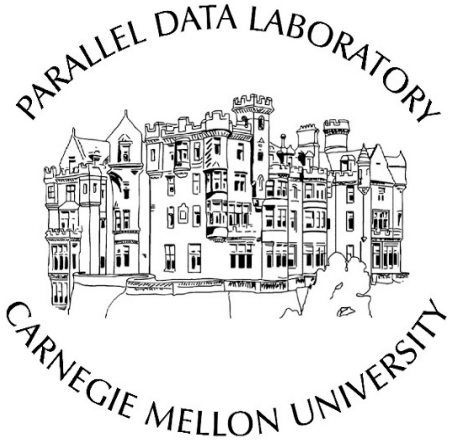


# Perspective: Semantic data management for the home

Brandon Salmon, Steven W. Schlosser*, Lorrie Faith Cranor, Gregory R. Ganger
*Carnegie Mellon University, *Intel Research Pittsburgh*

CMU-PDL-08-105
May 2008

**Parallel Data Laboratory**
Carnegie Mellon University
Pittsburgh, PA 15213-3890

## Abstract

*Perspective uses a new semantic filesystem construct, the view, to simplify management of distributed storage in the home. A view is a semantic description of a set of files, specified as a a query on file attributes. In Perspective, users can identify and control the files stored on a given device by examining and modifying the views associated with it. This approach allows them to reason about what is where in the same way (semantic naming) as they navigate their digital content. Thus, in serving as their own administrators, users do not have to deal with a second data organization scheme (hierarchical naming) to perform replica management tasks, such as specifying redundancy to provide reliability and data partitioning to address device capacity exhaustion. A set of extensive user studies confirm the difficulties created by current approaches and the efficacy of view-based data management.*

**Acknowledgements:** We thank the members and companies of the PDL Consortium (including APC, Cisco, EMC, Google, Hewlett-Packard, Hitachi, IBM, Intel, LSI, Microsoft, Network Appliance, Oracle, Seagate, Symantec, and VMware) for their interest, insights, feedback, and support. This material is based on research sponsored in part by the National Science Foundation, via grant #CNS-0326453, and by the Army Research Office, under agreement number DAAD19–02–1–0389. Brandon Salmon is supported in part by an Intel Fellowship. Special thanks to Rob Reeder, for all his help in setting up the user study.

**Keywords:** home storage, views, perspective, storage management, contextual analsysis, user studies

# 1 Introduction

Distributed storage is coming home. An increasing number of home and personal electronic devices create, use, and display digitized forms of music, images, videos, as well as more conventional files (e.g., financial records and contact lists). In-home networks enable these devices to communicate, and a variety of device-specific and datatype-specific tools are emerging. The transition to digital homes gives exciting new capabilities to users, but it also makes them responsible for administration tasks usually handled by dedicated professionals in other settings. It is unclear that traditional data management practices will work for "normal people" reluctant to put time into administration.

In this paper we present the Perspective[1] distributed filesystem as part of an expedition into this new domain for distributed storage. As with previous expeditions into new computing paradigms, such as distributed operating systems (e.g., [14, 17]) and ubiquitous computing (e.g., [27]), we are building and utilizing a system representing the vision in order to gain experience. In this case, though, we (the researchers) poorly represent most of the user population. Most will be non-technical people who just want to be users, but must (begrudgingly) deal with administration tasks or live with the consequences. Thus, various forms of organized user studies will be required as home storage applications evolve.

One early lesson from our contextual analyses is that home users do not organize and access their data via traditional hierarchical naming. Computing researchers have long talked about attribute-based data navigation (e.g., semantic filesystems [7, 24]), while continuing to use directory hierarchies, but it is common practice for home/personal storage. Popular interfaces (e.g., iTunes, iPhoto, and even drop-down lists of recently-opened Word documents) allow users to navigate file collections via attributes like publisher-provided metadata, extracted keywords, and date/time. Usually, files are still stored in underlying hierarchical file systems, but users are insulated from naming at that level and often have no idea where in the namespace given files end up.

Users have readily adopted these higher-level navigation interfaces, but difficulties arise when those same users need to use the underlying hierarchical namespace. Unfortunately, that is exactly what traditional administration mechanisms require. For example, partitioning data across computers for capacity management and keeping multiple copies of certain data for reliability require identifying specific collections of files. We use the term *replica management tasks* to refer to tasks that require understanding and manipulating the locations of data replicas (whether singular or multiple).

Current approaches to specifying policies for replica management tasks are tightly tied to hierarchical namespaces. Specifically, subtrees of the hierarchy are kept in "volumes", and decisions about handling of volumes are conveyed to the system. Whereas professional system administrators are very comfortable with this abstraction, home storage users struggle with it. In particular, they must overcome the disconnect between the data organizations they use normally and those involved with replica management tasks.

To correct the disconnect, the Perspective distributed filesystem replaces the traditional volume abstraction with a new primitive we call a view. A *view* is a compact description of a set of files, expressed much like a search query, and a device on which that data should be stored. For example, one view might be *"all files with type=music and artist=Aerosmith" stored on Brandon's iPod* and another *"all files with owner=Brandon" stored on Brandon's laptop*. Each device participating in Perspective maintains and publishes one or more views to describe the files that it stores. A user can see what is stored where by examining the set of views in the system, and she can control replication and data placement by changing the views of one or more devices. Since views describe sets of files using the same attribute-based style as their other tools, view-based management should be easier for users. Views also allow sets of files to overlap and to be described independently of namespace structure, reducing the need for users to worry about application-internal file naming decisions or unfortunate volume boundaries.

[1] In seeing many views, one gains Perspective.

This paper describes how views are supported in Perspective and used for replica management tasks. With the more flexible management abstraction, new algorithms are needed to reason about reliability and provide for permanence. We also describe a set of view-based management tools, collectively called Insight, for assisting users with those tasks. Although other underlying architectures could be used, Perspective supports its view-based management layer with a distributed storage layer that also uses views to provide for efficient search, synchronization, and replica coherence.

Our design is guided by a *contextual analysis* of home data usage, which we conducted to explore the management tasks and usage patterns found in the home. Lab studies, based on use of Insight, indicate that views do indeed simplify replica management tasks for home storage, and confirm that users struggle when the management abstraction differs from the navigation mechanism, and are confused by distinctions between cached and permanent file copies. However, our results also show that with access to the correct abstractions, users achieve accuracy rates of up to 87% on sample data management tasks. Perspective has also been deployed in a five-person household for eight months, at a small scale including a custom DVR and two laptops, to the delight of its enthusiastic users.

This paper makes five primary contributions. First, it identifies the mismatch between data navigation abstractions (attribute-based) and management abstractions (volume-based) for home storage. Second, it presents the view as a management abstraction that removes this mismatch. Third, it provides results from a contextual analysis regarding home storage and replica management tasks. Fourth, it describes Perspective, a system that supports views and their use for replica management tasks. Fifth, it demonstrates the usability of the view abstraction, relative to traditional approaches, for replica management tasks.

# 2 Background: In-situ user study

The first step of this work was a thorough *contextual analysis*: a set of interviews conducted in the context of (that is, at the site of) the environment under study. Contextual analyses are common in HCI research, and provide a wealth of in-situ data, perspectives, and (sometimes most importantly) real-world anecdotes of the use of technology.

## 2.1 Contextual analysis methodology

We worked with eight households for our study, covering a total of 24 users (not counting small children), 81 devices, and over 12 hours of video. We recruited households with three or more members around the Pittsburgh, PA, and Washington, DC, areas who used a variety of digital information, such as digitally recorded TV and digital music, but were not computer professionals. To recruit participants, we used a combination of direct email, personal contacts, and fliers posted in public locations.

