2010 34th Annual IEEE Computer Software and Applications Conference Workshops

# The Case for Content Search of VM Clouds

Mahadev Satyanarayanan*, Wolfgang Richter*, Glenn Ammons†, Jan Harkes* and Adam Goode*
*Computer Science Department
Carnegie Mellon University, Pittsburgh, PA 15213
Email: {satya,wolf,jaharkes}@cs.cmu.edu, agoode@andrew.cmu.edu
†IBM Research
IBM Watson Research Center, Yorktown Heights, New York
Email: ammons@us.ibm.com

***Abstract*—The success of cloud computing can lead to large, centralized collections of virtual machine (VM) images. The ability to interactively search these VM images at a high semantic level emerges as an important capability. This paper examines the opportunities and challenges in creating such a search capability, and presents early evidence of its feasibility.**

***Keywords*- data-intensive computing; discard-based search; forensic search; provenance; Diamond; cloud computing; virtual machines; VCL; RC2; EC2; Internet Suspend/Resume; ISR; deduplication; content-addressable storage**

## I. Introduction

As cloud computing gains momentum, collections of virtual machine (VM) images are starting to grow within clouds. These VM images embody precious information that is incidental to their primary purpose. For example, they contain specific versions of dynamically linked libraries (DLL's), tool chain versions, intermediate states of data transformation, persistent cache state, browsing bookmarks, and many other silent witnesses to the precise state of the world at the moment when the VM image was created. The ability to search this frozen state could be valuable in tasks such as debugging and troubleshooting, establishing the provenance of data or code, criminal forensics, and quality control. In this paper we examine issues relevant to creating such a search capability and sketch the outline of an implementation.

## II. Large VM Clouds

The term *cloud computing* means many things to many people. This paper focuses on approaches that encapsulate execution state in a VM image. For brevity, we use the vendor-neutral and format-independent term *parcel* to refer to such encapsulated state. A parcel may contain both volatile state (a memory image) and persistent state (a disk image), or just persistent state.

Figure 1 characterizes this computing space along two dimensions. The "Parcel Storage" dimension shows where parcels are stored when not in active use; the "Parcel Execution" dimension shows where they run.

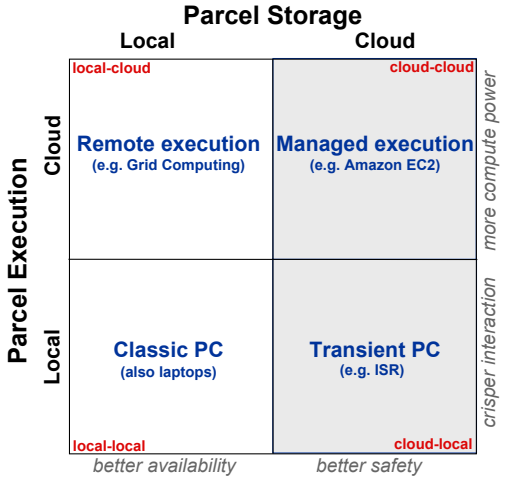



Figure 1: Taxonomy of VM-based Cloud Computing

Our interest is in the two right quadrants, which contain approaches that store parcels in the cloud. The top right quadrant contains approaches such as Amazon's EC2 [1], IBM's Research Compute Cloud (RC2) [2], and the open source Eucalyptus infrastructure [3], where parcel storage and execution both happen within the cloud. The bottom right quadrant contains approaches such as the Internet Suspend/Resume® system (ISR) [4], [5], the Stanford Collective [6], [7], and MokaFive [8], where parcels are stored in the cloud but execute on a computer that is close to the user.

Collections of parcels in the cloud tend to be large and long-lived. For example, an enterprise that deploys ISR will add one parcel per user per day if it creates a daily snapshot of each user's parcel for backup and disaster recovery. While strict retention policies can slow this growth, the underlying dynamics are biased towards expansion of parcel collections. This is shown in Table I, which provides data on users and parcels from three clouds with diverse architectures, user communities, and workloads.

