Abstract

Modern storage systems have evolved from a simple model in which names denoted the information necessary to know both *where* a file was stored as well as *what* that file represented. Today, users routinely store information in a variety of different distinct storage silos, each of which has its own naming scheme, characteristics, and meta-data support. Storage silos can consist of storage local to the computer, shared with other local computers, stored in geo-replicated distributed storage systems, focused more on object storage than traditional hierarchical file organization, and communicated through non-traditional storage systems as part of collaborative efforts. There is little to assist those using the myriad of separate storage silos keep track of the relationships between them.

What this means is a user trying to locate a document she saved while collaborating with a colleague will need to search through each of these storage silos that she routinely uses: was the document attached as an e-mail, sent via a collaboration service, stored in a shared cloud storage location will have a difficult time locating that document six months later.

Revision: September 10, 2021

Chapter 1

Motivation and Problem Statement

For every particular thing to have a name is impossible. — First, it is beyond the power of human capacity to frame and retain distinct ideas of all the particular things we meet with: every bird and beast men saw; every tree and plant that affected the senses, could not find a place in the most capacious understanding. — **John Locke**, *An Essay Concerning Human Understanding* [1]

Naming is an essential human task. We use names to describe identity, relationship, and property. Thus, naming is quite broad, as it can be quite concrete — *identity* being particularly concrete — to quite abstract concepts such as *similarity*.

Computer storage primarily focuses on *identity*, with some limited flexibility to encode relationships and properties. It has been "good enough" to keep us suffering, but not good enough to have enjoyed decades of criticisms and attempts to find a better model.

One early model, Memex, described how computers can help human users by serving as a form of "augmented memory" describes a system in which properties and relationships are used to make humans more productive [2]. Despite stunning progress in many aspects of computer storage systems in the past 70 years, those storage systems have failed to improve on naming and organization in any meaningful way that permits realizing this plausible vision of how information should be organized.

1.1 Thesis Statement

To address the demands to organize, locate, and manage the rapidly growing body of data collected, stored, and organized by modern computer systems we must rethink the existing naming model of current computer storage systems and implement a richer, dynamic, and functional naming scheme capable of satisfying significantly more of those needs.

1.2 Why this is a Computer Systems problem

Paradigm paralysis refers to the refusal or inability to think or see outside or beyond the current framework or way of thinking or seeing or perceiving things. Paradigm paralysis is often used to indicate a general lack of cognitive flexibility and adaptability of thinking. — The Oxford Review Encyclopedia of Terms (2021).

One challenge in developing this work is the resistance of the computer systems community to conside~~ring~~ *naming* ~~as~~ a systems problem. One common response is to posit that this is an Human-Computer Interface (HCI) problem. While there are certainly aspects of data visualization that *are* HCI problems, a review of the current state of affairs suggests this is clearly untrue.

- The computer systems community has *already* taken over at least some aspect of naming; it is dereliction of responsibility to now insist the current state of affairs is not tightly tied to earlier choices by the computer systems community.
- The HCI community has been evaluating, reviewing, and proposing potential alternatives to the current computer storage naming paradigm for*decades* but these solutions fall short of being viable because it is not sufficient to change a single application — even something as core to the problem as the file browser — to resolve this problem.
- The computer storage community does admit to the challenges here and have implemented system-specific solutions to improving naming, but human users do not live in a reality in which such narrow solutions resolve the naming challenges of the larger system — it is not sufficient to argue that any storage system has solved this problem when users are called upon to use multiple storage systems on a daily basis.

The framework of modern file systems differs little from the framework used in Multics in 1965, itself based upon the familiar file cabinet metaphor described for electronic storage and data organization in the 1950s. The technological components that are used to construct file systems have changed dramatically in the ensuing decades, as storage devices have become smaller, higher density, more capable and less expensive. In 1965 the total amount of data storage was less than the capacity of a single Non-Volatile Memory Express (NVMe) storage device, yet the *framework* for how we organize, present, managed, and manipulate data has not changed. If anything, additional aspects of how we build file systems have ossified into common models: tight integration with the operating system, a presumption of block structured storage and memory page management.