For each household, we conducted an hour-long interview with the members as a group in their home with all of their digital storage devices present. For the first half hour, we asked open-ended questions about data backup, device failure, and data sharing. For the second half hour, we asked each participant to bring forward each of their digital storage devices individually, tell us the device's capacity and utilization, sort through the data on the device, and then break down the data stored on that device into classes based on reliability preference. For each class of data, we asked the participants to tell us the size of the class and the relative importance of the data not being lost.

We believe that a study of eight households, 24 users and 81 devices provides a significant cross section from which we can draw useful insights. Studies of similar size are common in the HCI literature [2, 13]. From our contextual analysis we identified five key replica management tasks.

## 2.2 Data organization

In our interviews, we found a fundamental disconnect between the way that users accessed data normally, and the way in which they had to manage that same data. In normal data access most users utilized semantic abstractions through applications like music players, frequently accessed files lists, and desktop search. Most users also used email and online photo sites, and expressed little difficulty in navigating these primarily attribute-based systems.

In contrast, when managing this same data users were required to use the file hierarchy. We found that many users were completely unaware of the directory hierarchy that existed on their computers, since it was almost always obscured by applications. Of 24 users we interviewed, we found that at least 7 of them did not know how to use the file browser. Many users told us "I don't know where my music is stored, I just access it through iTunes."

Even experienced users had trouble with the disconnect between application data access and the filesystem hierarchy. Kyle described that the most frequent use of his backup drive was actually to find files that were lost on his computer hard drive. He explained:

*Kyle: "I understand my backup drive because everything goes exactly where I put it. But, on my hard drive, every piece of software has someplace it downloads stuff, and towards the end of your computer's lifetime, you have so much junk installed, you don't know where you set it to download to. So, you go back to an application and you download something, and you don't know where it goes."*

Marcy, a less experienced user, explained that she couldn't seem to find files once she had saved them, because she often wasn't sure where they went.

*Marcy: "It reminds me of those drop down files, it goes C Drive and then My Desktop, or My Computer, and then you save the file there, and then you can't find it again. And sometimes you can recover it by search, but I've definitely lost files on the computer."*

## 2.3 Mobility

Home users carry a plethora of devices with them outside the home. All of the users we interviewed had cell phones, and most had music players that they carried outside of the home. A majority (13 out of 24) of the users we interviewed also had computers that they frequently used outside of the home. It is critical that users be able to access their data on these portable devices.

While network access is frequently available, firewalls, corporate security policies and other barriers often make it difficult to connect to home devices from remote locations. For example, Kyle had to email data to himself from work and home, because his work firewall kept him from even checking his home mail, let alone connecting over the WAN. We believe that these observations argue for a decentralized storage solution that supports disconnection between devices, to allow users to carry data stored on devices from place to place, without requiring network access.

A number of users indicated that the data they stored would not fit in its entirety on their portable devices. Vania expressed his frustration about his photos:

*Vania: "At some point we just know this partition is photos and we back the whole thing up [and don't try to divide it], but it virtually becomes impossible to load it up on my laptop, because it doesn't have enough space."*

We observed that only 2 of 24 users had devices on which they regularly placed data in anticipation of needs in the near future. Instead, most users decided on a type of data that belonged on device (i.e. all my music, files for this semester), and only occasionally revisited these decisions when prompted by environmental changes. More frequent update operations usually involved copying all new files matching the device data criteria onto each device.

## 2.4 Reliability

*Jill: "I've learned not to keep anything that I care about somewhere that could be destroyed [on a hard disk]. You know, it's like you get your heart broken so many times, you don't go there again. It's kind of like that."*

With the increase in digital photography and video and the crossover between home and work environments, it is common that users have data stored on home devices they are unwilling to lose. However, we found that users were loathe to spend money on more capacity than was needed, and often explicitly chose to complicate their backup strategy to save money. During our interviews, 6 of our 24 interviewees explicitly mentioned cost as a reason they adopted more cumbersome data management strategies. Aaron summarized this point well:

*Aaron: "I can't be as organized as I'd like because I have no central place to back up stuff. ... [but] if it's a choice between shoes [for the kids] and a nice external hard drive, then it will be the shoes."*

Even when resources were not as tight, users did not want to spend money on extra capacity if not needed. Sally explained:

*Sally: "I don't have the money [for an external hard drive]. That's money I'd rather use to go to Spain."*

These concerns about cost suggest that uniformly backing up all of the data in a home will not be acceptable to many users. Cost will increase as a concern as more media (TV shows, movies, etc.) become prevalent in digital form. We found that media-heavy users especially needed to differentiate between data types. Even Kyle, who was very concerned with reliability and backed up the family data regularly on DVDs stored in a safe deposit box at the bank, was unwilling to back up all their data and had a separate policy for raw video footage and finished items to save space and time.

We found that most users differentiated between different types of backup based on attributes that ran counter to the attributes used in their normal data access methods. By forcing them to use a single set of attributes, the hierarchies made both tasks more complex. Aaron expressed his frustration at having to dilute his carefully chosen data access schemes with placement decisions made on different attributes, and pointed to this mismatch as a significant source of disorganization:

*Aaron: "I'm very conscious about the way I name things; I have a coding system. But the thing is, that doesn't work if you have everything spread out. The coding system makes sense when there's a lot of other things around, but not when it's by itself."*

It also did not seem to be sufficient to use a one-size-fits-all approach, as different users had very different opinions about the importance of their data. For example, while most users cared little about losing recorded TV shows, one household listed this data as some of the most important in the house; more important than any of their documents, and more important to one user than her photos. They were avid TV watchers, and during the regular TV season this loss would mean missing a show and having no way to catch up. When a year earlier their DVR had cut off the end of "American Idol" before the final decision, the experience was traumatic enough to warrant a heartfelt mention during the interview. Another user in another household said that the data she missed most from a previous failure was the music she had purchased. This is in contrast to other users, who cared little for music or other media, but were vitally concerned about photos.

Cost concerns notwithstanding, over half of users (14 of 24) did use some sort of backup to protect their data. However, many users expressed a distaste for spending time on these tasks and had sporadic backup strategies. Despite the distaste for spending time on managing reliability, very few used automated backup tools (2 of 14 users), which many participants described as being too complex for their needs and difficult to understand. Instead, most users who backed up their data (12 of 14) copied data by hand from device to device. Ramona summarized the sentiment we heard from several users:

*Ramona: "I don't know that I necessarily trust those programs that back up your information because*

*I feel like I know what I want, and I don't know what that's doing. I don't necessarily know that it's getting everything that I need, or that it's not duplicating the information that I already have."*

The fact that users crave visibility into what their tools are doing on their behalf is a particularly powerful insight which drove our design. We found that users want to understand what is being done on their behalf, and are very distrustful of tools that completely hide complexity. However, they wish to use semantic naming mechanisms when accessing their data, and do not wish to be forced into using a hierarchy for these tasks.

## 2.5 Capacity management

*Vania: "Yeah, we've been short on space on pretty much everything. We had like 150GB initially, we ran out of space, we bought 250GB, we ran out of space again. Then we bought a 500GB external drive recently, now 350GB are taken."*

We found that many users (11 of 24) had run out of space at some point. A number of these users (5 of 24) were heavy media users, either in video or audio. However, even average users spoke of running short on space, and having to remove data or buy new capacity.

We have already mentioned the importance of cost in determining the amount of capacity in the home. While disk capacity is cheap, Aaron echoed what we heard from many users:

*Aaron: "I imagine it isn't too expensive to buy a large drive these days, but you know, too expensive is relative."*

For those who managed capacity, most used time to differentiate their data, putting old files on secondary machines and recent files on active ones. Users also utilized distinctions by task or period in the user's life (e.g., files from college or a particular project) or type (e.g., raw movie files vs. processed videos) to manage capacity.

