Bringing the Cloud Down to Earth: Transient PCs Everywhere

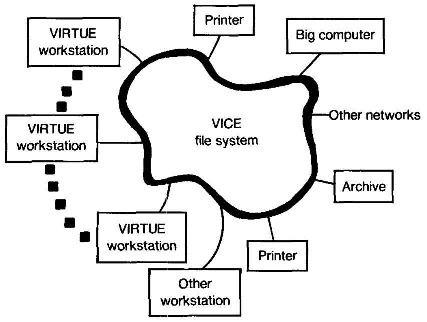
Mahadev Satyanarayanan[1], Stephen Smaldone[2], Benjamin Gilbert[1], Jan Harkes[1], and Liviu Iftode[2]

[1] Carnegie Mellon University
[2] Rutgers University

Abstract. The convergence of cloud computing with mobile computing opens the door to the creation of new applications and services that can be delivered to users at any time and any place. At the heart of this convergence lies a delicate balance between centralization and decentralization. We explore the forces underlying this balance, and examine the role of virtual machine (VM) technology. We observe that a VM-based model of cloud computing called a *Transient PC* offers an approach to "carry-nothing" mobile computing that harnesses the full power of local hardware at the edges of the Internet. In particular, we show how a zero-install Transient PC implementation can safely use local storage.

1 The Roots of Cloud Computing

The intersection of cloud computing with mobile computing is a hot topic today. There is a growing sense that at their nexus lies immense potential for the creation of new applications and services that can be delivered to users at any time and any place. Large dollar signs dance before the eyes of investors and entrepreneurs. What is the basis for this excitement? What are the aspects of this technological convergence that are truly new, and what is merely old wine in new bottles? To fully understand the tectonic forces that are driving mobile computing and cloud computing towards each other, it is important to briefly revisit the past.



(a) Original Image

The amoebalike structure in the middle, called VICE, is a collection of communication and computational resources serving as the backbone of a user community. Individual workstations, called VIRTUEs, are attached to VICE and provide users with the computational cycles needed for actual work as well as a sophisticated user–machine interface. (VICE stands for "Vast, Integrated Communications Environment"; VIRTUE, for "Virtue is reached through UNIX® and EMACS.")

(b) Original Caption

Fig. 1. Cloud Computing *circa* 1986 (Source: Morris et al, 1986 [1])

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2012

At the heart of this convergence lies a fundamental tension between *autonomy* and *interdependence* in distributed systems. This can equally well be characterized as a tension between *centralization* and *decentralization.* On the one hand, mobility and decentralization are all about freedom: the ability to do anything, anywhere, anytime in a completely unconstrained and untethered manner. On the other hand, total isolation is rarely desired by a user. Communication across space and time is essential for collaboration and knowledge sharing across users. Even with respect to a single user, there is often a need to access personal information resources (such as email or files) from the past and across many different mobile or static devices. Unless carefully constrained, the totality of these interrelationships can be overwhelming to a user. System-level mechanisms such as distributed file systems, databases, email clients and hypertext systems as well as design primitives such as cache consistency protocols, content-addressable storage, and atomic transactions can be viewed as order-preserving tools that try to simplify the complexity of information space-time for a user. Centralization of storage in a cloud can be also viewed as an order-preserving tool. Centralization is simpler, less expensive and more orderly than decentralization from the viewpoint of system management.

How to preserve order and hence reduce human-visible complexity (both for users and system administrators), while minimally constraining mobility and flexibility (in their broadest sense) across time, space and multiple computing devices is a fundamental challenge. Finding the "sweet spots" in this tradeoff space has challenged the system designers for over a quarter of a century.

One of the earliest efforts to reconcile this tension took place at the dawn of personal computing in the early 1980s. The goal of this effort, called *the Andrew project,* was to create a system in which users enjoyed the load-invariant, high-quality, feature-rich interactive environment of personal computing while preserving the ease of information sharing that had emerged as a valuable side effect of timesharing. Contemporary accounts of this project [1,2] discuss the design considerations that led to its architecture. The image and caption shown in Figure 1 are reproduced from one of those accounts. It is interesting to note the similarity to modern cloud architectures, even though the term "cloud computing" was nearly 20 years in the future. An Andrew user did not know or care where his data was stored: it was just "somewhere in the cloud" and magically appeared at his current location when it was accessed. An Andrew system administrator could move data across servers for load balancing or add new servers without disrupting users. We associate these attributes of mobility and serviceability with cloud computing today. Usage experience in Andrew revealed the importance of these attributes as early as 1990, as shown by the quotation in Figure 2.