While the HCI community is likely to contribute novel new ways of presenting information to users, they cannot do so until the systems community offers them the tools necessary to do so.

1.3 Challenges

TM ▶*I say quite a few things here, which I need to back up with suitable references.*◀

TM ▶*I have mulled over whether this belongs here or in the Background/Related work section.*◀

~~The computer systems community discourages challenging the current file system naming model: what we have works, it is not the purview of the systems community to address issues around utility, and our research is properly focused on supporting innovative new hardware solutions and optimizing their performance. *In spite* of this,~~ reality indicates that our storage models, which has already been evolving, is showing further signs of change. The space between file systems, which provide a basic abstraction of unmanaged data, and database systems, which provide a basic abstraction of managed data, is filled with a growing array of *semi-structured* mechanisms including: key-value stores, object stores, documents stores, no-SQL databases, data warehouses, and data lakes.

Each new mechanism that we invent for storing data creates a new “silo” of that information, often with its own semantics and meta-data, and seldom with any explicit way of tying related information together across such silos. Indeed, silos create the fundamental challenge for which I am proposing research to understand and evaluate potential solutions. Storage silos are often designed with specific usage patterns in mind: Intel DAOS is an HPC platform that is part of their "exascale" storage stack [1].

However, the commonly used mechanism for naming — the hierarchical namespace — confounds the storage *location* with the information *relationship*. In other words, ordinary applications familiar with using hierarchical file systems expect related objects, of whatever type, to be in or close to the same directory. The hierarchical file system model has been highly successful for more than a half-century, though that success is from the perspective of the storage community. The Human-Computer Interface (HCI) community has been pointing out that this model is deficient for many users.

Silos are not a new problem. The systems community has addressed them via "mount points". All mainstream consumer operating systems at present (Linux, MacOS X, and Windows) support mount points. NFS on UNIX is implemented using explicit mount points, much like media file system instances while AFS used a global namespace mechanism to connect its silos ("volumes") together so that users were given a single consistent namespace. More recent innovation has created internet-based namespaces, such as Amazon's S3, which is an object store. Some of those web namespaces can be mounted as file systems, such as using Web Distributed Authoring and Versioning (WebDAV), which converts the internet `GET/PUT` model into an hierarchical file systems namespace model. There are numerous "S3" FUSE file system implementations on github.com, not because they seek to be efficient but because that hierarchical model is the one understood by existing applications.

These solutions are inadequate: they still conflate *location* with *context*. Some of this is historical: the only reliable place where applications can store context is within the file name. Using file names to embed context ("meta-data") is well-understood [3]. Yet, embedding context in file names does not solve the problem of storing dissimilar types of data in the same location, which defeats the purpose of having specialized storage. Similarly, it does not solve the problem of placing data objects in their optimal storage location while preserving their relationship.

TM ▶*I'm not sure I like this example, but I'll leave it here for the time being.*◀ The need for this is increasingly clear. For example, Qumulo has focused on improving the manageability of large data collections: "When people are dealing with petabytes of storage and billions of files and they're using scaleable storage, they tend to run into problems not with storage itself but

[1] https://www.intel.com/content/www/us/en/high-performance-computing/daos-high-performance-storage-brief.html

the data"[2]. While Qumulo's approach is motivated by the same problem, their solution emphasizes scale and manageability within a new meta-silo, which does not address the core problem of organizing naming in a silo-independent fashion.