RC2, which researchers at IBM use for computing resources and as a testbed, fits into the top right quadrant of Figure 1 ("Managed Execution"). In this

978-0-7695-4105-1/10 $26.00 © 2010 IEEE
DOI 10.1109/COMPSACW.2010.97

| Cloud | Organization | Current Users | Current Parcels | Previous Users | Previous Parcels | Duration |
|---|---|---|---|---|---|---|
| RC2 | IBM Research | 126 | 444 | 47 | 122 | 9 months |
| VCL | North Carolina State Univ. | 13,334 | 523 | 12,121 | 494 | 12 months |
| ISR | Carnegie Mellon Univ. | 56 | 1,347 | 48 | 1,971 | 12 months |

Table I: Operational Characteristics and Growth of Three Clouds

system, most users create new parcels, which contain customized software stacks, by extending standard parcels that are provided by RC2's administrators. Once created, a parcel can be used, set aside for an extended period of time, then reactivated, and so on. This leads to a usage model in which each user tends to have a few parcels (444 parcels for 126 users). Over a 9-month period since its inception, RC2 shows substantial growth in users (126 from 47) and parcels (444 from 122).

The VCL cloud, which is used by instructors and students in the North Carolina State University system, also maps to the top right quadrant of Figure 1. Unlike RC2, most users cannot create new parcels: only instructors and system administrators can do that. When a student executes a parcel, he or she creates a new, temporary VM whose state cannot be saved in the cloud. Students store their long-term state (such as user files) outside the parcel, typically in a distributed file system such as AFS [9] or on detachable storage. Since VCL is in production use across the entire state university system, Table I shows that VCL has a very large number of users (13,334). However, the tight control on creation of new parcels results in only slightly more parcels in VCL than in RC2 (523 versus 444). Over a 12-month period, there has been modest growth in users (13,334 from 12,121) and parcels (523 from 494).

The ISR cloud at Carnegie Mellon maps to the bottom right quadrant of Figure 1 ("Transient PC"), and is the smallest of the three clouds in users (56). However, it is by far the largest in parcels (1,347) because a user can create a new parcel from a snapshot of his VM at any time. Over a 12-month period, there has been modest growth in users (56 from 48), but a significant decrease in parcels (1,347 from 1,971) due to purging and cleanup of inactive users and their parcels. Without purging and cleanup, there would have been 84 users and 3,059 parcels.

The growth of VM clouds poses a new challenge for administrators and users: how does one identify parcels that are relevant to a specific use context? This leads to the need for a search capability for VM clouds, which we examine in the next section.

## III. Content Search of Parcels

If efficient content search of parcels were possible, what new usage scenarios would this enable? We answer this question by describing a number of hypothetical use cases below. From these use cases, we deduce the requirements and constraints imposed upon the design of a search capability. We focus here on technical issues and ignore the broader legal and ethical issues, many pertaining to privacy, that will have to be addressed in any real-world deployment of this search capability.

- A bug has just been discovered during the top-level integration of components developed by a large and geographically distributed software development team. This bug is caused by a bad programming practice in one developer's code. Does anyone else in his group share this same bad practice? Can one check work still in progress to catch the problem as early as possible? Can one search recent parcels of team members to discover such instances?
- A graphic arts company is accused of copyright infringement because one of the photographs that it sells appears to be a derivative of someone else's copyrighted photograph. Can one search the company's parcels to discover instances of the original photograph or of intermediate stages of its transformation into the suspicious product?
- A company wants to ensure that its employees use only legally licensed software, that viruses have not infected its parcels, and that it complies with regulations governing the use of confidential data.
- A developer keeps snapshots of the machine on which he or she works. A user reports an unusual, hard to reproduce failure that the developer remembers seeing while working on an unrelated feature. The developer's search for a parcel that can demonstrate the failure is an ad-hoc combination of queries for commands and web-page visits associated not with the failure, but with the feature.
- An application is to be deployed to Amazon EC2. The application relies on standard services and software, which the deployer must configure properly for the EC2 environment. The deployer finds "best practice" examples by searching for parcels in EC2 that use the same services and software in a similar way.
- A scientific community builds a cloud-based parcel collection that contains the results and infrastructure behind its papers. A researcher has an idea, and searches the collection for related work—

including the experimental setup encapsulated in parcels.