> *User mobility is supported:* A user can walk to any workstation in the system and access any file in the shared name space. A user's workstation is personal only in the sense that he owns it.
>
> *System administration is easier:* Operations staff can focus on the relatively small number of servers, ignoring the more numerous and physically dispersed clients. Adding a new workstation involves merely connecting it to the network and assigning it an address.

Fig. 2. Cloud-like Attributes *circa* 1990 (Source: Satyanarayanan, 1990 [3])

2 VM-Based Cloud Computing Today

Fast forward to the present day. The buzz around cloud computing directly reflects renewed interest in the attributes shown in Figure 2, in the hope of finding a new "sweet spot" in the centralization-decentralization tradeoff space that reflects current economic reality. As hardware costs continue to plummet, the people cost of system admiminstration looms ever larger. The attention now being paid to the metric called *total cost of ownership* reflects this rise in people cost relative to hardware cost. There is pressure towards centralization because it tends to lower system administration costs.

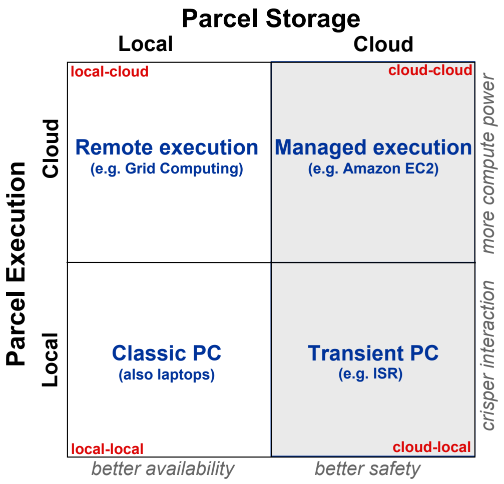


Fig. 3. Taxonomy of VM-based Cloud Computing

Virtual machine (VM) technology leads to new "sweet spots" in the tradeoff space that provide greater centralization without loss of user mobility. Figure 3 identifies these "sweet spots" as quadrants of a two-dimensional space. The vendor-neutral and format-independent term *parcel* in this figure refers to encapsulated VM state. A parcel may contain both volatile state (a memory image) and persistent state (a disk image), or just persistent state. The "Parcel Storage" dimension shows where parcels are stored when not in active use; the "Parcel Execution" dimension shows where they run. The top right quadrant, labeled *Managed Execution,* contains approaches such as Amazon's EC2 [4], IBM's Research Compute Cloud (RC2) [5], and the open source Eucalyptus infrastructure [6], where parcel storage and execution both happen within the cloud. The top left quadrant corresponds to *VM-based grid computing,* where a locally-created parcel is remotely executed. The bottom left quadrant, labeled *Classic PC,* is the degenerate case of a standalone PC, laptop or mobile device.

The bottom right quadrant of Figure 3, labeled *Transient PC,* is of particular interest to mobile computing because it makes possible a "carry-nothing" model of mobility. In this case, parcels are stored in the cloud but execute on a computer that is close to the user. That computer could, of course, be a mobile device. Alternatively, mobile computing can be realized by the user taking advantage of whatever hardware is nearby and letting the system magically deliver a relevant parcel upon use. The latter approach directly corresponds to the Andrew architecture of Figure 1, with the important difference that the granularity of location-transparent access is now an entire parcel rather than user files. In other words, the Transient PC model improves upon the Andrew model by ensuring that the operating system, applications and user files are all delivered from the cloud as a cohesive unit. The technique of *hoarding,* originally developed in the context of a distributed file system [7], can be also applied to parcel state. This enables the Transient PC model to support disconnected operation, a critical capability at the intersection of cloud computing and mobile computing.

There are many technical challenges in efficiently implementing the Transient PC model, the most obvious of these being efficient handling of large parcel size. Fortunately, these implementation challenges can be overcome, as shown by our work in the context of the Internet Suspend/Resume® system (ISR) [8,9] and by related work in the Collective [10,11] and MokaFive [12].

