Indaleko

Using System Activity Context to Improve Finding

by

William Anthony Mason

S.B. Mathematics, University of Chicago, 1987

MSc. Computer Science, Georgia Institute of Technology, 2017

A THESIS PROPOSAL SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

The Faculty of Science
(Computer Science)

The University of British Columbia
(Vancouver)

November 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis proposal entitled:

Indaleko

submitted by **William Anthony Mason** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

Joana McGrenere, Computer Science
Examination Chair

Margo I. Seltzer, Computer Science
Co-Supervisor

Ada Gavrilovska, Georgia Institute of Technology, College of Computing
Co-Supervisor

Sasha Fedorova, Electrical and Computer Engineering
Supervisory Committee Member

Norman Hutchinson, Computer Science
Supervisory Committee Member

Andrew Warfield, Computer Science
Supervisory Committee Member

Abstract

Human society is collecting data at an alarming rate: per-capita data generation is now over 1.7MB *per second*. We expect to send 361 billion e-mails *per day* by 2024. Rapid data growth, combined with increasing ways to store and present data to users creates a frustrating challenge finding specific documents a few days old, let alone those created months or years earlier.

Our data is scattered across physical locations. Existing storage is presented to us with old and new interfaces that blur the lines between file system and application. For example, my Outlook mailbox resides on my local disk drive in both databases and discrete files. Yet I use Outlook, not my local file system to find documents that were attached to e-mails.

On-demand cloud storage systems provide strong benefits yet also make it impractical to search locally because not all the content is resident to be indexed. Currently none of the cloud storage systems offer the rich extensible search tools found on modern desktop operating systems. Even if they were to provide such search, it would require querying each one in turn to find relevant files.

To address these challenges I propose *Finding as a Service*, which provides two important capabilities. First, it explicitly decouples *finding* objects from *storing* and *presenting* objects. Second, it exploits the observation that users' mental associations with objects are more complex than the arbitrary name, type, dates, or attributes on which users search today.

Finding as a Service requires two things: (1) a mechanism for exploiting the information that modern devices already capture and for capturing additional useful information that relates to interactions with digital data and the environment in which that data is used; and (2) the ability to collect, store, and query both existing and new usage context pertinent to digital information. Both of these requirements enable building powerful tools that helps users in *finding* digital objects efficiently.

Chapter 1

Introduction

What should I possibly have to tell you, oh venerable one? Perhaps that you're searching far too much? That in all that searching, you don't find the time for finding? — Siddhartha (1922), Hermann Hesse.

1.1 Activity Context

In 1945 Vannevar Bush described the challenges to humans of finding things in a codified system of records [1]:

> *Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.*
>
> *The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the*

intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.

This is as true in 2021 as it was in 1945. Thus, the question that motivates my research is: "Can we build systems that get us closer to that ideal?" My argument that we can by capturing additional information that is not necessarily useful to computers, but is useful to humans.

I call this additional captured information *activity context*. While similar to the ideas proposed in *Burrito* [2], I have broadened that idea beyond just "what a user is doing" to incorporate information about what the user is *experiencing* that corresponds to more human-like context information because it is useful for constructing *association*.

Thus an "activity context" is an answer to the question: *what is going on in relation to the current event on a digital object?* The job of the system becomes answering this question in a way that is useful.

More concretely, activity context is concerned with the environment in which a given digital object is accessed. Without restricting the abstract concept, concrete examples of what I consider to be elements of an "activity context" might include:

- Current weather.
- Notable news events.
- Focus website opened in a visible browser tab.
- User's mood.
- User's heart rate.

This is distinguished from *what the digital object is*. While understanding what something *is* has merit, the context in which a digital object is *used* yields additional understanding about that digital object.

Further examples, in the form of "use cases," can be found in Section 1.4.

1.2 Thesis

Collecting, storing, and disseminating information about system activity ("activity context") enables the use of Information Retrieval (IR) and Human Computer Interface (HCI) research to build powerful tools to improve human *finding* of pertinent digital data.

"Intelligent use of files depends on having sufficient knowledge about them: their purposes, structures, and contexts. Humans have traditionally made do by using their own methods for capturing and manipulating such knowledge, but this is not available to programs, nor is it necessarily convenient for humans [3]."

Determining *context* is a challenge with modern computer storage systems. It is unrealistic to expect human users to provide that context. Context is dynamic, imprecise, and not necessarily obvious, yet humans rely upon context to create associations. By making environmental context information available to programs, those same programs are able to present better options, which *is* convenient for humans.

1.3 Finding

The volume of digital data is growing exponentially. Thus, it is not surprising that solutions that worked when users were grappling with kilobytes (KB) or megabytes (MB) of data do not work in the face of this growing deluge.

IBM's first magnetic disk drive could store up to 2.5 megabytes(MB) [1] of digital data. In 2020, we add 1.7MB of data *per second per person* [2]. Today we have become digital hoarders, collecting and keeping so much data that we often cannot find specific objects when we need them. How did we reach this point?

Early persistent storage systems used a simple flat directory structure that gave a unique name to each object ("file"). Such a simple structure was sufficient to name and identify distinct data objects ("files"). *Finding* the correct file was just a matter of scanning and picking from the list. This simple structure did not scale well and was replaced by a model based upon how paper documents were organized. The hierarchical name space [4]–[7] is one in which files ("digital objects") are grouped into directories (sometimes called folders). Directories can also be grouped into other directories. This model mirrors how a filing cabinet works: multiple sheets of paper are gathered into a folder, folders are organized into drawers, drawers into filing cabinets, filing cabinets into rooms, etc. The directory and file metaphor was in use by 1958 [4] and persists today as the common model despite the volume of data being stored by a single computer storage device ("disk

[1] https://www.7dayshop.com/blog/terabyte-evolution/

[2] https://techjury.net/blog/big-data-statistics/

drive") increasing by at least 10^6 [3].

In addition to the challenges of scaling, the file cabinet metaphor imposes physical file limitations that are not valid for digital data. Physical file cabinets do not allow a document to be in two folders at the same time but there is no such restriction on electronic documents. The Multics researchers addressed this by creating the *link*, an idea that is still used in many modern file systems [5].

Computer networking enabled data sharing between users and computers but complicated naming. Remote data access was typically represented to users either as a hierarchical file system [8], [9] or an application program that programmatically connected users to remote data [10]–[12].

By 1990 the volume of data with which users interacted was so large that researchers questioned the utility of the hierarchical name space [13]. The Semantic File System (SFS) [14] suggested that organization of digital objects be more fluid so that users could group items together in ways that were semantically meaningful. The meta-data generated from semantic information generated from file contents permitted powerful query-based dynamic file organization.

While semantic file systems have not been widely adopted, the concept of extracting semantic information from file content is present in modern file indexing systems. These indexing services employ "transducers" to extract semantic information from files. Desktop search utilities (Windows Search, Apple Spotlight, Station for Linux) rely upon indexing services to provide their functionality.