Users also complained of the inflexibility of standard volume-based filesystems:

*Vania: "I reformatted my computer a couple of times. The reason is mostly because you start, and you create this partition for non-program stuff, and you end up stuffing C with a lot of things, and then C runs out of space so that D doesn't run out of space and then your computer gets slow, and then you have to reformat."*

## 2.6 Device churn

*Aaron: "Stranded is a good word for [the data left on my old computers]. An electronic desert island out there. An e-Island."*

Our interviews paint a very dynamic picture of the devices in the home. Users' devices were constantly changing due to upgrades, changes in employment, failure, and living conditions. One of our interviewees, Marcy, had owned 13 different cell phones in the last few years.

For example, 4 of our 24 users were currently or had recently been without a computer for an extended period of time during which they stored data on friend or family machines, or kept it on CDs or DVDs until getting a new machine. Another two users were in the process of pulling personal data from work machines due to a change in employment. Moving data forward from an old machine was often a difficult task:

*Vania: "You format a computer, you have everything fresh and clean and its easy, and then you start trying to get data back in and it's a mess."*

The result was a trail of data strewn across old and new devices. Four users explicitly mentioned data stranded on old devices because of the difficulty moving it forward. One user even described calling home to ask her mother to try to access files on an old desktop machine and read the needed information back to her.

### 2.7 Discussion

The contextual analysis provided a wealth of knowledge that has informed our system design, implementation and evaluation. First and foremost, the study directed us to study the five aspects of home storage management described above: organization, mobility, reliability, capacity, and device churn. In the following sections, we show how Perspective allows users to address these tasks. In addition, our contextual analysis has brought into focus key themes that informed our design of Perspective.

**Naming mismatch**: Today's systems present home users with a mismatch between the way applications present and access data and the way that data appears in the underlying filesystem. We found that users are generally comfortable with the way applications handle data, which often uses attributes and tags, but are unable map from this semantic view of their data into an of the underlying filesystem hierarchy. This dichotomy suggests that a unified attribute-based scheme can serve both high-level applications and low-level management tasks.

**Lack of observability**: Users have little insight into what automated management tools like backup programs are doing on their behalf and, therefore, do not trust them. We believe that a successful management system will employ automated tools, but that any policies imposed by those tools must be understandable by users.

**Many attributes, one hierarchy**: There are many axes along which users can and do subdivide their data, e.g., time, owner, type, etc., and the users in our study clearly use several of these dimensions in their data organization, and are surprisingly comfortable with their use. The real challenges seemed to appear when they were required to force groupings made for several different purposes into the same hierarchy.

**Lack of device centralization**: The users in our study employed a wide variety of computers and devices, and while it was not uncommon for them to have a set of primary devices at any given point in time, these devices changed rapidly, the boundaries between them were very porous, and users resented and delayed work to consolidate data or functionality. For these reasons, while centralization is very appealing from a systems design standpoint, and has been a key feature in many distributed filesystems, it seems to be a non-starter with home users. Therefore, we chose a decentralized, peer-to-peer architecture for Perspective, which more cleanly matches the realities we encountered in our contextual analysis.

## 3 Views

To address the data management challenges we observed in our studies, we propose the use of semantic management using *views*, a term that we borrow from database terminology. In Perspective, a view is a concise description of the data stored on a given device. Each view describes a particular set of data, defined by a semantic query, and a device on which the data is stored. For example, a sample view might be *"all files where artist='Aerosmith' " stored on the Desktop* , or *"all files where owner='Brandon' and type='Contact'" stored on Brandon's cell phone*, or *"all files where time created < Dec 2, 2007" stored on external hard drive*. A view-based replica management system guarantees that any object that matches the view query will eventually be stored on the device named in the view.

Figure 1 shows the combination of management tools and storage infrastructure that we envision, with views serving as the connection between the two layers. Users can set policies through management tools like the Insight tools described in Section 4, file browsers, or applications from any device in the system at any time. Tools implement these changes by manipulating views and the underlying infrastructure, Perspective, in turn enforces those policies by keeping files in sync among the devices according to the views.

Views provide a clear division point between tools that allow users to manage data replicas, and the underlying filesystem that implements the policies. Views are semantic and, thus, are defined in a human-understandable fashion, addressing the critical issue of observability. Views are also flexible enough to provide the semantic data access users need, allowing them to serve both access and management tasks

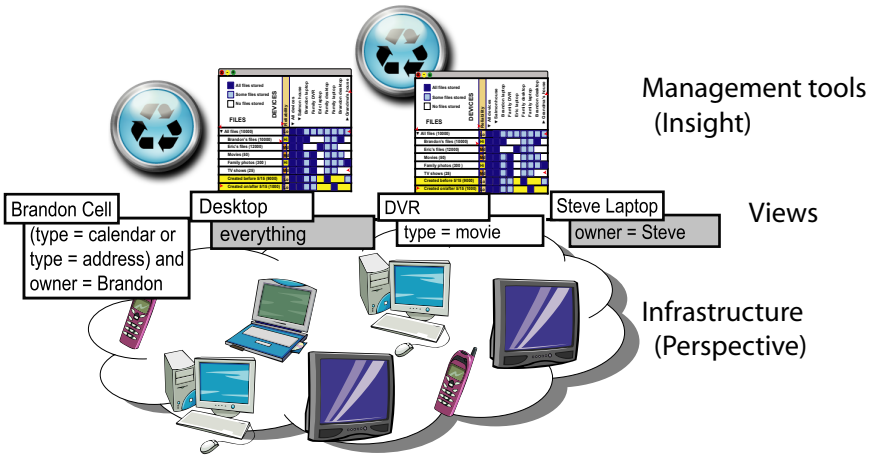


Figure 1: **View-based architecture.** Views are the interface between management tools and the underlying heterogeneous, disconnected infrastructure. By manipulating the views, tools can specify data policies that are then enforced by the filesystem.

with the same primitives. This allows users to understand and reason about the policies that exist in the system, regardless of whether those policies were set by the users themselves or by management tools on their behalf.

### 3.1 Query language

We use a query language based on a subset of the XPath language used for querying XML. Our language includes logic for comparing attributes to literal values with equality, standard math operators, and an operator to determine if a document contains a given attribute. Clauses can be combined with the logical operators *and*, *or*, and *not*. Each attribute is allowed to have a single value, but multi-value attributes can be expressed in terms of single value attributes if desired. We require all comparisons to be between attributes and constant values.

In addition to standard queries, we support support two extra operations needed for efficient faceted metadata and reliability analysis. The first is the *enumerate values* query, which returns all the values of an attribute found in files matching a given query. This operation must be efficient; luckily we can support it at the database level using indices, which negate the need for full enumeration of the files matching the query. The second is the *enumerate attributes* query, which returns all the unique attributes found in files matching a given query. While this operator also must be reasonably efficient, it is not used nearly as frequently at the enumerate values query, and so can be less efficient.

This language is expressive enough to capture many of the data organization schemes currently in use (directories, unions [17], faceted metadata [29], keyword search, etc.) but is still simple enough to reason about the overlap of queries, an operation that must be efficient for Perspective to be successful. The overlap comparison operator returns one of three values when applied to two views: one view *subsumes* the other, the two views have *no-overlap*, or the relationship between them is *unknown*.

### 3.2 Placing file replicas

In Perspective, the views control the distribution of data among the devices in the system. When a file is created or updated, Perspective checks the attributes of the file against the current list of views in the system

and sends an update message to each device which has a view that contains that file. Each device can then independently pull a copy of the update.