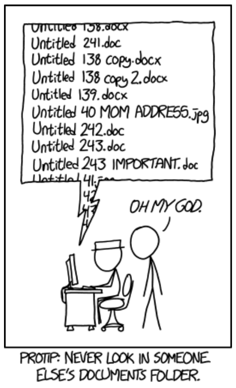

Figure 1.1: XKCD: Never Look in Someone Else's Documents Folder

Media file systems have long been organized using a simple internal key-value store. The BSD UNIX Fast File System used an "index node" (inode) as a description of a data object, such as a file or directory [4]. These nodes were indexed using an identifier, which acts as a simple key. The NTFS file system is structured similarly [5]. Recent work explored the idea of separating the namespace from storage with an emphasis on high performance solid-state storage (SSD) devices [6]. This echos earlier work suggesting using object storage devices (OSD) for file systems implementations [7].

Beyond separating naming from storage, there is also a trend to separate meta-data from storage as well. One early example of this is Lustre [8], with more recent work including Ceph and Gluster [9], [10].

The logic of separating meta-data from data solves one of the problems of storage, namely that the logical location of information can be distinguished from the physical location of data, which has been previously explored and deemed broadly beneficial. [11] This approach logically makes sense when considered in the multi-silo context as well, since a human user looking for something actually does not care *where* that the thing is located they care *what* the thing is.

This raises an intriguing question, one that is at the heart of my own research: *what is the purpose of naming?* Applications are quite happy to use

[2]https://www.seattletimes.com/business/storage-startup-qumulo-opens-the-kimono/

names that are meaningless to humans, such as content hashes, UUIDs, or randomly generated string names, such as are used for temporary application files. Indeed, if an application knows that a name is a fixed size, this can be used to simplify the implementation or to separate application named files from other files. An application that uses UUID names can limit its field of interest quite quickly by simply restricting itself to the 36 character format commonly used of 36 hexadecimal digits with four '-' separator characters breaking the UUID into five parts.

Humans use file names to provide *context* as to what is in the named object. Librarians assisting people in organizing data routinely suggest embedding contextual information in the file names; directories are then used to create additional organizational structure. There are numerous limitations to this approach, however, because humans do not organize things in the same fashion and even the same human does not organize things the same over time. This is such a common phenomenon even the popular press pokes fun at it (Figure 1.1.)

The inability to find specific things is *not* unique to computer data storage. Indeed the current situation for organizing digital data seems to be descended from the organizational structure of a library or a file cabinet, despite the fact the physical limitations on files, which in turn leads to a similar result: a large collection of files, spread over multiple different locations, each with its own organizational structure makes it difficult for the original creator of that content to find a specific file, let alone an outsider attempting to find relevant information.

Indeed, the system we have evolved works *in spite* of the obvious artificial limitations imposed by decades old decisions. Rather than being a deep insight that the current system works because it is the best of all possible models, it is a testament to human ingenuity and a tolerance for primitive, sub-standard systems.

There are a body of recommendations about file naming standards[3] [4] [5] [6]. Harvard Data Management suggests [7]:

- Think about your files

[3] Data Management for Researchers [12]

[4] Smithsonian: https://library.si.edu/sites/default/files/tutorial/pdf/filenamingorganizing20180227.pdf

[5] Stanford: https://library.stanford.edu/research/data-management-services/data-best-practices/best-practices-file-naming

[6] NIST: https://www.nist.gov/system/files/documents/pml/wmd/labmetrology/ElectronicFileOrganizationTips-2016-03.pdf

[7] https://datamanagement.hms.harvard.edu/collect/file-naming-conventions

- Identify metadata (e.g., date, sample, experiment)
- Abbreviate or encode metadata
- Use versioning
- Think about how you will search for your files
- Deliberately separate metadata elements
- Write down your naming conventions

Thus, there are common themes: a name provides context for *what* the given thing represents. Uniformity of information is also important — the "naming convention" permits not only identifying similarity but key elements of *difference* between any two named things. Thus, to return to the original question: "what is the purpose of naming?"