There are parallels between the challenges of indexing files and the challenges of indexing Internet web pages. Early search engines used a curation model in which humans decided what websites were of interest based upon the information within the web page itself — similar to the way that semantic information was extracted from files in the Semantic File System [4].

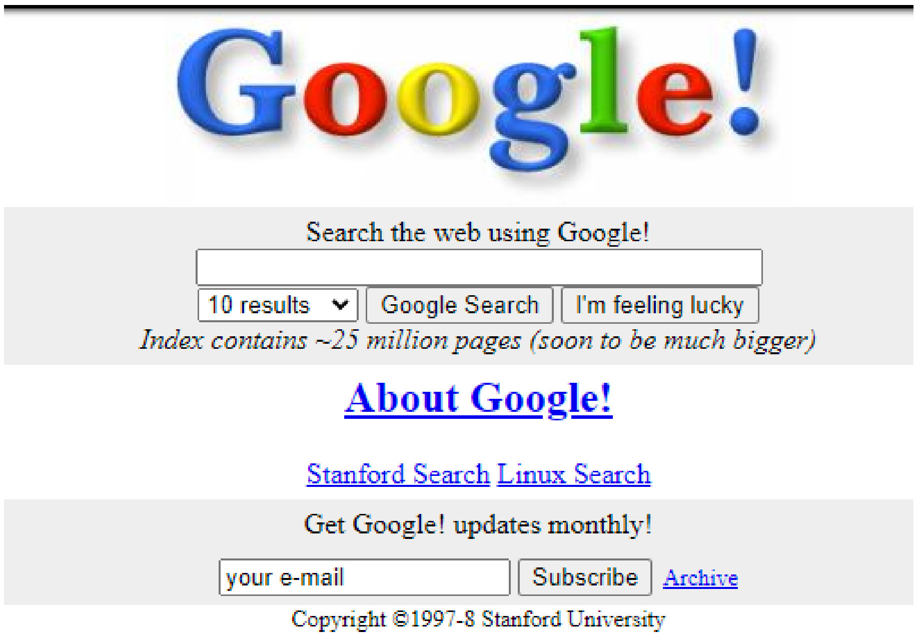
Internet web page indexing changed profoundly when two Stanford graduate students proposed a novel way to exploit the structure of Internet web pages to extract *usage* information from web pages that did not depend upon semantic content [15].

Google's website used to state the number of pages that they indexed. In 1998 the first capture of Google's website by archive.org shows they claimed to index more than 25 million pages (Figure 1.1). Google no longer publishes

[3]Disk drives were measured in MB in 1965 and are measured in TB today.

[4]https://www.hpe.com/us/en/insights/articles/how-search-worked-before-google-1703.html

Figure 1.1: First Wayback capture of google.com (google.stanford.edu) in 1998



Index contains ~25 million pages (soon to be much bigger)

Copyright ©1997-8 Stanford University

that number but industry estimates indicate the number is at least 10^3 more now than it was then, and this only covers a few percent of the entire content stored on the Internet [16] [5].

Could we utilize a similar technique for finding information within our own trove of files? While there are similarities between the Internet and our file collections, there are also significant differences. Files lack the level of common structure present in web pages, preventing simple extraction of references between files. Files (or digital objects) are stored in myriad locations with different access mechanisms: local storage, cloud storage, database, collaboration applications, e-mail programs, etc. Sometimes these overlap: your e-mail program stores some or all of your data on your local computer, within your local storage. However, you do not expect to use the tools for searching your local storage to find things within your e-mail software. Thus, we should consider them to be distinct storage locations. I refer to these distinct storage locations as *storage silos* (or just *silos*) to

[5] https://www.worldwidewebsize.com/

emphasize their inherently separated nature.

Network storage is presented in many different formats: an inexpensive disk drive attached to the local network represents "Network Attached Storage" (NAS) or a specialized parallel data cluster such as HDFS [6], DAOS [7], Lustre [8], or Ceph [9]. They typically support one or more common data sharing protocols, such as NFS [17] or CIFS [10]. They vary dramatically in how they are managed, accessed, and searched. In most cases there is no common interface — each represents a unique "storage silo."

Cloud storage is one specific type of network storage that is popular because it allows you to access your data from any of your devices, provides a reliable backup mechanism, and permits selective download to any given device. However, these benefits are paired with challenges when it comes to finding specific digital objects. If files are not present on your local device, the indexing services on those devices cannot assist you. You could download all of the content from the cloud storage providers to enable indexing, but that consumes considerably more bandwidth and storage and is impractical for devices that have resource constraints. While we can use the cloud providers' search services, that requires iteration over each of those services, using different interfaces with variable results.

Our files come from multiple sources including websites, e-mails, databases, and collaboration tools. Those documents are stored both locally and remotely. We create, access, and modify documents and then send them onwards using any of the variety of silos and collaboration tools at our disposal. Just a few days after we last accessed them we struggle to find those documents.

Given the diffusion of files across storage silos that occurs because of our sharing and use, we often find versions and related files scattered across multiple silos. We struggle to find these versions and determine when we have found the "right" one.

Returning to the question of using contextual information for improving *finding*, the research community has observed that adding contextual information, such as the current weather, to existing file collections materially improves human ability to find the relevant digital object [18]–[20]. Thus, it

[6] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[7] https://www.intel.ca/content/www/ca/en/high-performance-computing/daos-high-performance-storage-brief.html

[8] https://www.lustre.org/

[9] https://docs.ceph.com/en/pacific/cephfs/index.html

[10] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cifs/d416ff7c-c536-406e-a951-4f04b2fd1d2b

seems the answer is "Yes, using contextual information improves *finding*."

Google solves a simpler problem: an Internet web search need merely find *an* answer to the search query. A personal file search needs provide *the* answer. Thus, it may not be possible to provide a definitive answer, but narrowing the potential list of plausible answers leads users to the relevant file, which is the goal of *finding*.

The data that we need for creating activity context is already being collected by our computers. Modern computers collect vast amounts of information about us: what we do, where we are, with whom we communicate, the applications we use, the files we access, the music we play, the web pages we visit, even how we feel [21], [22]. We *know* this data exists because our own devices provide this information to third parties. Given this data is already being collected, we know the additional cost will be to store and make it accessible to applications.

Finding as a Service (FaaS) will use this existing information to solve our data finding problem. FaaS decouples *finding* objects from *storing* objects. FaaS facilitates *finding* by exploiting contextual information beyond the basic object characteristics widely available for searching today: names, types, and dates. By relating information we already have with how our digital objects are used, we provide the *activity context* to enable *Finding as a Service* (FaaS).

Activity context is important because it captures useful information about the environment in which files are created, consumed, and updated. *Activity context* need not be something the system ordinarily relates to the digital object. *Activity context* captures key information about the the user's wholistic environment.