When a device, A, receives an update message from another device, B, it checks that the updated file does, indeed, match one or more views that A has registered. If the file does match, then A applies the update from B. If there is no match, which can occur if the attributes of a file are updated such that it is no longer covered by a view, then A ensures that there is no replica of the file stored locally. Most views are *complete views* which guarantee that the device will always store all matching replicas. However, in some cases advanced users or tools may use *partial views*, which provide the same update functionality but leave the receiving device free to choose not to apply an update.

Perspective uses views as a first-class construct, exploited in implementing many aspects of the distributed filesystem. However, the system usage of views is much like EnsemBlue's persistent queries [16], which can be adapted to support view-based management. In this case, to implement a view the filesystem would register a persistent query looking for updates on all files that match the view. Note that this must include both files that currently match the query and files that matched the query before the current update.

In Perspective, each device is represented by an file in the filesystem that describes the device and its characteristics. Views themselves are also represented by files. Both device and view files are replicated on all participating devices so that users can manage data even if not all devices are currently present. Each device registers a view for all device and view files in order to receive updates.

### 3.3 View-based data management

This section describes how views can be used to address the five challenges we outlined in Section 2.

**Data organization**: Perspective extends the use of semantic data specification beyond naming, with which users have become accustomed, by using it as the key organizing principle of all of its management functions. Doing so eliminates the disconnect between high-level applications and low-level data organization, addressing the naming mismatch described above. Management policies such as those described below for mobility, reliability, and capacity management, can cut across many different attributes of the file space, including time, user, data type, etc. Semantic naming for views allows home users to flexibly describe sets of files to the system to match their desired policies.

**Mobility**: File mobility among collections of devices can easily be controlled using the semantic groupings enabled by views. Users and tools can specify mobility preferences along many different attribute axes, and Perspective will keep files in sync. We believe that the synchronization model supported by views matches well with the habits of common users. In our studies, we observed that users generally had relatively stable, yet semantically-meaningful, sets of files stored on a given device, and usually only added new files to this set of data. Views support this pattern of management by allowing users fine granularity in their decisions.

**Reliability**: Complete views provide permanence guarantees, as well as a means to reason about and manipulate the number of replicas of files in the system. This allows users to reason about these settings at the granularity that is natural to them. For some users, these settings will be very simple (e.g., all files stored on two well-provisioned devices). However, our interviews suggested that a number of users were unwilling to pay for backup of all their data. For these users, the views primitive also supports more fine-grained control with the exact same primitive (e.g., all documents stored on three devices, TV shows stored on one, everything else stored on two). Views accommodate different priorities by allowing users to specify reliability semantically, and thus at the granularity and on the boundaries they desire.

The semantic basis of views allows users to make these distinctions without muddying the data organization they use to access data on a daily basis, another challenge presented by hierarchies in our interviews.

Views also help users understand management tools by allowing tools to specify flexible policies in a uniform way, which users can still easily audit and change. If this language is constricted, such as volumes,

the more flexible policies found in many management tasks must be made in an ad-hoc fashion at the application level, leading to a loss of user comprehensibility.

**Capacity management**: Similarly, views allow users and tools to specify capacity management policies flexibly without cluttering the naming used to access information. In general, higher-level management tools should alert the user and take action when a device starts to approach its capacity, thus working to keep devices from actually running out of space. For example, an automation tool might offload all the files created before a certain date from an overloaded desktop computer onto a device with extra capacity.

**Device churn**: Semantic management addresses the challenge of device change. Views make it easy for users to copy the data stored on an old device onto a new device. To move all the data from an old device to a new one, a user simply creates views on the new device which match the views from the old device, and Perspective will automatically find the appropriate data and place it on the new device. This works even if the old device is no longer functional, assuming replicas of that data exist elsewhere in the system. Unlike a more conventional caching approach, views provide precise and concise information about what the data was stored on the old device, even if it is no longer functional. This simplifies both upgrade and failure recovery.

# 4 View-based tools: Insight

To explore view-based management, we built a set of tools and user interfaces with which users can manipulate views. We call this sample set of tools Insight.

**Customizable faceted metadata**: One way of visualizing and accessing semantic data is through the use of *faceted metadata*. Faceted metadata allows a user to choose a first attribute to use to divide the data, and a value at which to divide. Then, the user can choose another attribute to divide on, and so-on. Faceted metadata helps users browse semantic information by giving them the flexibility to divide the data as needed. But, it can present the user with a dizzying array of choices in environments with large numbers of attributes.

To curb this problem, we developed *customizable faceted metadata* (CFM), which exposes a small user-selected set of attributes as directories plus one additional *other categories* directory that contains a full list of possible attributes. The user can customize which attributes are displayed in the original list by moving folders between the base directory and the *other categories* directory. These preferences are saved in a customization object in the filesystem. The file structure on the left side of the interface in Figure 2 illustrates CFM. Perspective exposes CFM through the VFS layer, so it can be accessed in the same way as a normal hierarchical filesystem.

**View manager interface**: The view manager interface (Figure 2), allows users to create and delete views on devices and to see the effects of these actions. This GUI is built in Java and makes calls into the view library of the underlying filesystem.

The GUI is built on Expandable Grids [18], a user interface concept which was initially developed to allow users to view and edit file system permissions. Each row in the grid represents a file or file group, and each column represents a device in the household. The color of a square represents whether the files in the row are stored on the device in the column. The files can be "all stored" on the device, "some stored" on the device, or "not stored" on the device. Each option is represented by a different color in the square. By clicking on a square a user can add or remove the given files from the given device. Similarly to file permissions, this allows users to understand and manipulate actual storage decisions, instead of lists of rules.

An extra column, labeled "Summary of failure protection," shows whether the given set of files is protected from one failure or not, which is true if there are at least two copies of each file in the set. By clicking on an unbacked-up square, the user can ask the system to assure that two copies of the files are stored in the system, which it will do by placing any extra needed replicas on devices with free space.

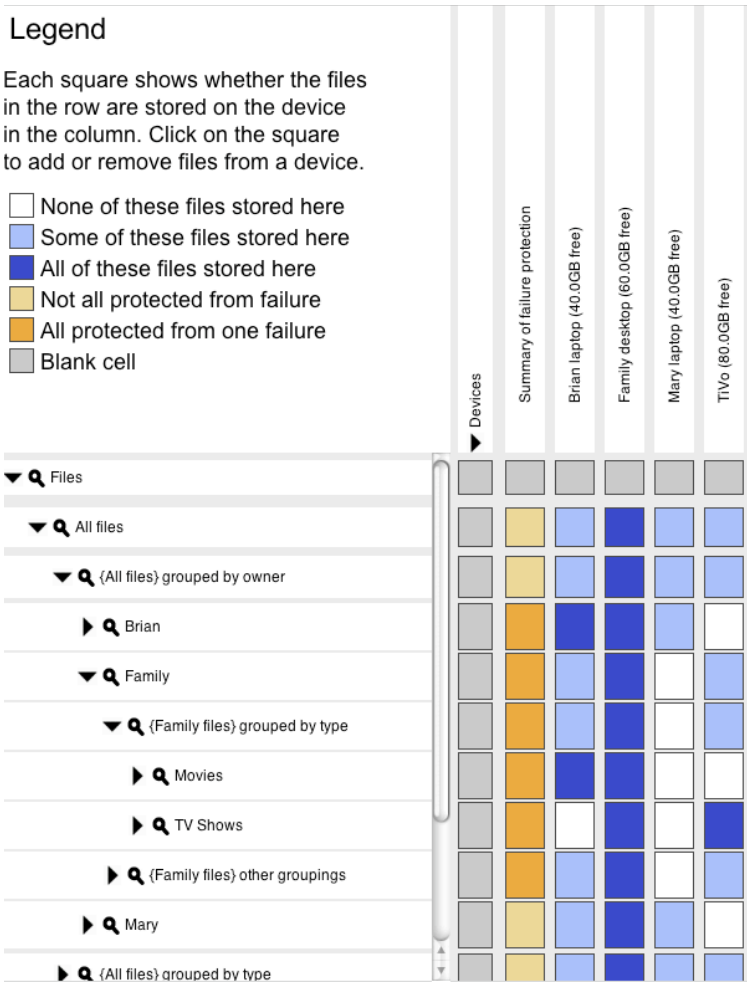


Figure 2: **View manager interface.** A screen shot of the view manager GUI. On the left are files, grouped using faceted metadata. Across the top are devices. Each square shows whether the files in the row are stored on the device in the column.