Several requirements for the search capability emerge from these examples. First, it must be possible to search the contents of parcels and not just meta-data such as parcel ownership, modification time, or operating system type. Second, the use cases require domain-dependent search queries. In the software example, a query is best expressed in terms of programming-language constructs; by contrast, in the photograph example, a query is best expressed in terms of primitives such as regions of specific color, regions of specific texture, human faces, specific animals, and so on. Simply viewing parcel contents at a low semantic level (such as regular expression search on ASCII strings) will not lead to satisfactory results. Third, the high semantic content of these searches implies direct involvement of a human expert in the search process, including many iterative refinements of the search query based on partial results. This is in contrast to data mining, where human involvement comes only at the beginning and the end of a complete data scan.

## IV. Discard-based Search as the Foundation

Since a parcel encapsulates arbitrary computing state, the need to search its contents at a deep semantic level with a human in the loop strongly suggests that classic indexed search is unlikely to be satisfactory. Indexed search, which is the dominant search metaphor today, is predicated on the ability to precompute suitable indexes for all conceivable queries that may be posed on the data. Interpreted in its full generality, this assumption implies an omniscience about the queries that may be posed in the future. In practice, real systems today severely limit the generality and semantic depth of queries to that of available indexes. The difficult challenges of indexing for queries of high semantic content have long been investigated by the knowledge retrieval community [10], [11]. Unfortunately, there have been no breakthrough advances to date.

To address these limitations, there has been recent work towards a fundamentally different search metaphor called *discard-based search* [12], [13]. This approach defers search computation until the specifics of the query are available rather than trying to precompute in advance of queries. This allows a query to be expressed as executable code on the raw data, rather being restricted by a declarative query language such as SQL. Further, the expertise, judgment, and intuition of the user performing the search can then be brought to bear on the specificity and selectivity of the current search. In contrast, indexed search limits even experts to the quality of the preprocessing that produced the index.

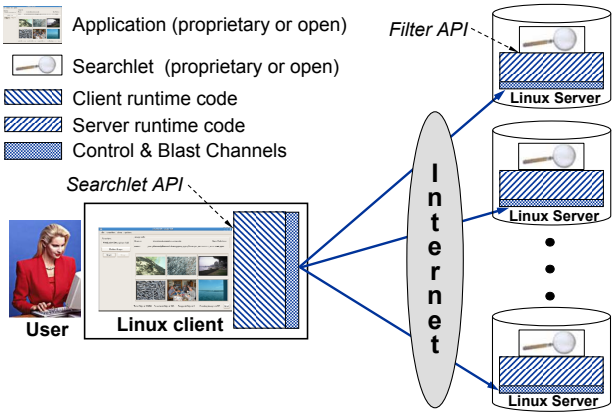



Figure 2: Discard-based Search Architecture

These unique strengths of discard-based search make it a good fit for the requirements presented in Section III.

Two optimizations can be used in discard-based search to ensure an adequate rate of return of results to an interactive user. First, high degrees of easily-exploited parallelism can be used. Second, temporal locality in search queries can be exploited by persistent caching of search results. This can be viewed as a form of *just-in-time indexing* that occurs incrementally as a transparent side-effect of user activity.

The *OpenDiamond® platform* is an open source implementation of Linux middleware for discard-based search [14]. Its extensible architecture, shown in Figure 2, cleanly separates the domain-specific and domain-independent aspects of the problem. User interaction typically occurs through a domain-specific GUI. The domain-specific code that performs early discard on servers is called a *searchlet.* It is typically composed of individual components called *filters.* For example, an image search application may provide filters for face detection, color detection and texture detection. Filters embody specific dimensions of knowledge, while searchlets express a multi-dimensional search query as a precedence graph of parametrized filters. The OpenDiamond platform also supports filters written in MATLAB [15] or as ImageJ macros [16], [17].