1.4 Use Cases

The following use cases provide specific scenarios that cannot be achieved using current systems. I maintain that *Finding as a Service* addresses these use cases, which supports my thesis.

- **The lost original.** Imagine that you received a spreadsheet from someone. You begin to edit it, add information, sift through it. At some point you realize that you sorted a subset of the columns, hopelessly scrambling the original information and your edits. You try to find the original source of the spreadsheet, only to realize that you cannot do so, even when you search in your e-mail program using the *name* of the file that was saved on your local drive as part of your editing process. The system should permit you to find the original source of the information, even though the related digital objects are in different *silos*.

- **The misplaced presentation.** You arrive at a meeting with your client after a long trip, only to realize the laptop computer you were using will not boot. You have your smart phone and you *think* you saved it to a cloud service. How do you find it so you can share it with a colleague at the meeting? The system should permit you to find your own digital data in your cloud storage regardless of which device you used to create it.

- **The multi-silo relationship problem.** A colleague shares their experimental data with you, which was stored in NREL's High-Performance Computing Data Center [11]. You then use that data as part of your own work, which you wish to share with a broader audience using Compute Canada [12]. You also shared your computational notebooks using your organization's account with Microsoft [13]. You created your slides in Prezi [14] and presented them to a different research group on their Discord server [15]. One of the people that attended downloaded two of your notebooks and created a new notebook from them. They then shared that notebook publicly via Google Colab [16]. Your colleague knows that she shared her data with you and wants to be able to quickly find the documents that you and others have shared based upon that original data. Collaborative work like this is a modern reality and it is unlikely that all work product will be co-located. The system should permit your colleagues to find the work you and others have shared with her *without* your intervention.

- **Multi-silo finding.** Hao is a visiting student from the country of Lemuria doing an internship with you in Camelot. While arranging for this internship, Hao required *numerous* different data objects: email messages with the host, offer letters, academic forms, a visa, boarding passes, project proposals, and more. The system should be able to provide you with a set of related files, regardless of their storage silo.

[11] https://www.nrel.gov/computational-science/hpc-data-center.html

[12] https://www.computecanada.ca/techrenewal/rdm/

[13] https://visualstudio.microsoft.com/vs/features/notebooks-at-microsoft/

[14] prezi.com

[15] discord.com

[16] https://colab.research.google.com/

- **The where did I get this information conundrum.** In our modern world we often have one (or more) web pages open when we are authoring one (or more) documents. A reasonable question to ask would then be “what web pages did I look at while writing this document?” Often it is not just that you looked at a given web page, but also if it was the last web page you looked at and how long you looked at it. The system should be able to provide you with a list of that web activity.

- **How do I share information while preserving privacy?** Zene, an investigative journalist who routinely receives sensitive information from third parties, is investigating the company from the prior use cases. Zene needs to be able store and access sensitive information, including information about the activity context of various e-mails, documents, pictures, and audio and video files. While Zene ensures that these data are encrypted, they need to also ensure that they can both find information and ensure that meta-data associated with those files is both usable and properly protected across silos. While Zene must protect their sources, they must also be able to associate evidence with those sources to make judgement calls about their validity. The system should support security and privacy policies for attributes that accomplish both.

This list of use cases is not exhaustive and is intended to provide cases that resonate with readers. They are use cases that are not addressed by existing systems. I review these existing systems and how they fail to address these use cases in Section 1.5.

1.5 Existing Solutions Fall Short

Prior work has addressed some of the challenges that I identified in the use cases (Section 1.4). I briefly introduce key aspects of how they fall short here and provide greater detail in Chapter 2.

A simple solution to the multi-silo namespace challenge is to graft those namespaces together. UNIX mount points [23] are perhaps the first instance of such federating namespaces. Distributed federation, as provided by distributed file systems such as NFS [8] and AFS [9] emerged in the 1980s soon after adoption of high speed networks such as Ethernet [24].

There is some work in cloud storage federated namespaces [25], [26].

Nextcloud [17] allows users to connect multiple Nextcloud instances and integrate with FTP, CIFS, NFS and object stores. This yields a classic hierarchical namespace structure with its known limitations [13], [27]. It does nothing to facilitate *finding*. Peer-to-peer sharing networks (e.g., IPFS [28]) implement a distributed file system where nodes advertise their files to users. MetaStorage [29] implements a highly available, distributed hash table, similar to Amazon's DynamoDB [30], but with its data replicated and distributed across different cloud providers. MetaStore offers a key-value store interface [18]. Farsite [31] organizes multiple machines into virtual file servers, each of which acts as the root of a distributed file system. Comet describes a cloud oriented federated metadata service [26].

None of these address the scaling problem that arises when data is analyzed and indexed away from where it is stored. Similarly, none of these address the integration of sensitive locally stored information about personal usage of these objects. Thus, one key benefit of better *finding* is that it should improve the efficiency of retrieving data stored across non-local storage silos.

Some prior work explored using extrinsic usage information for *finding*. Placeless [32] focused on using process level information extracted from their document processing system to associate files together. Similarly, Burrito [2] proposed *activity context*, which they define as "the user's actions at a particular time." Both look at narrow instances of the larger *finding* problem.

Provenance uses observable information about construction of a file to augment file search, which in turn improves findability [33]. Provenance search takes a narrow view of the activities of interest and are all largely *causality* focused. However, humans tend to think associatively [19], focusing on what else was happening — the cleaning crew came by their desk while they were writing that document, the discussion at a meeting with others, their location when they wrote a given document, or some other event that was happening around the time they interacted with a given document.

While environmental information is not as obviously related as causal relationships, prior work related to using statistical inference to establish relationships within the storage domain has demonstrated such mechanisms can be more efficient at identifying patterns that lead to higher efficiency [34].

[17] https://nextcloud.com

[18] https://cwiki.apache.org/confluence/display/hive/design

1.6 Contributions

The research to support my thesis will contribute the following:

1. Production of the *Finding as a Service* (FaaS) dataset, a collection of meta-data and activity context from a local system, that will enable me to explore potentially useful information for informing activity context data collection. In addition, I will publicly share this data set to enable other researchers to develop new techniques for users to find digital items.

2. *Indaleko* [19], my architecture for a system that captures, stores, and disseminates *activity context* without imposing excessive resource demand.

3. *Topish* [20], a single node implementation consistent with *Indaleko* that provides *FaaS* across multiple storage silos on a single system.

4. *Находка* [21], a distributed implementation of *Indaleko* that provides *Finding as a Service* (FaaS) across multiple systems using a combination of device private and cross-device shared storage silos.

5. An evaluation demonstrating that it is possible to capture activity context without imposing excessive overhead, in either space or time.

These projects focus on improving *finding*.