An extra row contains all unique semantic queries found in the views and where each of these file groups is stored, which can aid a user in determining what data is stored on a given device.

# 5 Implementation

In this section we describe how views can be effectively used by the filesystem itself to provide functions such as replica coherence, reliability reasoning, and search. Using views as a building block of the filesystem both eliminates the need to implement views as a separate set of mechanisms, and limits the possibility of the user understanding of the system and the system architecture from diverging. This section focuses on the high level design decisions in Perspective, and the changes to core filesystem functions to support the more flexible view-based location scheme.

## 5.1 View-based filesystem

The Perspective prototype is implemented in C++, and uses FUSE [5] to connect with the filesystem. It currently runs on both Linux and Macintosh OS X. The file store is implemented by storing data in the

machine's local filesystem and metadata in a SQLite database with an XML wrapper. The implementation is completely decentralized, and does not require any central point of control. Space constraints keep us from including many details in this section, refer to [19] for more detail.

The prototype system has been supporting one researcher's household's DVR, which is under heavy use; it is the exclusive television for him and his four roommates, and is also frequently used by sixteen other friends in the same complex. It has also been storing one researcher's personal data for about ten months. Our next steps include preparing the system to work in several non-technical households for a wider, long-term deployment over several months.

**Search**: All naming in Perspective is built on top of semantic metadata, therefore search is a very common operation, both for users and for the system itself. Views afford a mechanism for improving the locality of searches, as the system is able to easily avoid devices which cannot store files with particular attributes based on their published views. For example, it is simple to avoid searching for spreadsheets on a music player which only publishes a view for audio files. As well, Perspective can preferentially direct searches to well-provisioned devices with very inclusive views. Perspective also caches search results to improve performance.

**Replica coherence**: Because all devices store the device and view files for all other devices in the household, when a device makes a change to a file or creates a new file, it can use the views to decide which devices must hear about this update. The device forwards the new metadata for the file in an update message to all available devices. This mechanism automatically places new files, in addition to keeping current files up-to-date according to the current views in the system.

Devices which are not currently accessible will receive updates at synchronization time, when the two devices must exchange information about updates that they may have missed. We can still apply a modified update log to limit the exchanges to only the needed information, similar to that used in Bayou [26]. As in Bayou, the update log is only an optimization; we can always fall back on full synchronization.

While tools should help users address capacity problems before they arise, if a device does run out of space, it will refuse subsequent updates, and makes a mark in the device file noting the fact that the device is out of space. Until a user or tool corrects the problem, the device will continue to refuse updates, although other devices will be able to continue.

**Consistency**: Like most file systems which support some form of eventual consistency, Perspective uses version vectors to ensure that all file replicas eventually converge to the same version. We chose to support file-level consistency, like FICUS [10], rather than volume-level consistency like Bayou and other systems, because we do not expect the volume-level consistency to be needed for most home applications.

Any system which supports some form of eventual consistency must also handle *conflicts*, where two devices modify the same file without knowledge of the other device's modification. Views do not affect the choice of conflict resolution method. We choose to deterministically choose one version to advance, but put data for the other update in a related file so a tool can resolve the conflict later.

Like many other distributed filesystems, when a file is removed Perspective maintains a *tombstone* marker that notes the deletion of the file, and assures that all replicas of the file are deleted. Perspective uses a two-phase garbage collection mechanism like that used in FICUS between all devices with views that match the file to which the tombstone belongs.

**Security**: Security is not a focus in this paper, but is certainly a concern for users and system designers alike. While Perspective does not currently support it, we envision using mechanisms such as those promoted by the UIA project [3].

## 5.2 Reasoning about reliability

Semantic management gives users more flexibility in managing their data. However, we must make sure that this extra flexibility does not limit the ability of users and applications to reason about the reliability and

permanence of their data.

Relaxing the strict volumes model makes determining how many copies of a certain file exist in a system more challenging. We could enumerate all of the files in the system, but this is a costly operation, and only works if all data is currently accessible. Instead, we can compute the number of copies of a given query without having to enumerate all of the files in this query, by using an *overlap tree* which describes the different groups of data found in the system.

To create the overlap tree, we begin by creating a list of the attributes involved in views found in the household. We begin with a node containing a query for all data in the household as the root. We divide this set of data by the first attribute in our list, and use the enumerate values query described in Section 3.1 to efficiently find all the values of this attribute for the current node's query. We then create a child node from each value with a query of the form *<parent query> and attribute='value'*. We compare the query for each child node against the complete views on all devices. If the compare operator can give a precise answer (i.e., not *unknown*) for whether the query for this node is stored on each device in the home, we can stop dividing and return. Otherwise we recursively perform this operation on the child node, dividing it by the next attribute in our list.

Once we have computed the tree, we can use it to determine the number of full copies stored in the system of any given query, in addition to identifying where the query is stored. We do so by looking at all leaf nodes of the tree which overlap with the given query. In many cases, this can be done entirely through use of the in-memory compare operator, although we may occasionally need to perform more costly overlap detection by searching through files. In our test cases, we could compute the overlap tree in a few seconds, and compute overlaps quickly enough as users navigated their files.

Perspective maintains permanence by guaranteeing that files will never be lost by changes to views, regardless of the order, timing, or location of the change. We also provide a guarantee that once a version of a file is stored on the devices associated with all overlapping views, it will always be stored in all overlapping views. This provides users and applications a strong assurance on the number of copies in the system based on the current views. These guarantees are assured by careful adherence to three simple rules: 1) When a file replica is modified by a device, it is marked as "modified". Devices cannot evict modified replicas. Once a modified replica has been pulled by device holding a complete view covering it, the file can be marked as unmodified, and then removed. 2) A newly created complete view cannot be considered complete until it has synced with all devices with overlapping views, or synced with one device with a view that subsumes the new view. 3) When a complete view is removed, all replicas in it are marked as modified. At this point, the replicas can be removed when they conform to rule 1.

These rules assure that devices will not evict modified replicas until they are safely on some "stable" location (i.e., in a complete view). The rules also assure that a device will not drop a "stable" (i.e., one that is stored in a complete view) replica of a file until it has confirmed that another up-to-date replica of the file exists somewhere in the system.

# 6 Evaluation

Because our focus is building system primitives to support home data management, our primary evaluation must be the usability of those architectural primitives. This is a challenge, since we know of no way to evaluate the usability of an architecture in isolation from an interface, and thus cannot conclusively say that one architecture is inherently "more usable" than another. Yet, these critical system architecture decisions must consider their sizeable impact on usability, especially in systems designed for use by non-technical users. The researchers and some housemates use the prototype for everyday use, and are very happy with it, but that fact offers little insight into the target user community's reaction.