TM ▶*It seems that this Harvard list is a serious indictment of the existing system: it pushes the cognitive load onto the users, talks about meta-data , versioning, encoding,* and *capturing the naming convention.*◀

It seems clear that human naming is not the same as data storage naming, though we often treat them as the same. This can be seen in the data storage tendency to either use names that are entirely about locating a specific object, such as is the case with a key-value store, for example, or it combines location with identity, such as is typified by the Uniform Resource Identifier (URI) [13].

Thus, it would seem that naming is about:

Identity — the name of a thing should be sufficient to verify that it is the specific thing we seek. An example of this is biometric data for humans or a cryptographic hash for a storage object;

Location — the name of a thing could provide information that directly or indirectly specifies where it is located. A human example is how an identity card might specify a current residential address. A data storage example is the U.S. Library of Congress Classification System, which can be used to find a particular work within very large bodies of work even though books are typically not thought of as a storage media;

Relationships — the concept that one thing can be derived from another, whether in a direct form, such as a version of the same logical thing, or in a more indirect form, such as a provenance relationship showing that one thing was derived from another thing by applying some set of transformations to it;

Characteristics — this relates to something that is a property of the thing. For humans this might be the date of birth ("creation date"), height, weight, eye color, etc. For storage objects this might include timestamps, size, data type;

Context — at some some level, we often rely upon names to provide us with *context*. For humans, we assume that people with the same family name are likely to be members of the same family, even though "Aki Smith" is distinguished from "Dagon Smith".

When considered from this perspective, it seems clear why "naming is hard" — it plays multiple distinct roles, sometimes overlapping, sometimes interfering. These challenges are not well-served by existing naming support in computer storage systems.

TM ▶*I feel that I should capture the dynamic nature of naming, which really does differ from the traditional static model of it.* ◀

1.4 Contributions

[I don't let the cleaners in b]ecause ***I*** *know where everything in this room is. All the books, the papers — and the moment they start cleaning, those things get hopelessly organized and tucked away and I can never find them again.* — Crown of Midnight, Sarah J. Maas

This thesis proposal describes what I intend to contribute to our understanding of how to evolve naming in computer storage:

1. An explicit model for what computer storage naming is: how and why we name data objects, what a name *should be* in a computer storage context to correspond how names are used and formed; and

2. An architecture and design of a potential naming system that supports the naming model (item 1 that is sufficiently flexible to work across the full range of current and prospective computer storage needs: in-memory compute systems, small devices, computer workstations, enterprise scale storage systems, and distributed, geo-replicated cloud storage systems, which permits the construction of new purpose-built

storage systems with the strong naming support as well as integration of existing storage systems; and

3. Implement and evaluate a novel storage system implementation based on this design (item 2) that demonstrate strong naming support within a single storage silo model; and

4. Implement and evaluate an implementation of the richer naming system (item 2) on an existing computer storage system; and

5. Implement and evaluate the combination of both the novel storage system implementation (item 3) and the enhanced legacy system (item 4) into a unified system consistent with the model (item 2).

Chapter 2

Related Work

The optimal way to organize data within a storage silo has been extensively studied from multiple perspectives. That there are so many different approaches to evaluating the optimal way to organize things is a testament to both the importance and complexity involved in approaching this topic. This section will consider the following related work:

- The *storage* perspective, which is primarily rooted in the systems perspective. Storage in this context includes file systems and databases, each of which then has multiple different manifestations.

- The *provenance* perspective, which is related to systems but focuses on creating a context of explainability that is distinct from storage.

- The *human-computer interface* (HCI) perspective, which is related to how humans organize and find information within storage systems. There are a number of distinct perspectives to this work including: hieararchical data organization, personal information management, enhanced search, and cognitive data organization.

Each of these perspectives is complementary and assist in better understanding the problem: systems focuses on being able to efficiently and reliably store and recover information though often it is agnostic to the specifics of the information; provenance focuses on accountability and explainability; and HCI focuses on the human facing problems that need to be solved and tends to ignore how those solutions are implemented.