The remainder of this document provides more specific insight into these contributions and how I propose creating and disseminating them. In Chapter 2 I review the prior work that underlies my thesis: what types of storage silos exist, what information we already have available and why these are not sufficient to meet these use cases. In Chapter 3 I set out the research questions that I seek to answer to fully explore my thesis. In Chapter 4 I describe the structure of the system I propose building in order to support my thesis and how it addresses these use cases. In Chapter 5 I discuss how I propose evaluating my system. Specifically I attempt to address key questions, such as: "how well does it address these use cases?", "what are the

[19] *Indaleko* is Xhosa for pragmatics. In Linguistics, pragmatics is the study of meaning within a given context

[20] *Topish* is the Uzbek word for finding; the most recent graduate student from our research group is from Uzbekistan and has always been supportive of my research

[21] *Находка* is the Russian word for finding in recognition of the support for my research that I have received from both Ada Gavrilovska and Alexandra Fedorova.

performance and resource implications of using my system?", and "how well does it enable other communities to construct more effective finding tools?"

Chapter 2

Background

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).

Key topics crucial to understanding my proposal are:

1. Why finding is important. I discuss the related background in Section 2.1.
2. How to help people find things. I discuss the background related to finding things in Section 2.2.
3. Storage Access mechanisms. I discuss the APIs that are available to various services in Section 2.3.
4. Existing Meta-data. I discuss existing meta-data that are already known to exist and can be extracted in Section 2.4.

This information is useful in better understanding the architecture proposed in Chapter 4.

2.1 The Importance of Finding

In Section 1.1 I provided a basic definition of activity context and described Vannevar Bush's 1945 work observing the difference between computer index systems and human associative memory [1].

While preparing this proposal I spent time looking at the guides many libraries provided about the naming of files. I found a body of recommendations about file naming standards from significant academic and governmental sources and summarize that in Table 2.1. These recommendations are consistent with prior work [35].

In Table 2.1 I provide links to a number of organizations that publish recommendations for "naming files." While they vary somewhat, there is far more similarity than difference. Harvard's list is:

- Think about your files
- 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

The important observation here is how we rely upon the file name to provide context for *what* a given file represents. Uniformity of information is important — the "naming convention" permits not only identifying similarity but key elements of *difference* between any two named things.

It is difficult not to look at this as a modern indictment of a system that is fundamentally broken: requiring users understand meta-data, versioning, encoding, *and* capturing the naming represents a significant cognitive burden.

Saltzer pointed this out as well: "This approach forces back onto the user the responsibility to state explicitly, as part of each name, the name of the appropriate context [7]."

The *purpose* of the file system was to serve as the provider of "human-oriented names" [7, Table III]. Mogul observed that "Better file systems allow us to manage our files more effectively, solve problems that cannot now be efficiently solved, and build better software [3, p. 1]." Gifford observed: "[A] semantic file system can provide associative attribute-based access to the contents of an information storage system with the help of file type specific transducers... The results to date are consistent with our thesis that semantic file systems present a more effective storage abstraction than do traditional tree structured file systems for information sharing... [p. 22][14]"

Table 2.1: Sample Academic and Governmental Naming Conventions

University of Cambridge	https://www.data.cam.ac.uk/data-management-guide/organising-your-data
Harvard University	https://datamanagement.hms.harvard.edu/collect/file-naming-conventions
Smithsonian Institution	https://library.si.edu/sites/default/files/tutorial/pdf/filenamingorganizing20180227.pdf
Leland Stanford, Jr. University	https://library.stanford.edu/research/data-management-services/data-best-practices/best-practices-file-naming
United States National Institute of Science & Technology	https://www.nist.gov/system/files/documents/pml/wmd/labmetrology/ElectronicFileOrganizationTips-2016-03.pdf
University of British Columbia	https://researchdata.library.ubc.ca/files/2019/01/FileName_Guidelines_20140410_v03.pdf
University of Chicago	https://guides.lib.uchicago.edu/c.php?g=565143&p=3892706
University of Toronto	https://onesearch.library.utoronto.ca/researchdata/file-management
University of Washington	https://itconnect.uw.edu/learn/workshops/online-tutorials/web-publishing/web-publishing-at-the-uw/internet-file-management/4

Saltzer challenged us with general naming but set it aside for future research. Mogul captured the idea that *properties* could be used to store additional meta-data about files. Gifford explored the idea that information about *what* a file represents could be extracted and used to dynamically organize files based upon semantic content of the files themselves.

Despite these insights, the tools we have for *finding* remain primitive. Forcing users to embed context is a *naming* solution, but it creates cognitively challenging requirements such as "naming conventions." Similarly, creating tagging mechanisms using extended attributes or properties has not provided sufficient benefit to be broadly used. Semantic file systems have been implemented in modern indexing services and are somewhat useful but have clearly *not* solved the problem.

All of these solutions are "inward focused." That is, they focus on the *file* (or object): its attributes, which includes its name and its contents. They fail to understand the file's usage context: it's relationship to other files and *to other events in the user's environment.* Thus, the relevant prior systems work falls short of evaluating my thesis.

2.2 Useful Information for Finding

I summarize prior work useful to this proposal in Table 2.2. In some cases the precise information about what data was used is not clear from the available materials and thus is not included in Table 2.2. The table identifies the specific information, what *type* this represents, and the reference that provides this information. My classification of *type* is based upon previously proposed types [37] of *who*, *when*, *where*, *what*, *why*, and *how*.

The remainder of this section reviews the literature on which Table 2.2 is based and explains how it relates to my broader thesis.

Search is one tool that assists in *finding* but it is not the first choice for many human users [43]. Thus, improvements in search are only beneficial at the point that the existing organizational system has already failed.

The Human-Computer Interface community has provided an insightful definition of good design:

> *A well designed computer system permits use of the tools it offers without requiring users to dedicate extensive mental processing to operations inherent in the system design rather than the task. Furthermore, its tools are designed to also reduce task-specific mental processing, especially those types of processing that are performed more effectively by computers than by people,*

Table 2.2: Useful Context Information (See Section 2.2)

Information	Dimensions						Reference
	Who	When	Where	What	Why	How	
Social Media	X	X	X	X		X	[36], [37]
E-mail	X	X		X		X	[38]
Webpage	X	X	X	X		X	[38], [39]
Document [a]	X	X	X	X		X	[37]–[40]
Audio		X				X	[41]
Image		X	X			X	[41]
Video		X	X		X		[41]
Applications	X	X	X	X		X	[33], [40]
Behavior	X	X	X	X	X	X	[33], [40]
Change	X	X	X	X	X	X	[33], [40]
Calendar	X	X	X	X	X	X	[36], [37], [42]
Paths	X	X	X	X	X	X	[33], [37]
GPS		X	X			X	[36]
File System [b]	X	X	X	X		X	[33], [36]
Weather		X	X	X		X	[36]
Financial Data	X	X	X	X		X	[18]
New Suggestions							
Voice/Video Calls	X	X				X	
Collaboration [c]	X	X	X	X		X	
Music	X	X		X		X	
Medical Info [d]	X	X		X	X	X	

[a] PDF,RTF,Word, etc.