A searchlet is composed and presented by the application through the Searchlet API, and is distributed to all of the servers involved in the search task. Each server iterates through the locally-stored objects in a system-determined order and presents them to filters for evaluation through the Filter API. Each filter can independently discard an object. A server is ignorant of the details of filter evaluation, only caring about the scalar return value that is thresholded to determine whether a given object should be discarded or passed to the next filter. Only those objects that pass through the entire

| Version | Size (MB) | Files Same | Bytes Same | Release Date | Days Stale |
|---|---|---|---|---|---|
| 2.6.33.1 | 353 | 100% | 100% | 03/15/2010 | 0 |
| 2.6.33.0 | 353 | 99% | 98% | 02/24/2010 | 19 |
| 2.6.32.9 | 341 | 71% | 49% | 02/23/2010 | 20 |
| 2.6.31.12 | 326 | 56% | 33% | 01/18/2010 | 56 |
| 2.6.30.10 | 315 | 47% | 27% | 01/06/2010 | 68 |

Table II: Linux Kernel Source Tree Similarity

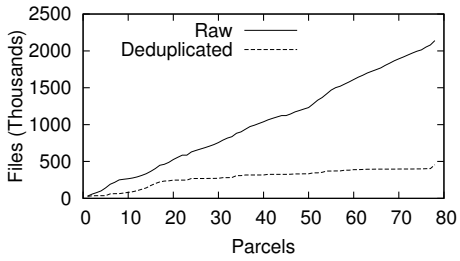
gauntlet of filters in the searchlet are transmitted to the client. Details on design and implementation of the OpenDiamond platform can be found elsewhere [18].

## V. Exploiting Data Similarity

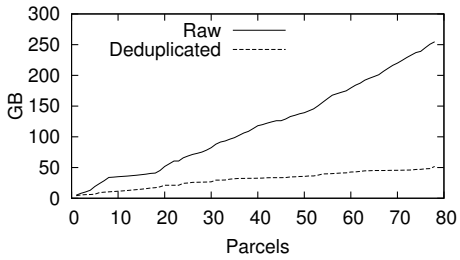
The large size of VM's (typically many GB) is a major consideration when implementing any VM-based application. Discard-based search of parcels is no exception. By definition, discard-based search involves runtime disk I/O to read the contents of each parcel, and processing overhead to apply a searchlet to this large amount of data. For a large collection of parcels, the total runtime I/O and processing can be intolerably large. Any performance optimization that could reduce this cost is valuable. We observe that *similarity of data content across parcels* is the source of such a performance optimization—a file that recurs in many parcels only needs to be examined once in any search. In addition, there can be significant storage savings through deduplication [19].

There are at least three different reasons for data similarity across parcels. First, many parcels use the same guest OS and applications. For example, if the parcels of two users both use Windows XP Service Pack 3 as the guest there are likely to be many executables, DLL's, and system configuration files that are identical in the two parcels. This first reason applies to RC2 and VCL, where users draw from a set of core parcels in a central repository. They can be customized, but there will be significant overlap. Second, a user typically modifies only a small part of his VM state at any time. Hence, snapshots of his parcel that are taken over a modest interval of time (i.e., typical durations of hours or days), are likely to have many regions that are unchanged. This second reason applies to ISR, where users often create daily snapshots of their parcels. Third, many files remain unmodified across successive releases of software. This last reason applies to all three clouds (RC2, VCL and ISR), because it is intrinsic to software evolution and independent of cloud design.

Table II shows the number of files that are identical in different releases of the Linux Kernel. Between the source code trees of Linux Kernel versions 2.6.33.0 and 2.6.33.1 that were released almost three weeks apart (19 days), nearly 99% of the files are identical. The total size of these identical files amount to 98% of the




(a) Number of Distinct Files




(b) Bytes in Distinct Files

Figure 3: Deduplication of VCL Parcels

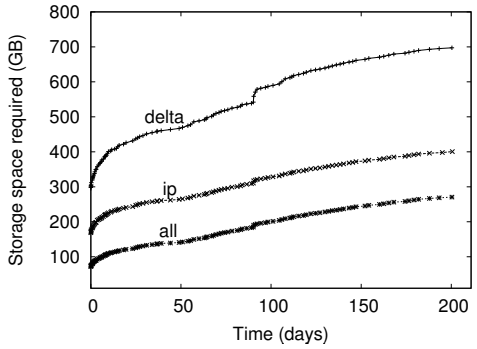



Figure 4: Deduplication of ISR Parcels (Nath et al [21])

total source code size in bytes. Even versions that are more than two months apart show substantial similarity: between versions 2.6.30.10 and 2.6.33.1, nearly 47% of the files are identical. These files account for nearly 27% of total source tree size in bytes. As reported by Tolia et al [20], other versions of the Linux kernel including the 2.2 series have significant numbers of identical files across versions. Extrapolating from these examples, we conjecture that inter-version data similarity is a property of all mature software systems.