3 Fully Exploiting Local Hardware Resources in Transient PCs

The Transient PC usage model is quite different from the ubiquitous email, Web access, and social networking capabilities provided by BlackBerries, iPhones, and other mobile devices. The strength of Transient PC systems lies in their ability to precisely, safely and rapidly re-create a user's Windows or Linux desktop environment as a thick client on borrowed hardware at any time and place. When a user chooses to use a Transient PC, he or she implicitly confirms the importance of its strengths over the mobile device alternatives mentioned above. It is therefore critical to take full advantage of the local hardware resources of a Transient PC in order to provide a satisfactory user experience.

The most important property of these local resources is their proximity to the user. For interactive applications, executing the application in a distant cloud and viewing the output locally (as would be the case with the top right quadrant of Figure 3) results in an unsatisfactory user experience. The long WAN latencies to the cloud hurt the crisp interaction that is so critical for smooth and non-disruptive interactions. Humans are acutely sensitive to delay and jitter, and it is very difficult to control these parameters at WAN scale. The work by Lagar-Cavilla et al [13] has shown that latency can negatively impact interactive response even when bandwidth is adequate. Networks with high delay-bandwidth products are therefore disastrous for remote execution of interactive applications; in contrast, local execution on a Transient PC does not suffer from this problem. This is particularly true for graphics-intensive applications such as scientific visualization and games, which can further take advantage of graphics acceleration provided by local hardware. Fitting such applications into the Transient PC model is predicated on the ability to virtualize graphics hardware. This was a questionable assumption for many years because of the lack of standardization in such hardware, but VMM-independent virtualization of graphics hardware is now feasible [14].

Another performance-critical resource is the local disk of a Transient PC. To achieve truly ubiquitous availability of Transient PCs, a zero-install solution would be ideal.

Table 1. Portable Storage Device Characteristics

			Size	Speed	Transfer Rate (MB/sec)	
Storage Device	Label	Type	(GB)	(RPM)	Read	Write
PNY Attache USB Flash Drive	SanDrive	USB	16	Flash	30.51 (1.01)	6.65 (0.29)
SanDisk MicroSD Card	MSD	USB	8	Flash	16.03 (0.23)	11.9 (0.2)
Apple iPod	IPOD	USB	20	4200	12.63 (0.18)	12.38 (0.17)
Internal SATA Drive	TransPart	SATA	250	7200	41.78 (2.48)	32.05 (1.14)

Transfer rate results are the mean of 5 measurements. Standard deviations are in parentheses.

This would obviate the need to require pre-installed software on the Transient PC. Instead, the user boots the hardware from a USB storage device to establish the Transient PC environment, and then accesses his cloud-based parcel. The user can thus transform any available hardware with an Internet connection into a Transient PC. Although conceptually simple, this approach runs into a serious performance obstacle. Portable storage devices sacrifice I/O performance in order to obtain the highest capacity and robustness at the lowest cost, size, and weight. As Table 1 shows, the I/O read and write performance of typical USB-attached storage devices is substantially slower that of an internal disk. This severely impacts operating system performance, including basic functionality such as swapping and application launch. Upgrading the interconnect to USB 3.0 will not eliminate this problem, since it is due to internal storage limitations.

To solve this problem, we have developed a mechanism called *TransPart* that constructs a transient virtual disk out of the free disk blocks of local hardware. This mechanism requires no modifications to the software or hardware of a borrowed computer, and thus preserves the zero-install attribute. TransPart currently supports discovery of free disk blocks from ext2/3/4 and NTFS file systems as well as Linux swap partitions.