[3], [14]–[46]

2.1 Storage

Much of storage work is dominated by the realities of how media and networks function, with a goal towards increasing both capacity and performance. The basic model of data organization within storage systems tends to separate into *structured* data — typically what is found in databases — and *unstrutured* data — typically what is found in file systems or object stores.

TM ▶*Describe structured data: why do people use databases, what are the strengths and weaknesses. How does this relate to data organiztion?* ◀

TM ▶*Describe unstructured data: why do people use file systems or object stores, what are the strengths and weaknesses? How does this relate to data organization?* ◀

TM ▶*Describe the work that has been done in semantic file systems, metadata augmentation, and non-hierarchical organization structure (e.g., Ground and Placeless).* ◀

2.2 Provenance

TM ▶*Explain provenance. It is a more recent area of study, but it also captures more of the kinds of information that will be useful to me.* ◀

2.3 Human-Computer Interface

The traditional primary organizational model provided to users has been the file/folder metaphor TM ▶*Need citation to 1965 Daley and the 1956 storage papers*◀, which was itself derived from physical filing cabinets. However, electronic storage is not bound by the same restrictions as a physical filing cabinet, as can be seen even in early work TM ▶*Again, this is Daley*◀ where the model added *links*; the closest equivalent to this in filing cabinets would be to make duplicate copies of a document — I have seen exactly this routinely practiced by bookkeeping and accounting professional, whether using physical or electronic filing systems. This works because the objects are generally *immutable*, but for electronic storage systems without deduplication it is not particularly space efficient.

The Human-Computer Interface community has been studying human-centered data organization models for decades. For example, the HCI community observed that hierarchical file structure is challenging for users with low spatial abilities [47]. This suggests why storage developers would not even see there is a problem here: the study of computer science correlates well with the development of spatial abilities [48]. In my own discussions with even senior computer scientists it is the introduction of *silos* that seems to make the hierarchical abstraction break. “Did I store that in Dropbox, or Google Drive, or was it on my laptop or my desktop computer?”

The wealth of research here is astonishing, yet does not appear to influence the design of storage systems to exploit the results of that research. Indeed, the systems community seems to be focused on a *search based* solution while the HCI community research suggests that search is *not* preferred by users — instead, users want to *navigate*:

> *When retrieving a file the user needs to choose between folder based navigation and query based search. There are obvious intuitive advantages of search for both retrieval and organization. Search seems to be more flexible and efficient for retrieval. It is flexible because it does not depend on users remembering the correct storage location; instead, in their query users can specify any file attribute they happen to remember...*
>
> *In fact, regardless of search engine quality, people consistently use search only as a ‘last resort’ for that minority of cases where they cannot remember file locations...* [49]

Thus, from an HCI perspective it would seem that one potential research direction would be to consider potential interfaces that mimic navigation over a search interface.

For example, recent work around data curation explores the idea of a “data dashboard” since the first step of curation is finding specific data to curate [50]. This work builds upon prior research showing that when presenting data to users it is important to ignore storage silo boundaries. Indeed, my reading of this work is that it presents what seems to be navigation even though it is implemented using a search mechanism. Equally important, this work also points to the benefits of not changing the *location* of data, but rather allowing construction of useful relationships via metadata. Thus, this work supports my ideas of ignoring silo boundaries and providing metadata for use by a similar tool. What it does not explore is a way for data storage to provide enhanced metadata, dynamic update, and notification of changes, which are important elements to making such a dashboard more useful for data visualization.

Chapter 3

Model

In Section 1.3 I attempted to capture key aspects of how naming is used by human users and from that proposed a basic list of key elements to consider in a comprehensive naming model: identity, location, relationship, characteristic, and context. While this is a good place to begin my analysis more is needed to construct a robust model. For example, one of the challenges of ~~aspects of~~ naming is that they are *dynamic*: the storage location of a given data object might change. The *object* is not different but its storage location is: users are unlikely to care about this specific detail until they go to actually retrieve it and find out that location is inaccessible.