The cumulative effect of these diverse sources of data similarity can be substantial. Figure 3 shows the impact of deduplication at the whole-file level on 78 VCL parcels based on Windows XP. The number of files with distinct content grows much more slowly than the total number of files. Figure 3(a) shows less than 500 thousand distinct files out of two million total files in

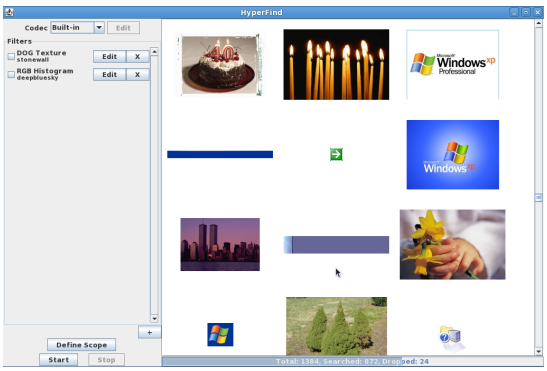



Figure 5: Searching JPEG Files within Parcels

78 parcels—almost all from the production NCSU VCL environment. The storage capacity and I/O bandwidth savings from deduplication of these files is substantial: 50 GB rather than 250 GB, as shown in Figure 3(b).

As reported by Nath et al [21], deduplication is highly effective in ISR parcels. Figure 4, based on data from a 25-user ISR-3 cloud in 2005, shows the potential benefits of deduplication in ISR. The curve labeled "delta" shows the growth of total server storage if all modified chunks of a parcel are saved at each snapshot. The curve labeled "ip" shows storage growth if deduplication is done strictly within successive versions of a parcel. The curve labeled "all" shows storage growth if deduplication is done across all versions of all parcels.

## VI. Preliminary Evidence of Feasibility

We have begun work towards the creation of a search capability for VM clouds along the lines described in this paper. Our design leverages a parcel storage engine called *Mirage* [22] from IBM Research that was created to control VM image sprawl in clouds. Mirage parses the virtual disk partitions in a parcel, then extracts and deduplicates individual files from those partitions. It is currently able to parse Linux file system formats such as `ext2` and `ext3`, as well as the Windows NTFS format. We have extended the server backend of the OpenDiamond platform to redirect file requests from searchlets to Mirage rather than obtaining those files from the server's local file system. Our extension incorporates three critical pieces of infrastructure: a data source abstraction layer, a database containing file-level semantics, and a scoping mechanism to limit search queries to files of certain types and to provide access control. This extension to the OpenDiamond platform enables a seamless integration of discard-based search and data stored within VM clouds.

Although our implementation is incomplete, there is enough functional for a proof of concept. Figure 5 shows a screenshot from an OpenDiamond application that enables users to search images in various formats such as JPEG, TIFF, and PNG. A user specifies a search query by selecting one or more filters from a predefined set (for example, a human face filter) or by defining by example a filter for a simple property such as color or texture. The user's experience in using the application is indistinguishable from searches on data stored on local file systems. The application only has to search $20,167$ unique files (671 MB) as opposed to $100,153$ (2 GB) raw files in the 78 VCL parcels depicted in Figure 3 due to the benefits provided by deduplication. We have also begun work towards implementing some of the applications that were alluded to in Section 3 such as source code search, and virus search.

## VII. Conclusion

The growth of cloud computing will lead to the emergence of large VM clouds that encapsulate valuable computing state. As these collections expand, it will become increasingly important to be able to search their content at a high semantic level. In this paper we have shown why discard-based search, originally developed for interactive search of complex non-indexed data on server disks, is valuable to extend to VM clouds. In making this extension, deduplication is an important storage optimization to leverage for scalability. Based on positive preliminary evidence of feasibility, we are now working towards a full implementation of the ideas described in this paper.

## Acknowledgments

Vas Bala of IBM Research suggested the idea of applying discard-based search to VM images. We gratefully acknowledge Peng Ning and Xianqing Yu of North Carolina State University for sharing their collection of VM images with us, and providing us with data on the VCL cloud. Herb Lee of IBM Research helped us obtain usage data on the RC2 cloud. This research was supported by the National Science Foundation (NSF) under grant number CNS-0614679, 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, or Carnegie Mellon University.