[b] The distinction between *paths* which involve understanding application behavior and file system activity, which is about specific files or directories demonstrates potential complexity due to data collection from multiple sources in the same system.

[c] Discord, Slack, Teams

[d] Wearable Monitor, Insulin Pump

> *such as calculations and* ***accurate storage and recall of large amounts of pre-specified information*** *[44]*[bold face is my addition].

Information retrieval researchers are thus more interested in evaluating the outcome of a human finding the object they seek, rather than the efficiency of the underlying search infrastructure [45]. The need to find things is pervasive: clearly there are obvious things like the need to access a particular document, but there are other important needs, such as the need to find objects that must be removed, such as when it is required to comply with legal obligations, such as the "right to be forgotten" under the GDPR [1] [46]. Some users prefer to delete content they know they will no longer need [47]. While the reasons for finding and removing content vary, the ability to do so efficiently is directly related to *finding* that information.

Prior work has suggested a number of different factors that can be used to find content:

- Naming similarity; similarly names files are often related.
- Content similarity: files with similar content are often related.
- Temporal similarity: files that were created and/or accessed around the same time are more likely to be related [48].
- Causality: files that are created using the same tool are more likely to be related.

Prior work has also observed that using contextual information from "personal digital traces" clearly assists in finding relevant personal data: "Work in Cognitive Psychology has shown that contextual cues are strong triggers for autobiographical memories [37]." The source code for the data collection tools used by the authors is still available, albeit dated [2].

The *finding* problem is one at the heart of the personal information management research efforts, whose work suggests that one reason search is not preferred is that it *takes longer* than navigation [43]. The semantic file system work fits well with this observation, as one of its primary contributions was the observation that search results can be represented as "virtual directories," which provides a means of presenting search results as a form of navigation [14].

[1]https://gdpr.eu/right-to-be-forgotten/
[2]https://github.com/ameliemarian/DigitalSelf

Similarly, prior work established the need to be able to support a broad range of storage locations — the "storage silos" that I have previously mentioned. In *Stuff I've Seen* researchers found that the ability to search across silos in a uniform fashion led to increased utilization of such tools [49]. Indeed, the observation in the personal information management community repeatedly stresses the importance of supporting cross-silo management of digital data.

This makes sense: when we are looking for a specific object that we know exists, we do not particularly care *where* it is stored.

The research on "table top interfaces" is another useful example of how the HCI field is exploring alternative interfaces. Their findings include the fact that hierarchical structures do not work well in collaborative table top systems [50]. There are similarities between table top interfaces and mobile interfaces in terms of their interaction models, e.g., no keyboards or mice. The mobile device solution initially was to create silos for each application's files. While it freed the applications and users from the underlying hierarchical name space, it led to an explosion in the number of silos on a single device. Some applications (e.g., cloud storage on mobile devices) still expose hierarchical interfaces but even they tend to demote the hierarchical name space in favor of other presentation models. In essence, the virtual directories of the semantic file system have become the primary interface.

The *MyLifeBits* project followed one person's goal of organizing their own data: "We hoped to substantially improve the ability to organize, search, annotate, and utilize content. Also, we wanted to obtain a unified database in contrast to the many data "islands" being created including mail, contacts, and meetings, finances, health records, photos, etc. Frustration with the file system led to testing the suitability of databases for personal storage, and ultimately into research about next generation storage systems [41]." Note that the authors identified many of the same problems that still exist today, including the multi-silo problem, yet those problems remain unsolved.

The use of richer contextual information to better search and organize data is one that seems to be a perennial favorite for greater exploration. The database community has observed specific types of context that are useful: applications, behavior, and change: "In decoupled systems, behavioral context spans multiple services, applications and formats and often originates from high volume sources [40]…"

The use of "personal digital traces" is close to the work that I have proposed as part of my thesis. The authors observe: "Search of personal data is usually focused on retrieving information that users know exists in their own data set, even though most of the time they do not know in which source or device they have seen the desired information. Current search tools such as Spotlight and Gmail search are not adequate to deal with this scenario where the user has to perform the same search multiple times on different services or/and devices rather than search over just a single service. Besides, traditional searches are often inefficient as they typically identify too many matching documents. [37]"

Finally, I note that the importance of context spans disciplines. The study of *pragmatics* in Linguistics relates to the understanding of meaning within the context in which it is used: "Pragmatics is a field of linguistics concerned with what a speaker implies and a listener infers based on contributing factors like the situational context, the individuals' mental states, the preceding dialogue, and other elements. [3]" Closer to home, the database community has explored the importance of context in terms of human understanding: "Context has often a significant impact on the way humans (or machines) act and on how they interpret things; furthermore, a change in context causes a transformation in the experience that is going to be lived. The word itself, derived from the Latin *con* (with or together) and *texere* (to weave), describes a context not just as a profile, but as *an active process dealing with the way humans weave their experience within their whole environment, to give it meaning.* [51]"

2.3 Storage Silo Access

The number of novel storage implementations is large. For example there are dozens of file systems actively in use [4]. In addition, there is a constant stream of new proposed variants [52], [53].

Thus, rather than review the myriad of file systems that exist and contribute novel storage silos, I instead focus on classifying storage silos by the interface that is used to access them.

- File system APIs. Most file systems use or support a POSIX like interface, which typically includes create, open, close, read, write, delete, rename, and read directory. In addition, most local file systems provide monitoring interfaces, which permits monitoring state change.

- Object Store. The OpenStack Object Store interface provides a useful definition of a robust implementation that maps to the HTTP proto-

[3]https://www.masterclass.com/articles/pragmatics-in-linguistics-guide

[4]See https://en.wikipedia.org/wiki/Comparison_of_file_systems

col quite closely [5]. Object stores are commonly used because of their simplicity and sufficiency for a range of uses in both device local and internet enabled applications. Thus, object stores normally support an authentication protocol and object access protocol using get (retrieve object contents and meta-data), put (object create or update), copy, delete, head (retrieve object meta-data), and post (update object meta-data).

- Cloud storage. There is a greater range of APIs for cloud storage, with each implementation typically providing documentation. Many of these consist of "Web APIs" which are implemented using the HTTP or HTTPS protocols. Areas of common functionality, albeit varying implementation, are authentication, file access, and file change notifications. Google Drive, Dropbox, Amazon S3, and Microsoft OneDrive all support mechanisms for authentication, file access (including meta-data access) and file change notifications. They do not use a common API for doing this so that an importation/interaction layer must be written for each one of them; ideally I expect to be able to produce a common event format that I can use with all of them.