To evaluate the impact of views on usability, we conducted a lab study where we had non-technical

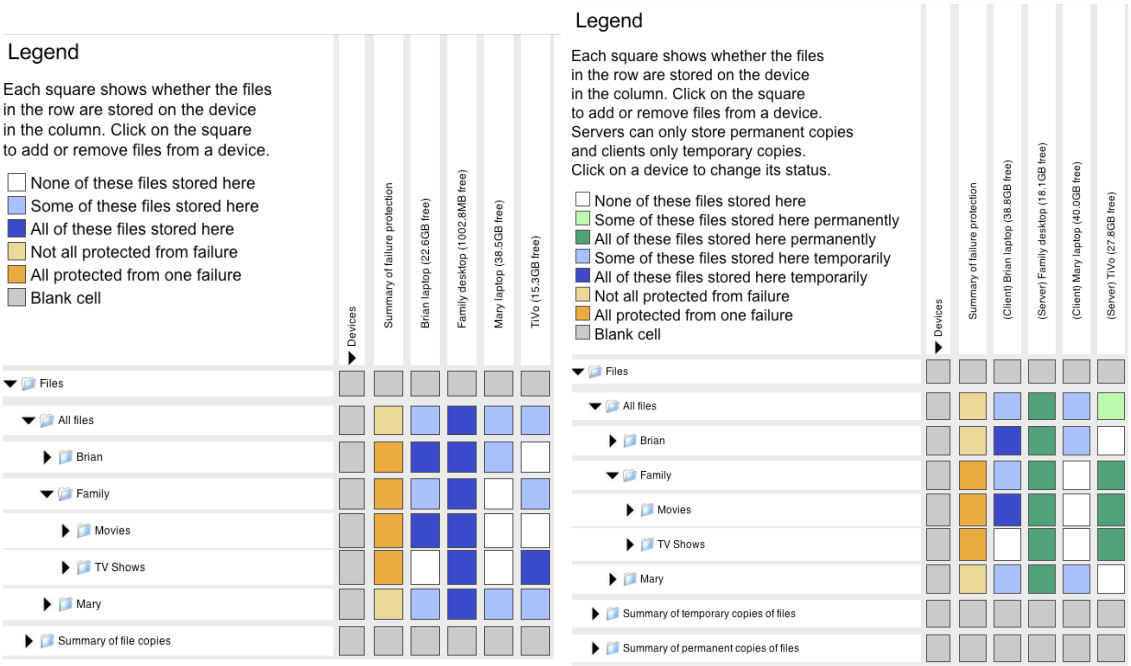


Figure 3: **Comparison interfaces.** On the left is a screenshot of the views-directory interface, on the right the volumes interface.

users use a set of interfaces to perform home data management tasks, and measured accuracy and completion time. We designed a user interface for each architectural primitive. But, to insulate our results as much as possible from the interface, we made the interfaces as similar as possible, while still being faithful to the underlying primitives. Because a well-defined user interface does not currently exist for these tasks, we chose to compare our management primitives using the Expandable Grids UI design.

We compared views as defined by customizable faceted metadata, against a more traditional volume-based approach. However, these two primitives differ in two fundamental ways: (1) the view-facet interface is semantic, and (2) it gives users freedom to place any subset of their files permanently on any device, without requiring preset volume boundaries. To tease out these two differences, we also evaluated a views-directory primitive, which uses hierarchical data organization, but still allows users full flexibility in placing data replicas.

**Views-facet interface**: The interface described in the Insight section. It uses CFM to describe data, and allows users to place any set of data described by the faceted metadata on any device in the home.

**Views-directory interface**: This user interface replaces the CFM organization with a traditional directory hierarchy, otherwise it is identical. In particular, it allows users to place any set of data (a subtree of the hierarchy) on any device. Figure 3 shows a screenshot of this interface.

**Volumes interface**: This user interface represents a similar interface built on top of a more classical volume-based system. It also uses a directory hierarchy to organize data, like the views-directory interface. However, the volumes interface has several additional restrictions and complexities required by the architecture. Each device is classified as a client or server, and this distinction is listed in the column along with the device name. The volumes abstraction only allows permanent copies of data to be placed on servers, and it restricts server placement policies on volume boundaries. We defined each root level directory (based on user) as a volume. The abstraction allows placement of a copy of any subtree of the data on any client device, but these replicas are only temporary caches and are not guaranteed to be permanent or complete. The interface distinguish between temporary and permanent replicas by color. The legend in the corner of

the interface displays a summary of the rules for servers and permanent data and for clients and temporary data. Figure 3 shows a screenshot of this interface.

## 6.1 Experiment design

Our user pool consisted of students from two nearby universities in non-technical majors who stated that they did not use their computers for programming. We did a between-group comparison, with each participant using one of the three interfaces described below. We tested 20 users in each group, for a total of 60 users overall. We also ran a 20-user pilot test before starting the study, to tune our interfaces and experimental design.

We created a filesystem with just under 1,000 files based on the filesystems we observed in our contextual analysis. We created a setup with two users, Mary and Brian, and a third "Family" user with some shared files. Our setup included four devices: two laptops, a desktop, and a DVR (described as a TiVo, as users were unfamiliar with the term DVR). We chose to create the file structures for the user, since having users create their own hierarchy would make it more difficult to simulate the history involved in a real filesystem. If anything, we expect this approach may be biased in the favor of hierarchical systems, as the structures we are using come from a study of hierarchical systems.

The users performed a *think-aloud* study in which they spoke out loud about their current thoughts and read out loud any text they read on the screen. This provided us insight into why users found particular tasks difficult and how they interpreted the information given to them by the interface. All tasks were performed in a *latin square* configuration, which guarantees that every task occurs in each position in the ordering, and each task is equally likely to follow any other task, in order to limit the impact of ordering effects.

### 6.1.1 Tasks

Each participant performed the same set of tasks, which we designed based on our contextual analysis. We also used the interviews to determine the attributes used to grouped data. The majority of users divided their data across four major categories: user/type (39%), task (26%), type alone (12%), and user alone (10%).

In total, our participants performed 13 data management tasks during the test, 9 of which were direct comparisons between interfaces. While space constraints do not allow us to include the full text for all of them, we include the text of several tasks here to illustrate their nature. The text for each task appeared on-screen during the test.

**TV (TV)**: "Brian and Mary's TiVo is connected to their TV, and they like to watch recorded TV shows using the TiVo because of the large screen. However, sometimes when Brian sits down to watch the TiVo, he can't see some of the recorded TV shows, which frustrates him. Make sure Brian can watch any TV show using the TiVo, even if the other devices in the house may be turned off or outside of the house."

**Mary's files (MF)**: "Sometimes computer devices will fail for various reasons. If any one device failed in the home, is Mary's data guaranteed to be safe? Don't worry about Brian's files or the Family files for this question."

**Batman Begins (BB)**: "Brian wants to watch the movie Batman Begins on his laptop on an upcoming trip. Make sure he can do so."

**Travelling Brian (TB)**: "Brian took his laptop on vacation with him today. What files will he be able to access while on vacation?"

## 6.2 Observations

We were surprised at how well novice users were able to perform these tasks, even with the more complex volumes interface. The overall accuracy rate for the views-facet interface, was 87%. The views-directory interface trailed slightly with 81%. The volumes interface with the lowest accuracy of 69%.