## References

[1] Amazon Inc., "Amazon Web Services - Elastic Compute Cloud," http://aws.amazon.com/ec2/. [Online]. Available: http://aws.amazon.com/ec2/

[2] G. Ammons, V. Bala, S. Berger, D. M. Da Silva, J. Doran, F. Franco, A. Karve, H. Lee, J. A. Lindeman, A. Mohindra, B. Oesterlin, G. Pacifici, D. Pendarakis, D. Reimer, K. D. Ryu, M. Sabath, and X. Zhang, "RC2: A living lab for cloud computing," IBM, IBM Research Report RC24947, 2010.

[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009.

[4] M. Kozuch and M. Satyanarayanan, "Internet Suspend/Resume," in *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, Callicoon, NY, June 2002.

[5] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, A. Surie, D. R. O'Hallaron, A. Wolbach, J. Harkes, A. Perrig, D. J. Farber, M. A. Kozuch, C. J. Helfrich, P. Nath, and H. A. Lagar-Cavilla, "Pervasive Personal Computing in an Internet Suspend/Resume System," *IEEE Internet Computing*, vol. 11, no. 2, 2007.

[6] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, and M. Rosenblum, "Optimizing the Migration of Virtual Computers," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec. 2002.

[7] R. Chandra, N. Zeldovich, C. Sapuntzakis, and M. Lam, "The Collective: A Cache-Based System Management Architecture," in *Proceedings of the Second Symposium on Networked Systems Design and Implementation*, May 2005.

[8] "MokaFive Home Page," http://www.mokafive.com.

[9] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, "Scale and Performance in a Distributed File System," *ACM Transactions on Computer Systems*, vol. 6, no. 1, February 1988.

[10] T. Minka and R. Picard, "Interactive Learning Using a Society of Models," *Pattern Recognition*, vol. 30, 1997.

[11] A. Yao and F. Yao, "A General Approach to D-Dimensional Geometric Queries," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, May 1985.

[12] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. Ganger, E. Riedel, and A. Ailamaki, "Diamond: A Storage Architecture for Early Discard in Interactive Search," in *Proceedings of FAST 2004*, San Francisco, CA, April 2004.

[13] M. Satyanarayanan, R. Sukthankar, L. Mummert, A. Goode, J. Harkes, and S. Schlosser, "The unique strengths and storage access characteristics of discard-based search," *Journal of Internet Services and Applications*, vol. 1, no. 1, 2010.

[14] "OpenDiamond Home Page," http://opendiamond.cs.cmu.edu.

[15] "MATLAB Home Page," http://www.mathworks.com/.

[16] "ImageJ Home Page," http://rsbweb.nih.gov/ij/.

[17] W. Burger and M. J. Burge, *Digital Image Processing*. Springer, 2008.

[18] M. Satyanarayanan, R. Sukthankar, A. Goode, N. Bila, L. Mummert, J. Harkes, A. Wolbach, L. Huston, and E. de Lara, "Searching Complex Data Without an Index," Computer Science Department, Carnegie Mellon University, Tech. Rep. CMU-CS-09-178, December 2009.

[19] K. Jin and E. L. Miller, "The effectiveness of deduplication on virtual machine disk images," in *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, Haifa, Israel, 2009.

[20] N. Tolia, J. Harkes, M. Kozuch, and M. Satyanarayanan, "Integrating Portable and Distributed Storage," in *Proceedings of the 3rd Usenix Conference on File and Storage Technologies*, San Francisco, CA, March 2004.

[21] P. Nath, M. Kozuch, D. O'Hallaron, J. Harkes, M. Satyanarayanan, N. Tolia, and M. Toups, "Design Tradeoffs in Applying Content Addressable Storage to Enterprise-Scale Systems Based on Virtual Machines," in *Proceedings of the USENIX Technical Conference*, Boston, MA, June 2006.

[22] D. Reimer, A. Thomas, G. Ammons, T. Mummert, B. Alpern, and V. Bala, "Opening Black Boxes: Using Semantic Information to Combat Virtual Machine Image Sprawl," in *VEE'08: Proceedings of the 2008 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, Seattle, WA, March 2008.