- Databases. There are several types of common databases, including relational databases such as Oracle, MySQL, Microsoft SQL, PostgresSQL, IBM DB2, Sybase, and Teradata. They all support some variant of the common structured query language which is typically accessed via programming libraries that simplify the various differences. Non-relational databases have become increasingly popular in recent years. Examples of commonly-used non-relational databases include MongoDB, Cassandra, Redis, and Neo4j and can be classified as document stores, column store, key-value stores, and graph stores.

- Applications. While applications *also* tend to consume services from other storage layers, the context of those operations is often not visible. A file system has no way of knowing that the file just created by the e-mail program was an attachment, rather than a data file used by the application program itself — but the application is aware of this important contextual understanding. There is far less structure or regularity of applications. These applications tend to have their own unique interfaces. For example, graphical user interface based "file browsers" often have "hooking" interfaces that permit intercept-

[5] https://docs.openstack.org/api-ref/object-store/

ing higher level operations such as “copy a file.” Collaboration applications such as Slack, Discord, and Teams also have extension interfaces for interacting with them but these interfaces do tend to be specific to the application. Commonly used non-web based e-mail programs such as Thunderbird and Outlook have public APIs for building extensions.

One of the challenges in trying to create a classification system for “storage silos” is that the dividing line is often not clear. For example, if one accesses an Oracle database via a REST API, is that a database or cloud storage? For the purposes of understanding the range of APIs this general breakdown is sufficient and allows me to identify broad categories of silos to consider using in my work.

2.4 Existing Meta-Data

The volume of available meta-data is high enough that one of the challenges I face in conducting experiments to evaluate my hypothesis is coping with the volume of information.

My summary of useful information in Table 2.2 includes references to prior work that draws upon existing meta-data. The work regarding collection of personal digital traces is particularly germane, because not only did the authors identify useful information, they made their own tools publicly available.

Beyond this, I can point to existing meta-data sources that I know exist and can be used as part of my own work. One is the extended Berkeley Packet Filter (eBPF) support that is available on Linux and being added to Windows. eBPF is an extensible framework for collecting data from the running operating system by injecting “hooks” that allow detailed monitoring. There are already existing eBPF filters that provide extensive meta-data. For example, the Linux OSQuery interface has an eBPF alternative backend for data collection. The community developing and extending eBPF is quite active and the scope of information already available is extensive [6]. From the perspective of testing my thesis, I expect it will not require extensive development of new software.

Windows has an operating system level introspection package known as Event Tracing for Windows [7] (ETW) as well as Microsoft’s recent work on supporting eBPF on Windows (likely by leveraging their ETW work) [8].

[6] https://ebpf.io/blog/ebpf-updates-2021-02

[7] https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing

[8] https://microsoft.github.io/ebpf-for-windows/

Both ETW and eBPF should provide ample existing meta-data from which to draw upon. Combined with the Personal Digital Tracing tools [36], there is a rich set of existing meta-data from which to draw upon, making the design and implementation of the tools I need to test my thesis simpler.

Chapter 3

Research Questions

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* [54]

The goal of my research is to develop a framework for gathering and disseminating rich data tracking activity across a user's silos and devices to facilitate end-user *finding*. While the evaluation of these finding questions lies in the domain of HCI researchers, my research must enable the collection, aggregation, storage, and querying the data that these researchers can use to evaluate different finding approaches. Thus, my research will answer the following questions:

1. **What data comprises activity context?** As previously observed (Section 2.2) there is a wealth of potential information that could be included in the activity context. This question asks: "what data should be included in the activity context?"

2. **How do we capture activity context?** While there are numerous different potential sources for activity information, how do we capture and store it?

3. **How do we collect rich data (including activity context) from multiple storage silos and make that rich data accessible efficiently?** We know from prior work that one can capture *all* state for a

single computer system with surprisingly moderate cost [55], but that approach does not make the collected state easily available. Thus, collecting and storing the rich data is not sufficient, we also need to make it accessible to applications. How do we make it available to applications in a manner whereby the benefits of better findability outweigh any impact on application performance?

4. **How do we facilitate query of this rich meta-data?** It is important that both users and applications have a mechanism via which they can exploit this extensive collection of rich meta-data. What does the query interface for this rich meta-data look like?

5. **How can applications leverage this rich meta-data?** Applications must programmatically find the documents with which they interact. Most applications use temporal relationships, such as "recently accessed documents," as a primary mechanism to present users with a set of files they might wish to access. When that fails, they fall back to offering the user files in a directory or list that the application deems 'likely'. This question asks: "if an application has access to activity context data, how can it use the information to give users a better collection of candidate files from which to select?"

6. **How do we preserve the privacy of sensitive meta-data?** In a model that incorporates extensive amounts of personally identifiable information there is a very real risk that this information will prove to be economically valuable to someone; how do we ensure that the user retains ownership and control of that information so that it is only released when they approve doing so?

I intend to provide data that will allow other researchers to answer related questions such as:

- What relationships are most valuable in helping users find data?
- Does activity context provide better information for user search than semantic information alone?
- What interfaces best allow users to leverage rich meta-data search capabilities?

The questions themselves that are core to my proposed thesis work revolve around exploring and broadening the of an *activity context*. However, activity context by itself does not replace the other prior work that has added semantic understanding around files, e.g., semantic file systems, formal concept analysis, tag file systems, graph file systems, etc. Thus, my proposed system (Chapter 4) includes support for this prior work but extends it by augmenting previous mechanisms with this additional rich usage information that I call “activity context.”

Chapter 4

Architecture

We are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size. — Metalogicon (1159) John of Salisbury.

My goal with this architecture is to capture a broad description of the services I anticipate are required to achieve my goals. To that end, my architecture seeks to provide a broad framework on which my own work as well as potential future work can be constructed.

My focus in supporting my thesis will include building specific, likely limited, implementations of tools that fit within my architecture. These tools will then allow me to answer the research questions described in Chapter 3.

In keeping with good software architectural principles, I strive to ensure the architecture is sufficiently general and not necessarily tailored narrowly to my particular task: that narrowing can be done as part of my own tool development.

4.1 Features

The tool I propose constructing incorporates support for existing functionality as well as the new functionality that I propose adding. In Table 4.1 I have set out the specific features that I anticipate providing by the tool. In turn, I use these features when constructing my proposed architecture.

Feature	**Existing Technologies**	**No Solution**
ACTIVITY CONTEXT	timestamps and geo-location, image recognition, browsing history, ticketing systems, application-specific solutions like Burrito [2].	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 flexible data organization and views

Table 4.1: Use-case driven functional requirements.

4.1.1 Activity Context

As Burrito demonstrated [2], the *context* in which data were accessed or created is often a useful attribute on which users wish to search, e.g., "*I'm looking for the document I was editing while emailing Aki about their favorite wines.*"

To the best of our knowledge, there is no modern system that supports queries using rich context across applications.

I might be able to use timestamps or application-specific tags or history information in queries, but it is laborious, if not impossible, to intersect data from multiple applications and/or multiple silos.

