VStore++: Virtual Storage Services for Mobile Devices

Sudarsun Kannan, Karishma Babu, Ada Gavrilovska, and Karsten Schwan

Center for Experimental Research in Computer Systems
Georgia Institute of Technology
{sudarsun,karishma,ada,schwan}@cc.gatech.edu

Abstract. This paper addresses media sharing via an approach that offers 'fungible' storage, where storage services implement virtual stores that are dynamically mapped to suitable 'nearby' or otherwise available physical devices. In particular, the novel VStore++ system provides seamless and flexible data storage, access, and sharing services, by exploiting virtualization technology to aggregate and make use of both 'nearby' and private storage (e.g., in a mobile user's home), and public storage resources offered on remote cloud platforms.

Key words: mobile virtualization, cloud computing

1 Introduction

Mobile devices with their increased CPU speeds, core counts, memory sizes, and improved communication rates may well become the next generation personal computers. To meet the resulting increased end user demands for rich and diverse types of services on these platforms, however, industry must address constraints that include issues with battery life, processing capabilities dwarfed by those of server systems, limited storage, smaller display form factors, and others. In response, our research is exploiting the fact that mobile devices are often surrounded by and used in contexts where there are many other resources that could enhance their capabilities. Consider, for instance, the enormous aggregate processing power and storage capacity available in say, a soccer stadium in the forms of other spectators' devices, the server systems supporting broadcast and organizational functions, and devices engaged in ancillary tasks like security. Another example are users' homes where there may be home PCs, laptops, and computerized home entertainment systems. Further, often associated with such resources is locally captured state like home videos, security images, or the context information needed to distinguish important from less important content.

Locally available resources and context suggest solutions that use distributed, multi-device service implementations. We formulate the following simple principles for mobile service realization and delivery:

- *Fungibility for dynamic flexibility*: physical resources should be 'fungible', so as to create dynamic options in the mappings from the resources applications

believe they are using – virtual resources – to the physical resources actually being used.

- *Explicit cooperation*: devices must agree to participate in service delivery, creating sets of cooperative devices operating in common domains (e.g., a user's home).
- *Guided active management*: since the 'best' mappings of virtual to physical resources depend on current context, user needs, and resource availabilities, active management of these mappings must have continuous inputs from methods that monitor these factors.
- *Automation and independence*: guided management should not require end user participation, i.e., it should be automated, and in addition, management should be independent of specific operating systems or application frameworks being present on mobile devices.
- *Universal operation*: managed services should function wherever mobile devices are used, which implies that there must also be ways to store and access global state across disconnected periods of operation. Access to Internet-based services, therefore, is a critical element of any solution for fungible mobile services.

Current solutions that 'simply use the Internet' are insufficient. Although they clearly enhance mobile devices via remote (e.g., for storage, DropBox, Gmail, etc.), they are lacking in terms of independence and universal operation. This is because of (1) disconnected operation – where sometimes, mobile devices will not be able to access Internet services like those offered by public cloud infrastructures,(2) undue communication overheads or costs – referring to the performance (or lack thereof) or the expense of reaching Internet resources in comparison to the lower costs of using local resources and exploiting locally available state, and (3) data privacy or security, for which it may be preferable to use and maintain local resources that cannot be accessed by others.

The VStore++ service described in this paper provides efficient, fungible storage for mobile devices. This is done by exploiting distributed storage via a 'cooperative' model in which devices choose to participate in joint data storage and access. Interactions may take place across wireless networks, as in the aforementioned case of the soccer stadium, across the Internet, when using Internet-based resources like cloud storage, or across wired links when the mobile device operates in a user's home. The outcome is a 'personal mobile cloud' comprised of a dynamically varying set of interacting devices that cooperate to provide end users with seamless storage services.

The implementation of VStore++ attains fungibility and independence by operating at the virtualization level, where it can use local disks, remote machines' stores, or even Internet-connected storage in ways that are transparent to and independent of end users, application frameworks, and even the operating systems running on mobile devices. At the same time, since the context in which the mobile device operates will change dynamically, as will end user requirements, VStore++ will track resource availability in order to direct requests to whichever resource is currently accessible, using a global index maintained

during its operation. This is done in ways that maintain user-defined data access controls. Further, VStore++ as a service can be used wherever there is connectivity to participating devices and using whichever connectivity methods are currently available, but automation in terms of making such choices or determining suitable storage targets remains subject of our future work. The outcome is a storage service accessible from a wide range of platforms, independent and potentially decoupled from their inherent constraints.