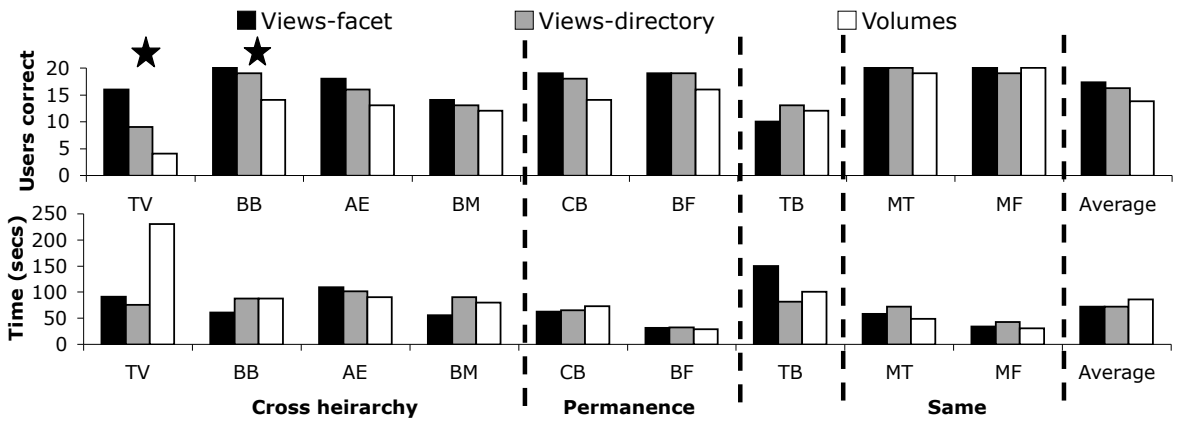


Figure 4: **Lab test results.** The top graph shows the accuracy results from our lab study; the graph shows how many of the 20 users completed each task correctly (higher is better). The lower graph shows similar results for time (lower is better). The two starred tasks contain statistically significant comparisons between the volume and views-facet interfaces on accuracy. The tasks are grouped into four categories, and the final entry is an average over all tasks.

The think-aloud nature of our study allowed us to observe a variety of trends in the way users utilized these interfaces. For example, users did have trouble getting accustomed to common UI conventions, such as expanding file groups, and working with popup menus; in the pilot test, before we added some initial nuts-and-bolts interface training, they quickly became lost. Initial confusion may also be due in part to the Expandable Grid interface, which provides a large amount of information at once. However, after several tasks, most users mentioned that they felt quite comfortable with the visualization.

We also found that users often assumed that a user's files must be on their laptops, despite evidence in the interface to the contrary. We started many of our tasks in what we expected would be a fairly intuitive state: all the files stored on the desktop and none anywhere else, but users seemed puzzled at why a user's data wouldn't be on their laptop or why TV shows wouldn't be on the DVR, and they would sometimes assume this must be the case. When first using the interface, a number of users tried to drag a user's laptop around as proxy for the data they owned, even though the interface stated there was no data stored there. This suggests that users may have difficulty initially separating data categories from where they are stored.

As suggested by our contextual analysis, users found hierarchies difficult to use. Users often failed to expand file groups, even after the training task walked them through this operation, and instead puzzled for long periods of time about how to reason about the folders. Even the faceted metadata case suffered some from this challenge, as the information is presented as a set of groups into which the user can drill down.

Finally, we found that users seemed quite comfortable with faceted organization after some experience. We thought that previous experience with hierarchies would make users likely to misunderstand a faceted organization.

### 6.3 Results

Figure 4 shows the task-by-task results from our lab test. The upper graph shows accuracy, defined as the number of users who did the task correctly, and the bottom graph shows the average time for those users who did complete the task correctly.

**Statistical analysis**: We performed a statistical analysis on our accuracy results in order to test the

strength of our findings. Because our data was not well-fitted to the chi-squared test, we used a one-sided Fisher's Exact Test. We used Benjamini-Hochberg correction to adjust our p values to correct for our use of multiple comparisons. As is conventional in HCI studies, we used $\alpha = .05$.

We compared the accuracy of the directory interface to the volume interface, the facet interface to the volume interface, and the facet interface to the directory interface, on the tests in which we expected a difference. Two of our comparisons had p values below $\alpha$, and thus were statistically significant. The facet interface was statistically better than the volume interface for two of the four cross hierarchy tasks. The remainder of our results, while suggestive of trends, are not statistically conclusive with the size of our study.

**Cross-hierarchy tasks**: The TV (TV), Aerosmith (AE), Brian's Music (BM) and Batman Begins (BB) tasks all asked users to perform operations on file groups that were not consistent with the boundaries of the file hierarchy. They were asked to put all the music or TV shows in the household on a particular device, or to find a file by type, without specifying a user. For all of these tasks, more users correctly completed the problem using the facet interface, showing, as expected, the advantage of allowing policy specification on multiple attributes. For two of these tasks, the more complex tasks, the facet interface was statistically significantly better than the volume interface: the TV (TV) task (p=.001) and the Batman Begins (BB) task (p=.03).

The BM and AE tasks were very simple cross-hierarchy tasks; in each case the user simply had to understand to look in both users' folders. However, the TV task was slightly trickier, making the user look both in Family and Mary directories for television shows. The BB task also asked users to find a file, but did so when data was split between two devices based on time, creating a more complex hierarchy. Based on our contextual analysis, we expect real-world system problems to be even more complex, as the number of overlapping directories grows and the overlaps become increasingly ill-defined.

While this test is a bit skewed by the fact that users did not create these file trees,we found in our contextual analysis that people often had little knowledge of where things were stored in their households either, as many of the decisions had been made for them by applications or in the past for long-forgotten reasons. These tasks are also significantly simpler than real-world tasks because the file tree is small and nicely organized.

**Permanence tasks**: The Concerned Brian (CB) and Brian's Files (BF) tasks asked users to manipulate or analyze reliability. They included the possibility of confusing temporary and permanent replicas in the volume interface case. This is reflected in the relatively poor accuracy of the volume interface, but equivalent performance between the view-directory and view-facet interfaces.

**Enumeration tasks**: The Traveling Brian (TB) task is the only task where the view-directory and volume interfaces performed better than the view-facet interface. Due to a glitch in the interface, the majority of users performed this task by exhaustively enumerating the file tree. The extra cost and time in the view-facet interface appears to be due to the fact that the extra flexibility makes it more difficult for users to exhaustively enumerate the tree

We do not expect users to enumerate the entire file tree to accomplish the task, especially with larger trees. Instead, the interface should provide a list of all files stored on the device to the user. While our interface provided this information via the summary row, which summarized where data was stored in the home, in all three interfaces most users either missed the feature, or did not understand it. While we believe fixes to the interface can eliminate this problem, it is an area for further study.

**Same tasks**: The Mary's Travels (MT) and Mary's files (MF) tasks did not test any of the differences between the interfaces, and thus show no difference.

**Dividing data**: We also tested how spreading data across multiple devices affected users' ability to reason about data reliability. This capability is only supported by the view-based interfaces, so our comparison only includes the view-facet and view-directory interfaces. Each experiment shown in Figure 5 asked the user to determine if a given set of data described by the standard hierarchy is protected from a given failure.

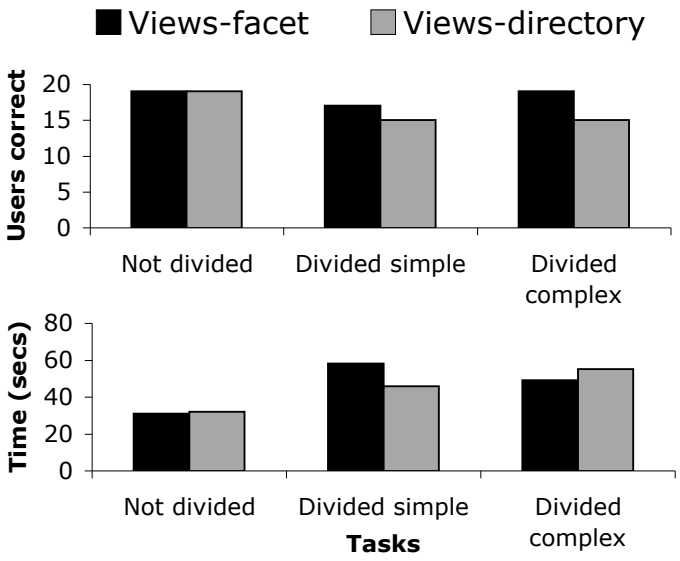


Figure 5: **Dividing data.** This graph shows the accuracy and time results for a set of tasks that asked reliability questions about data divided in different ways.