4.1.2 Cross-silo Search

Users share documents in myriad ways: via messaging applications, on cloud storage services, and via online applications. Users should not need to remember which mechanism was used to share a particular document and should have some easy way of organizing and searching through a collection of such distributed documents.

4.1.3 Data Relationships

Documents can be related in arbitrary ways. This relationship information can be used to facilitate and enable better search results. So far, I have identified three specific relationships that are particularly important:

1. *copy* is a bit-for-bit identical replica of some data, in other words two items with different names store the same data. Deduplication functionality in storage systems frequently takes advantage of the prevalence of copies to reduce storage consumption. However, knowing that two items with different names are, in fact, the same is also valuable information for *users*.

2. *conversion* is a reversible, repeatable transformation that changes the representation of data, without changing its semantics, e.g., converting a CSV file into JSON.

3. *derivation* refers to data that has been computationally derived from another object by altering its content, e.g., adding a row to a spreadsheet.

Classifying the relationship in such cases may not be obvious: if I export an Excel format spreadsheet as a CSV file, I may lose data such that the relationship is a *derivation* rather than a *conversion*. A conservative implementation would thus define this as a *derivation* in the absence of knowledge that an inverse transformation exists.

While storage systems can recognize copies, they cannot distinguish conversions from derivations. However, from a user's perspective, these operations are quite different: a conversion can be repeated, which is not necessarily true of a derivation.

4.1.4 Notifications

Users frequently want to be notified when documents change, and many storage services offer this functionality.

However, users might also want notification when data on which they directly or indirectly depend changes. This requires both a notification system and an awareness of the data relationship between different objects.

4.1.5 Personalized Namespaces

Users have different preferences and mental models to organize their documents, frequently a source of conflict in a multi-user setting. I need a way to provide each user the ability to personalize their document structure.

4.1.6 From Use Cases to Architecture

Recall that in Section 1.4 I provided a series of use cases for consideration. Each use case and feature presents a situation that cannot be solved with our existing mechanisms within the context of the multi-silo world.

In Table 4.1, I identify existing technology that can be brought to bear on the problem, while teasing apart the precise details that are missing.

Repeatedly, I find that critical information necessary to provide a feature is unavailable, that providing such information is non-trivial, and that obtaining it creates a collection of privacy challenges.

4.2 Proposed Architecture

Indaleko is a family of services that enable sophisticated search and naming capabilities. The key features that differentiate *Indaleko* from prior work are:

1. incorporating object relationships as first class meta-data,
2. federating meta-data services,
3. recording activity context,
4. integrating storage from multiple silos, and
5. enabling customizable naming services.

Data continues to reside in existing and to-be-developed storage silos. *Indaleko* interacts with these silos, collects and captures metadata, and provides a federated network of metadata and naming services to meet the needs of the use cases in Table 4.1.

4.3 *Indaleko* Services

Figure 4.1 illustrates the *Indaleko* architecture. *Indaleko* allows for different deployment scenarios. The five services can be run independently, they can

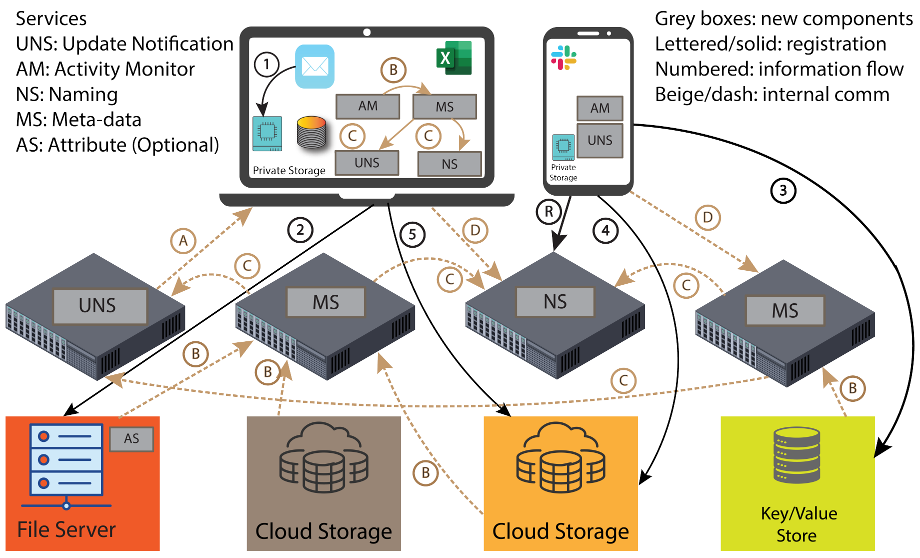


Figure 4.1: *Indaleko* Architecture (Section 4.2).

be co-located and bundled together to run on a local device, integrated into an OS, or available as web-based services.

In the balance of this section, parenthesized numbers and letters refer to the arrows in Figure 4.1. There are five main components:

1. **Metadata servers (MS)** are responsible for storing attributes and provide a superset of capabilities found in existing metadata services [26], [56]. Users can register a Metadata Server with activity monitors or attribute services, which allows the Metadata Server to receive updated attributes from storage objects and activities (B). Thus, there can be multiple sources of attributes including the user itself. Metadata servers may retain the full or partial history of attribute updates or maintain only the most recent value.

2. **Namespace servers (NS)** connect to one or more Metadata Server and use the metadata to provide users with a personalized namespace that allows both manual organization (i.e., a hierarchical namespace) and rich search capabilities. The benefit of supporting a hierarchical namespace is that it provides a path for backwards compatibility as well as a mechanism for enabling virtual directories [14] to enable existing applications to benefit from the enhanced capabilities of *Indaleko*. The benefit of rich search capabilities is that it enables us to

build those virtual directories for use by the hierarchical components as well as propose and evaluate alternative data exploration tools. Users can register with a Namespace Server (R) that uses one or more Metadata Servers to obtain relevant attributes from them (C). Additionally, users can be part of a corporate Namespace Server that allows sharing of their select metadata with other users via standard enterprise public-key cryptography.

3. **Activity monitors (AM)** run on the user's devices. Their main function is to observe temporal relations, activity context, and relationships between objects on a user's device and transmit them to a Metadata Server (D).
4. **Attribute services (AS)** extract attributes from storage objects and transmit them to an Metadata Server (B). An Attribute Service might be invoked on updates, run once or periodically. For example, a file system Attribute Service would update the object's metadata with basic attributes such as size or modification time. There can be many Attribute Services that extract more "interesting" attributes, e.g., image recognition, similarity, or other classifiers.
5. **Update notification server (UNS)** provides notification mechanisms. Users can register interest in changes of attributes or underlying storage and will receive a message on change events (A) to which they have access.