VStore++ has been implemented on Atom-based machines that are virtualized with the Xen open source hypervisor. Its evaluation uses a prototypical 'home' setup in which there is cooperation among mobile devices, tethered home machines like PCs, and remote services like Amazon's EC2 storage. Our ongoing work is exploring mobile services other than storage, is developing methods for automatically guided management, and is improving the ubiquity by inter-operating by use of other service delivery means, such via cable or satellite TV-based connections. Performance results reported in this paper clearly demonstrate the utility of fungibility, showing much improved performance when storage resources are local vs. in public clouds and showing advantages with local aggregation when there is substantial local state or when there are requirements for fast local response (as with home security systems, for instance).

2 VStore++ Architecture and Implementation

To provide end-users with mobile devices seamless access to state stored on diverse locally present storage resources, as well as remote, publicly available compute and storage cloud platforms, VStore++ must implement transparent enforcement of varying sharing policies on objects, and search for content belonging to other domains, both local and remote, via standard content access interfaces (e.g., the file system interface) In addition, there must be interfaces for dynamically associating with data accesses additional functionalities such as trust management, access control, location attributes, and methods for data manipulation or customization. VStore++ attains these implementation goals using as its service interface that of an object-based file system, virtualized for use by guest domains. VStore++'s basic object API is enhanced with additional object-level metadata (e.g., privacy attributes) and with 'activity specifications' that make it possible to associate data manipulation functions with object accesses. VStore++ is implemented in a trusted service domain, which is 'dom0' in its Xen-based prototype.

The main components of VStore++, illustrated in Figure 1, are:

1. ***Virtualized object-based file system***: VStore++ is a virtualized storage service exposing an object-based file system interface. Internally, it uses a standard file system to represent objects, using a one-to-one mapping of objects to files. The current implementation is based on the PVFS object-based file system, but alternative, more lightweight implementations are currently under consideration (e.g., based on SyncFS).

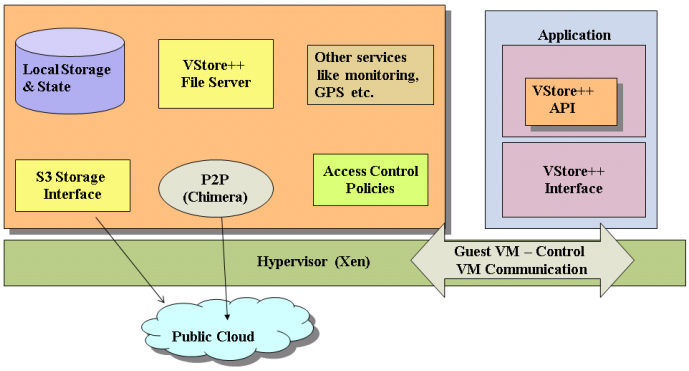

Fig. 1. VStore++ Architecture

2. ***Peer-to-peer service***: this layer provides the facilities for establishing peer-to-peer overlays, for object searches, for request routing and for dynamic overlay management (e.g., participant discovery, etc.) in ad hoc mobile environments. The current implementation is based on the lightweight Chimera peer-to-peer overlay system. In order to support robust and efficient tag-based routing services, as needed by VStore++ to enable discovery of and access to remotely stored tagged data, a scalable DHT layered on top of Chimera's key based routing service provides file system-like semantics, i.e., insert and retrieve operations. The DHT layer also provides additional features, including caching, replication, node failure detection, and handling of new node arrivals.
3. ***Enhancement services***: additional services associated with VStore++, at runtime, can provide functionality that includes (i) run-time trust mechanisms [2], (ii) methods that protect data via data manipulation functions that range from simple read/write permissions to content-dependent processing like watermarking for images, etc., or (iii) location mechanisms that operate with or without GPS hardware support, by using approximate GPS coordinates obtained via services such as Georgia Tech's WhereAmI or mechanisms such as WiFi triangulations, marker recognition, or others.
4. ***Interface to public/remote clouds***: in order to seamlessly enable access to state or services available on 'nearby' vs. remote resources, such as those present in current public cloud platforms like Amazon's EC2, VStore++ transparently integrates corresponding cloud client components. The same metadata services which keep track of location of 'nearby' content, are used to encode operations and accompanying attributes needed for access to the remote cloud storage.