For the "not divided" task, we placed all of the data on two devices. In the two division cases, we divided one of the copies across two devices. In the divided simple task we divided on an attribute contained in the hierarchy. In the divided complex task on an attribute not contained in the hierarchy. The view-facet interface outperforms the view-directory interface in both cases, although none of the comparisons are statistically significant.

# 7 Related work

**Data organization**: One approach to storing data in the home is to put all of the data on a single server and make all other devices in the home clients of this server. Variations of this approach centralize control, while allowing data to be cached on devices [11, 23]

To provide better reliability, AFS [22] expanded this concept, allowing multiple copies of volumes to be stored on a tier of servers, each connected in a peer-to-peer fashion. However, volumes do not allow clients to access data when they are out of contact with the servers. Coda [20] addresses this problem by allowing devices to enter a disconnected mode, where they use locally cached data defined by user hoarding priorities. However, this mechanism cannot provide the reliability guarantees allowed by volumes because it makes no guarantee on what data resides on what devices. Views extend this notion by allowing volume-style reliability guarantees along with the flexibility of hoarding in the same abstraction.

A few filesystems suggested more flexible methods of organizing data. Footloose [15] proposed allowing individual devices to register for data types in this kind of system, but did expand it to general publish/subscribe-style queries, or explore how to use this primitive. BlueFS extended the hoarding primitive to work on a storage device basis, rather than a client basis [12].

**Consistency**: Peer-to-peer systems such as Bayou [26] and FICUS [10] extended synchronization and consistency algorithms so they could work even when devices are mobile, allowing these systems to blur or eliminate the distinction between server and client. Other systems, such as Ensemblue [16], provided support for groups of client devices to form device ensembles [21], which can share data separately from a server through the creation of a temporary pseudo-server. The Perspective filesystem borrows the consistency

and synchronization mechanisms from Bayou and FICUS, and extends them to handle the more flexible organization of views.

**Search**: We believe that effective home data management will use search on data attributes to allow flexible access to data across heterogeneous devices. The Semantic Filesystem [7] proposed the use of attribute queries to access data in a file system, and subsequent systems showed how these techniques could be extended to include personalization [8]. Flamenco [29] uses "faceted metadata", a scheme much like the semantic filesystem's. Many newer systems borrow from the Semantic Filesystem by adding semantic information to filesystems with traditional hierarchical naming. Microsoft's proposed WinFS filesystem also incorporated semantic naming [28]. Perspective uses views to provide an efficient implementation of a decentralized semantic store, on top of local semantic access, but also uses semantic naming and organization to perform management tasks.

**Publish/subscribe**: Our system architecture based on views fits the publish/subscribe model. Most publish/subscribe systems focus on providing search in a wide-area and highly dynamic environment, whereas our architecture seeks to use them as a foundation for providing consistency and reliability in a smaller and less volatile environment.

The HomeViews [6] project uses a similar primitive to allow users to share read-only data with one another over the wide area network. They combine capabilities with persistent queries to provide access control, and provide search over data, but do not use this abstraction to target replica management tasks.

**User studies**: While we believe our contextual analysis is the first focused on home data organization and reliability, researchers have conducted a wealth of studies on technology use and management, especially in the home [13, 25, 9, 1, 2, 4]. We borrow our methods from these previous studies.

# 8 Conclusion

Home users struggle with replica management tasks that are normally handled by professional administrators in other environments. Perspective provides distributed storage for the home with a new approach to data location management: the view. Views simplify replica management tasks for home storage users, allowing them to use the same attribute-based naming style for such tasks as for their regular data navigation.

# References


[1] R. Aipperspach, et al. A Quantitative Method for Revealing and Comparing Places in the Home. UBICOMP, 2006.

[2] A. J. B. Brush and K. M. Inkpen. Yours, Mine and Ours? Sharing and Use of Technology in Domestic Environments. UBICOMP 07, 2007.

[3] B. Ford, et al. Persistent personal names for globally connected mobile devices. OSDI. USENIX Association, 2006.

[4] D. M. Frohlich, et al. Breaking up is hard to do: family perspectives on the future of the home PC. *International Journal of Human-Computer Studies*, **54**(5):701–724, May. 2001.

[5] Filesystem in User Space. http://fuse.sourceforge.net/.

[6] R. Geambasu, et al. HomeViews: Peer-to-Peer Middleware for Personal Data Sharing Applications. SIGMOD., 2007.

[7] D. K. Gifford, et al. Semantic file systems. SOSP. Published as *Operating Systems Review*, **25**(5):16–25, 13–16 Oct. 1991.

[8] B. Gopal and U. Manber. Integrating Content-based Access Mechanisms with Heirarchical File Systems. OSDI., 1999.

[9] R. E. Grinter, et al. The work to make a home network work. European Conference on Computer Supported Cooperative Work (ESCW), 2005.

[10] R. G. Guy. *Ficus: A Very Large Scale Reliable Distributed File System*. PhD thesis, published as Ph.D. Thesis CSD-910018. University of California, Los Angeles, 1991.

[11] A. Karypidis and S. Lalis. OmniStore: A system for ubiquitous personal storage management. IEEE International Conference on Pervasive Computing and Communications. IEEE, 2006.

[12] E. B. Nightingale and J. Flinn. Energy-efficiency and storage flexibility in the Blue file system. OSDI. USENIX Association, 2004.

[13] J. O'Brien, et al. At home with the technology: an ethnographic study of a set-top-box trial. CHI, 1999.

[14] J. K. Ousterhout, et al. The Sprite network operating system. *IEEE Computer*, **21**(2):23–36, Feb. 1988.

[15] J. M. Paluska, et al. Footloose: A Case for Physical Eventual Consistency and Selective Conflict Resolution. IEEE Workshop on Mobile Computing Systems and Applications, 2003.

[16] D. Peek and J. Flinn. EnsemBlue: Integrating distributed storage and consumer electronics. OSDI, 2006.

[17] R. Pike, et al. Plan 9 from Bell Labs. United Kingdom UNIX systems User Group. United Kingdom UNIX systems User Group, Buntingford, Herts, 1990.

[18] R. W. Reeder, et al. Expandable grids for visualizing and authoring computer security policies. COMPCONFall., 2007.

[19] B. Salmon, et al. *Putting home storage management into Perspective*. Technical Report CMU-PDL-06-110. Sep. 2006.

[20] M. Satyanarayanan. The evolution of Coda. *ACM Transactions on Computer Systems*, **20**(2):85–124. ACM Press, May. 2002.

[21] B. Schilit and U. Sengupta. Device Ensembles. *IEEE Computer*, **37**(12):56–64. IEEE, Dec. 2004.

[22] B. Sidebotham. VOLUMES – the Andrew file system data structuring primitive. EUUGAutumn. EUUG Secretariat, Owles Hall, Buntingford, Herts SG9 9PL, Sep. 1986.

[23] S. Sobti, et al. Segank: a distributed mobile storage system. FAST. USENIX Association, 2004.

[24] C. A. N. Soules and G. R. Ganger. Connections: Using Context to Enhance File Search. SOSP. ACM, 2005.

[25] A. S. Taylor, et al. Homes that make us smart. *Personal and Ubiquitous Computing*. Springer London, 2006.

[26] D. B. Terry, et al. Managing update conflicts in Bayou, a weakly connected replicated storage system. SOSP. Published as *Operating Systems Review*, **29**(5), 1995.

[27] M. Weiser. The computer for the 21st century. *Scientific American*, Sep. 1991.

[28] WinFS 101: Introducing the New Windows File System, March 2007. http://msdn.microsoft.com/en-us/library/aa480687.aspx.

[29] P. Yee, et al. Faceted Metadata for Image Search and Browsing. CHI 2003, 2003.