Figure 4 illustrates the operation of TransPart. During the host boot process from a USB device, TransPart constructs the virtual disk in two phases. In the first phase, TransPart enumerates local devices and then discovers individual storage volumes stored

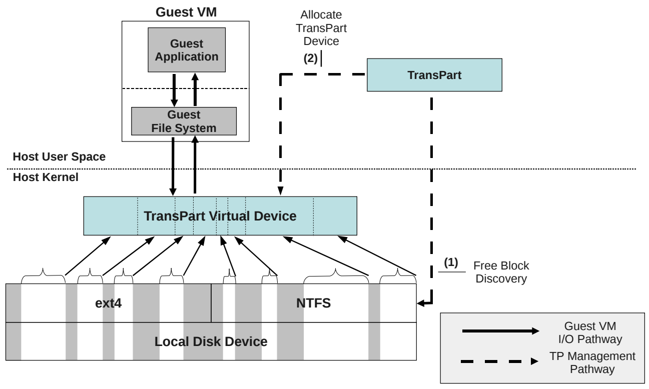


Fig. 4. TransPart Implementation

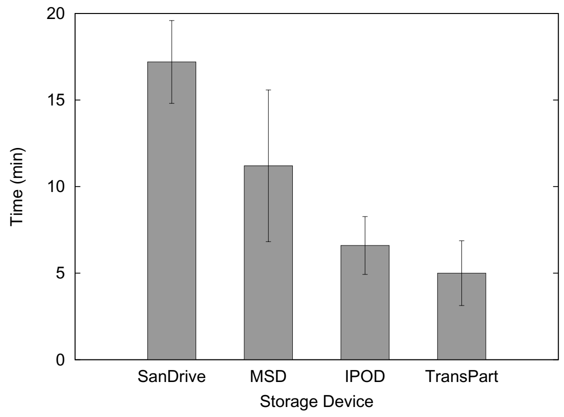


Fig. 5. Completion time of the Postmark v1.51 benchmark (minutes)

on these devices. Such volumes include physical disk partitions as well as aggregate volumes such as software RAID and Logical Volume Manager volumes. Each storage volume usually contains a single file system or swap partition. In the second phase, TransPart searches through each storage volume to discover the available free blocks. Most modern file systems maintain a set of block allocation tables as meta-data on disk. TransPart utilizes file system on-disk semantics to properly parse the file system meta-data and to discover free disk blocks. Once TransPart has discovered free disk blocks, it allocates a TransPart device for the guest VM. An ext4 filesystem is created inside the TransPart device, and this filesystem is used to store parcel data demand-fetched from the cloud. All data written to the TransPart device is encrypted, so that a user never leaves behind residual state that could compromise his privacy.

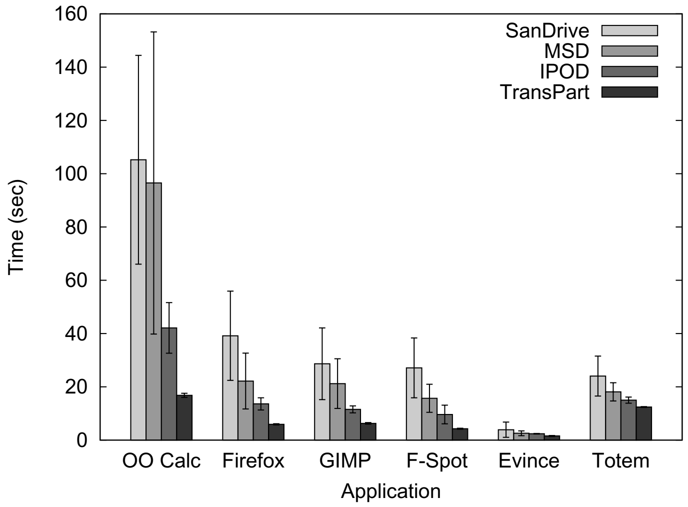


Fig. 6. Application launch latency time for 6 common desktop applications (seconds)

Our experiments show that TransPart can significantly improve the I/O performance of a Transient PC. For example, Figure 5 shows the performance of the well-known Postmark benchmark [15] on typical desktop hardware. The graph shows that TransPart offers significant performance improvement over the portable storage alternatives shown in Table 1. Figure 6 shows launch latency for a number of common Linux applications on the same Transient PC. Once again, the use of TransPart provides a significant improvement in user-visible performance. Experimental details of the results shown in Figures 5 and 6, as well as additional implementation details on TransPart are available in related technical reports [16,17].

