topics in wireless systems design, mobile networking, and network management. Bahl received Digital's Doctoral Engineering Fellowship Award in 1995 and SIGMOBILE's Distinguished Service Award in 2001. In 2004, Microsoft nominated him for the innovator of the year award. Bahl is a fellow of the ACM and the IEEE. Contact him via http://research.microsoft.com/~bahl/.



Ramón Cáceres is a Lead Member of Technical Staff at AT&T Labs in Florham Park, NJ, USA. His research interests include mobile and pervasive computing, virtualization, security, and privacy. He holds a Ph.D. from the University of California at Berkeley and is a member of IEEE, ACM, and USENIX. He was born and raised in Dominican Republic. Contact him at ramon@research.att.com.



Nigel Davies is head of the Computing Department at Lancaster University and an adjunct associate professor of computer science at the University of Arizona. His research interests include systems support for mobile and pervasive computing. He focuses in particular on the challenges of creating deployable mobile and ubiquitous computing systems that can be used and evaluated "in the wild." Contact him at nigel@comp.lancs.ac.uk.

How do these resource requirements depend on the specific applications supported? How do they vary over time in the short 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*, in which the user performs some preuse action to check a cloudlet's host software.[19,20] A different approach is *reputation-based trust*, in which the user verifies the cloudlet service provider's identity 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, whereas the second approach is more fragile but also more usable because it's fast and minimally intrusive. A useful everyday metaphor is drinking water from a faucet: you can boil the water before drinking (trust establishment) or infer safety because you live in an industrialized country (reputation-based trust). Time will tell which of these approaches proves more viable in real-world deployments.

Another deployment challenge relates to the assumption that a relatively small set of base VMs will suffice for a large range of applications. A mobile device with an overlay generated from a base VM that's too old might not be able to find a compatible cloudlet. This problem could be exacerbated by the common practice of releasing security patches for old OS releases. Although the patch's effect could be incorporated into the overlay, it would increase overlay size. A different approach would be to trigger generation of new overlays when security patches are released, which mobile devices would then have to download. Deployment experience can help us choose a good trade-off in this design space.

Although 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 for users occurs in diverse ways at any time and place.

ACKNOWLEDGMENTS

We acknowledge Roy Want for his many contributions to the ideas expressed in this article, and for helping to write and critique its early drafts. We thank the reviewers for their constructive feedback and suggestions for improvement. This research was supported by the US National Science Foundation (NSF) under grant number CNS-0833882. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the NSF, Carnegie Mellon University, Microsoft, AT&T, or Lancaster University. Internet Suspend/Resume is a registered trademark of Carnegie Mellon University.

REFERENCES

1. M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," *Proc. ACM Symp. Principles of Distributed Computing*, ACM Press, 1996, pp. 1–7.
2. J. Carbonell et al., "Context-Based Machine Translation," *Proc. 7th Conf. Assoc. for Machine Translation in the Americas*, Assoc. Machine Translation in the Americas, 2006; http://www.neurosecurity.com/articles/lang/AMTA-2006-Carbonell.pdf.
3. A. Adler and M.E. Schuckers, "Comparing Human and Automatic Face Recognition Performance," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 37, no. 5, pp. 1248–1255.
4. H.A. Lagar-Cavilla et al., "Interactive Resource-Intensive Applications Made Easy," *Proc. Middleware 2007: ACM/IFIP/Usenix 8th Int'l Middlewae Conf.*, Springer, 2007, pp. 143–163.
5. N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying Interactive Experience on Thin Clients," *Computer*, vol. 39, no. 3, 2006, pp. 46–52.
6. S. Osman et al., "The Design and Implementation of Zap: A System for Migrating Computing Environments," *Proc. 5th Symp. Operating Systems Design and Implementation*, Usenix Assoc., 2002; http://www.ncl.cs.columbia.edu/publications/osdi2002_zap.pdf.
7. M. Kozuch and M. Satyanarayanan, "Internet Suspend/Resume," *Proc. 4th IEEE Workshop Mobile Computing Systems and Applications*, IEEE CS Press, 2002, pp. 40–46.
8. M. Satyanarayanan et al., "Pervasive Personal Computing in an Internet Suspend/Resume System," *IEEE Internet Computing*, vol. 11, no. 2, 2007, pp. 16–25.
9. R. Caceres et al., "Reincarnating PCs with Portable Soul-Pads," *Proc. 3rd Int'l Conf. Mobile Systems, Applications, and Services*, Usenix Assoc., 2005; http://www.usenix.org/events/mobisys05/tech/caceres/caceres.pdf.
10. C. Sapuntzakis et al., "Optimizing the Migration of Virtual Computers," *Proc. 5th Symp. Operating Systems Design and Implementation*, Usenix Assoc., 2002; http://suif.stanford.edu/collective/osdi02-optimize-migrate-computer.pdf.
11. C. Clark et al., "Live Migration of Virtual Machines," *Proc. 2nd Usenix Symp. Networked Systems Design and Implementation*, Usenix Assoc., 2005, pp. 273–286.
12. H.A. Lagar-Cavilla et al., "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing," *Proc. EuroSys 2009*, ACM Press, 2009.
13. A. Muthitacharoen, B. Chen, and D. Mazieres, "A Low-Bandwidth Network File System," *Proc. 18th ACM Symp. Operating Systems Principles*, ACM Press, 2001; http://pdos.csail.mit.edu/papers/lbfs:sosp01/lbfs.pdf.
14. N. Tolia et al., "Opportunistic Use of Content-Addressable Storage for Distributed File Systems," *Proc. 2003 Usenix Annual Technical Conf.*, Usenix Assoc., 2003, pp. 127–140.
15. S. Annapureddy, M.J. Freedman, and D. Mazieres, "Shark: Scaling File Servers via Cooperative Caching," *Proc. 2nd Symp. Networked Systems Design and Implementation*, ACM Press, 2005, pp. 129–142.
16. M. Kozuch et al., "Seamless Mobile Computing on Fixed Infrastructure," *Computer*, vol. 37, no. 7, 2004, pp. 65–72.
17. M. Annamalai et al., "Implementing Portable Desktops: A New Option and Comparisons," tech. report MSR-TR-2006-151, Microsoft Research, Oct. 2006.
18. A. Wolbach et al., "Transient Customization of Mobile Computing Infrastructure," *Proc. MobiVirt 2008 Workshop on Virtualization in Mobile Computing*, ACM Press, 2008.
19. S. Garriss et al., "Trustworthy and Personalized Computing on Public Kiosks," *Proc. Mobisys 2008*, ACM Press, 2008, pp. 199–210.
20. A. Surie et al., "Rapid Trust Establishment for Pervasive Personal Computing," *IEEE Pervasive Computing*, vol. 6, no. 4, 2007, pp. 24–30.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