To facilitate developing my naming model, I rely upon several use-cases that have arisen during the course of my research around this topic, both working with collaborators as well as discussions for ordinary users that are unfamiliar with my research area.

3.1 Use Cases

To motivate the model I propose for *Indaleko*, I first start with a series of potential use cases. TM ▶*Note that I've started with the two from the original* Indaleko *paper, but I think it might be worthwhile to add one or two more to provide a more well-rounded model.*◀

Data Processing

Compliance

Memex

Asset Management

Feature	**Existing Technologies**	**No Solution**
ACTIVITY CONTEXT	timestamps and geo-location, image recognition, browsing history, ticketing systems, application-specific solutions like Burrito [3].	Link related activity across apps, record browsing history and chat conversations relevant to the creation of the data object, storing it in ways that are secure and compact.
CROSS-SILO SEARCH	Search by name, creator, content across silos, app-specific searches (e.g., Spotlight)	Unified search across all kinds of storage, including file systems, object stores, apps and devices
DATA RELATIONSHIPS	De-duplication of documents, versioning of specific files, git ancestor relation	Explicit notion of data identity, tracking different versions across different silos as data is transformed
NOTIFICATIONS	File watchers (INotify), synchronization status, manually inspecting modified time	Ability to subscribe to specific changes on attributes
PERSONALIZED NAMESPACE	Hierarchy plus hard/soft links. Use of tags.	Creating personalized namespaces with with flexible data organization and views

Table 3.1: Use-case driven functional requirements.

Using these uses cases as motivation, I propose that *Indaleko* support the features as shown in Table 3.1 in greater detail in Section 3.2.

3.1.1 Data Processing

TM ▶*This is from the HotStorage paper submission*◀

Aki and Fenix are preparing a report summarizing their work on a data analysis project for a customer. Fenix sends an email to Aki containing a CSV file with original data. Aki opens this document in Excel, formats and filters it, adds additional data from a corporate storage silo, and then returns the Excel document to Fenix on Slack. Fenix is away from their desk when it arrives, so they open it on their phone, uploading it to a cloud drive. Fenix then sends the link to Aki for editing with update notifications. Finally, Fenix sends a PDF of the report to the compliance officer who promptly asks, “Where did this data come from?” Aki and Fenix are preparing a report summarizing their work on a data analysis project for a customer. Fenix sends an email to Aki containing a CSV file with original data. Aki opens this document in Excel, formats and filters it, adds additional data from a corporate storage silo, and then returns the Excel document to Fenix on Slack. Fenix is away from their desk when it arrives, so they open it on their phone, uploading it to a cloud drive. Fenix then sends the link to Aki for editing with update notifications. Finally, Fenix sends a PDF of the report to the compliance officer who promptly asks, “Where did this data come from?”

3.1.2 Compliance

Delete Request: Some time later, the compliance officer requests that all documents containing a customer’s data must be deleted. To help with finding all relevant customer data, Dagon joins the project and examines the report and requests the original data from which it was produced. Aki remembers that they gave the original data to Fenix shortly after collecting it, but does not remember the name, location, or even how the relevant files were transmitted. Thus, Aki has to manually search possible locations and applications, sendsing references to documents to Dagon, who then starts organizing these files to methodically identify the ones that might contain the customer’s data. In the process, many of the other team members’ references to the documents stop working.

TM ► *Discussion with Ada: how do people do this already? Why are those solutions insufficient? Some sites are accessible and others are not. How do they prove they are GDPR compliant? What about when people move from dynamic to static memory? Could I use storing the hash value as a mechanism for motivating this because it facilitates finding things. Much like the Apple content hash for child p0rn. How about extracting text from pictures to avoid censorship?* ◄

3.1.3 Memex

The “memex” is a device posited by Vannevar Bush in 1945 [2]:

> *“Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, ”memex” will do. A memex is a device in which an individual stores all his books, records, and communications,*