4 Closing Thoughts

The convergence of cloud computing and mobile computing is the latest chapter in a long-running dialectic between centralization and decentralization in system design. Over the past 50 years, the pendulum has swung back and forth between these extremes. Today, cloud computing represents a thrust in which the forces of centralization are ascendant. Mobile computing, on the other hand, represents a thrust in which the forces of decentralization dominate. Where these tectonic forces meet, there will inevitably be a lot of heat generated and, hopefully, also some light. In this paper, we have put forth the view that use of virtual machine technology can lead to new "sweet spots" in the space of system architectures that try to reconcile the tradeoffs between centralization and decentralization. Notably, it yields the Transient PC computing model which preserves the centralization benefits of cloud computing without sacrificing mobility or usability.

Acknowledgements. This research was supported by the National Science Foundation (NSF) under grant numbers CNS-0509004, CNS-0833882 and CNS-0831268, and an IBM Open Collaborative Research grant. 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, IBM, Rutgers University, or Carnegie Mellon University.

References

1. Morris, J.H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S., Smith, F.D.: Andrew: a Distributed Personal Computing Environment. Commun. ACM 29(3), 184–201 (1986)
2. Satyanarayanan, M., Howard, J.H., Nichols, D.A., Sidebotham, R.N., Spector, A.Z., West, M.J.: The ITC Distributed File System: Principles and Design. In: Proceedings of the 10th ACM Symposium on Operating Systems Principles, Orcas Island, WA (December 1985)
3. Satyanarayanan, M.: Scalable, Secure, and Highly Available Distributed File Access. IEEE Computer 23(5), 9–18, 20–21 (1990)
4. Amazon Inc.: Amazon Web Services - Elastic Compute Cloud, `http://aws.amazon.com/ec2/`
5. Ammons, G., Bala, V., Berger, S., Da Silva, D.M., Doran, J., Franco, F., Karve, A., Lee, H., Lindeman, J.A., Mohindra, A., Oesterlin, B., Pacifici, G., Pendarakis, D., Reimer, D., Ryu, K.D., Sabath, M., Zhang, X.: RC2: A living lab for cloud computing. IBM Research Report RC24947, IBM (2010)

6. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The Eucalyptus Open-Source Cloud-Computing System. In: CCGRID 2009: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (2009)
7. Kistler, J.J., Satyanarayanan, M.: Disconnected Operation in the Coda File System. ACM Transactions on Computer Systems 10(1) (February 1992)
8. Kozuch, M., Satyanarayanan, M.: Internet Suspend/Resume. In: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, Callicoon, NY (June 2002)
9. 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., Lagar-Cavilla, H.A.: Pervasive Personal Computing in an Internet Suspend/Resume System. IEEE Internet Computing 11(2) (2007)
10. Sapuntzakis, C., Chandra, R., Pfaff, B., Chow, J., Lam, M., Rosenblum, M.: Optimizing the Migration of Virtual Computers. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, MA (December 2002)
11. Chandra, R., Zeldovich, N., Sapuntzakis, C., Lam, M.: The Collective: A Cache-Based System Management Architecture. In: Proceedings of the Second Symposium on Networked Systems Design and Implementation (May 2005)
12. MokaFive, Inc.: MokaFive Home Page, `http://www.mokafive.com`
13. Lagar-Cavilla, H.A., Tolia, N., de Lara, E., Satyanarayanan, M., O'Hallaron, D.R.: Interactive Resource-Intensive Applications Made Easy. In: Cerqueira, R., Pasquale, F. (eds.) Middleware 2007. LNCS, vol. 4834, pp. 143–163. Springer, Heidelberg (2007)
14. Lagar-Cavilla, H.A., Tolia, N., Satyanarayanan, M., de Lara, E.: VMM-Independent Graphics Acceleration. In: Proceedings of the 3rd ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE), San Diego, CA (2007)
15. Katcher, J.: PostMark: A New File System Benchmark. Technical Report TR3022, Network Appliance (1997)
16. Smaldone, S., Harkes, J., Iftode, L., Satyanaryanan, M.: Safe Transient Use of Local Storage for VM-based Mobility. Technical Report CMU-CS-10-110, School of Computer Science, Carnegie Mellon University (2010)
17. Gilbert, B., Goode, A., Satyanarayanan, M.: Pocket ISR: Virtual Machines Anywhere. Technical Report CMU-CS-10-112, School of Computer Science, Carnegie Mellon University (2010)