3 Experimental Evaluation

For brevity, this section outlines only some of the experimental results of our research, with the dual goals of illustrating the feasibility of supporting virtualized

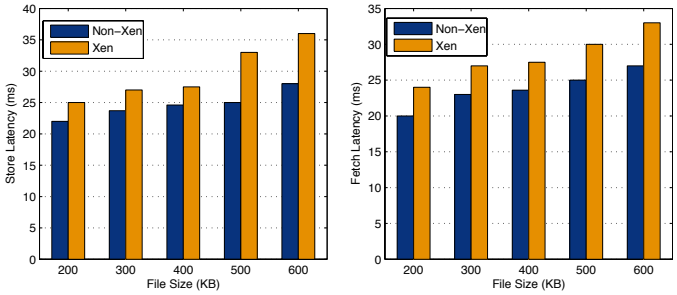

Fig. 2. VStore++ virtualization overheads.

file sharing solutions such as VStore++ on mobile platforms and demonstrating the potential advantages of such a solution compared to pure 'Internet'-based methods. Experiments are conducted using the current VStore++ implementation on Xen-3.0.4 as the virtualization platform, with PVFS-2.6.3 as the object-based storage service, and Chimera 1.20 for DHT and peer-to-peer communication services. The experimental testbed consists of dual-core 1.66GHz Intel Atom N280 netbooks and multiple 3 GHZ 64-bit core-duo laptops, running Linux 2.6.16 on Xen. Access to Amazon S3 cloud storage is via a home-based broadband link, to better emulate future wireless bandwidths.

To determine the overheads due to virtualization, we compare the virtualized VStore++ implementation with one that runs on a non-virtualized system, where the client directly interacts with the 'back-end' server. The graphs in Figure 2 show that the use of Xen contributes from 4.5% to 15.74% to the data access latencies of VStore++'s operations. In large part, these are due to our use of TCP sockets for VM-dom0 communication. It is known that such costs can be reduced substantially by using shared memory based VM-VM communication mechanisms already available in the open source community. At the same time, the moderate costs of this un-optimized implementation enable functionality and flexibility that is otherwise not easily attainable – storage fungibility, independence, and universal operation, as well as the ability to transparently extend the data access service with additional functionality, such as location transparency, rich access control policies, useful data manipulations (e.g., for privacy protection), etc.

Additional experiments evaluate the impact of virtualization on sustainable throughput rates, with results indicating that for smaller file sizes (200 KB) virtualization impacts performance by no more than 10%. As file sizes increase, particularly as the distributed store becomes more full, overheads increase. Such costs are accompanied, however, by substantially increased total storage capacity and flexibility. Overheads can be reduced through additional optimizations of Xen for small form factor devices, and by using additional hardware-level virtualization support available on these (Atom-based) devices.

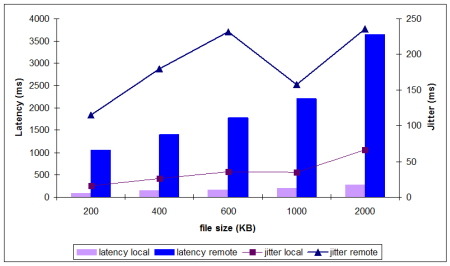

Fig. 3. Benefits of service delivery via local vs. remote entities.

Finally, to demonstrate the importance of enabling access to local- vs. remote 'cloud' services, we compare between the latencies and response time variability between accesses to 'nearby' storage (i.e., in our prototypical 'home' environment) vs. to Amazon's EC2 storage service. The measurements in Figure 3 show that access to local storage can not only be seamlessly integrated under the VStore++ interface, but that it also results in superior access properties (e.g., response time and jitter), which may be critical for certain types of services.

4 Conclusions and Future Work

VStore++ is an experimental vehicle for understanding and exploring ways to construct 'personal clouds' comprised of dynamic sets of mobile and stationary computing platforms. By using virtualization technology to access storage service at the object level rather than at the block level, VStore++ can associate useful semantic data with storage objects, such as privacy or access control metadata, and can then use such information to enforce diverse end user requirements. Performance evaluations show that the VStore++ hybrid solution of using both local and remote resources for data storage can be superior to the purely Internet-based service offering now available to mobile platforms, with additional benefits derived from the ability to deal with trust and data privacy concerns.

Our ongoing and future work is exploring two avenues toward further enriching mobile services. One direction of research is considering other services, such as those needed for multimedia delivery and customization, and for universal service operation in lieu of certain dynamic resource deficiencies. Another direction is to automate virtual to physical resource mappings under changing conditions, such as when a mobile device's battery become depleted.

References

1. Seshasayee, B., Narasimhan, N., Biljani, A., Pai, A., Schwan, K.: VStore - Efficiently Storing Virtualized State Across Mobile Devices. In MobiVirt'08 (2008).
2. Kong, J., Schwan, K.: ProtectIt: Trusted Distributed Services Operating on Sensitive Data. In EuroSys'08 (2008).