In Figure 4.2 I show a simplified image depicting how this might work on a single computer system, with data collection from a variety of sources on the local computer, including resources that are both local to the system and remote resources accessible and in use on the computer. Information is ingested by the local *Indaleko* components and then presented to applications via a query interface; one likely use of this query interface would be using it to form "virtual directories" that allow legacy application interactions with the namespace. Unlike Figure 4.1 this diagram omits details of the internal structure and instead explains one way in which it could fit into the local device's environment.

4.4 *Indaleko* Working Example

To make the *Indaleko* architecture concrete, I revisit our use-cases from Table 4.1 and walk through the cases to illustrate how *Indaleko* supports the various actions and events.

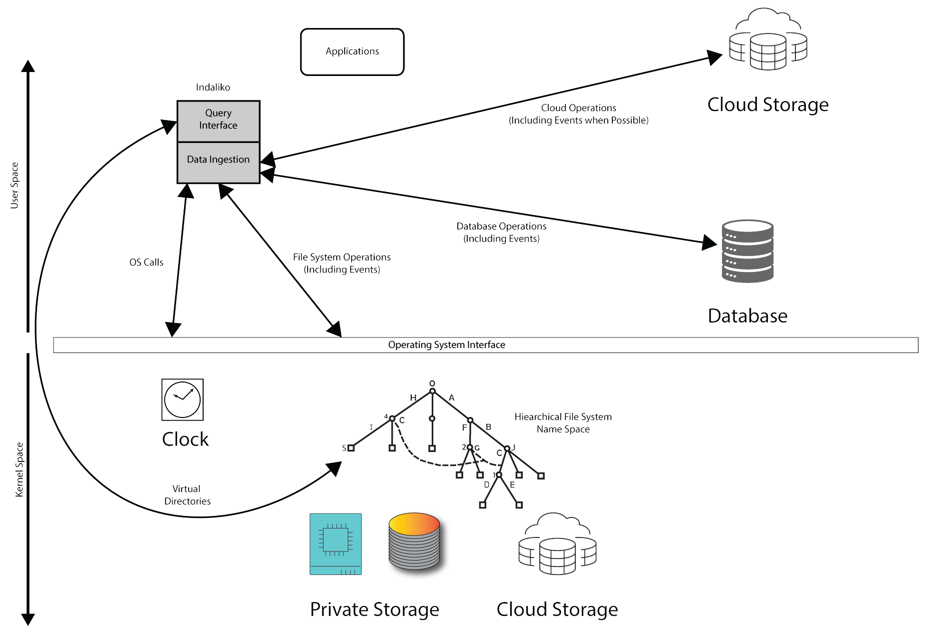

Applications
Indaliko
Query Interface
Data Ingestion
Cloud Operations (Including Events when Possible)
Cloud Storage
Database Operations (Including Events)
Database
OS Calls
File System Operations (Including Events)
User Space
Operating System Interface
Kernel Space
Clock
Hiearchical File System Name Space
Virtual Directories
Private Storage
Cloud Storage

Figure 4.2: Single computer components for *Indaleko*.

4.4.1 Storing the e-mail attachment

Aki's act of saving the CSV file that Fenix sent in email corresponds to the creation of a new object on the cloud storage silo, i.e., the file system (4). The file server is *Indaleko*-aware, so the Attribute Service co-located with it extracts attributes from the document and forwards them to the Metadata Server (B).

The Activity Monitor on Aki's laptop detects that the CSV file came via company email from Fenix. It then captures the activity context identifying the relationship between the e-mail and the CSV file and transmits it as additional metadata about the CSV file to the Metadata Server (that already contains metadata extracted by the Attribute Service). Moreover, because there is a company-wide namespace service, *Indaleko* establishes that the e-mail attachment, the CSV in the file server, and the one on Fenix's laptop (from which the file was sent) are exact copies of each other.

Many applications already record some form of activity context, e.g., chat history, browsing history. Such histories provide a rich source of additional metadata. Other activity context, specifically the relationship between objects, such as the fact that a particular file was saved to a local storage device from an email message, requires more pervasive monitoring as found in, e.g., whole provenance capture systems [57]. *Indaleko* is agnostic about the precise data that comprises activity context, but allows for storing and accessing activity context as metadata.

4.4.2 Creating the Excel file

Aki opens the comma separated value (CSV) file using Excel and stores it as a spread sheet. This creates a new object. The Activity Monitor detects that the newly created spreadsheet is a conversion from the CSV file, either via a notification from *Indaleko*-aware Excel or by monitoring the system calls executed on the local system. Aki proceeds to modify the data by filtering it in Excel and saving the changes. The Activity Monitor records this event and updates the meta-data of the spreadsheet to record the derivation-relationship. Ideally a *Indaleko*-aware version of Excel specifies to the Activity Monitor the exact type of the relationship (in this case a derivation); otherwise the Activity Monitor informs the Metadata Server about an unspecified data relationship by observing the opening of a CSV file and a subsequent creation of the Excel file.

Aki proceeds to upload the new Excel file on Slack, which triggers the creation of a new storage object as Slack creates a local copy, the addition of new metadata to Metadata Server via the AS, and the addition of a *copy* data relationship between the original Excel file and the Slack's copy. The Activity Monitor notices (by monitoring Slack chat) that the file was shared with user Fenix and promptly notifies the Metadata Server, which adds this detail to its metadata.

Once Aki is done, its local Metadata Server has been updated with three new objects: the CSV file, the corresponding Excel file, and Slack's copy of the Excel file. There is a data relationship linking all three and metadata informing us that the original CSV came from Fenix and that the final Excel file was also shared with that same person. If Aki wanted to remember what happened to the data from the original CSV from Fenix, they could query their local personal Namespace Server, which would track down this history by querying the Metadata Server metadata.

4.4.3 Sharing the spreadsheet

Fenix receives the Excel file from Aki via Slack on their phone, a sequence of metadata events similar to those described earlier takes place, except the phone does not run a local Namespace Server or Metadata Server. Fenix now uploads the file to the company's cloud drive (4). The Metadata Server (by way of the Attribute Service) reflects the creation of a new object and records its remote location. The use of a company-wide namespace and metadata service enables *Indaleko* to record that the file in the cloud drive is, in fact, a copy of the one received via Slack. Further, Fenix informs their personal Namespace Server that they wish to notify Aki about all updates to the file on the cloud drive. Thus, whenever an Attribute Service sends updated attributes to the Metadata Server, Fenix receives a notification.

The sharing relationship between the personal Namespace Server of Aki and Fenix, and the exchange of the relevant cryptographic credentials, would have been set up earlier.

4.4.4 Data origin and delete requests

When the compliance officer asks about the origin of the data, Fenix can query the corporate Namespace Server to obtain the complete history of the report. This includes the spreadsheet from which the report was derived and the e-mail or Slack messages that transmitted the files.

The corporate Namespace Server was configured to be aware of the locations of the collaborating users' personal Namespace Server. Moreover, because of the activity contexts captured by the Activity Monitor, *Indaleko* is able to identify documents that were created during any activity involving the customer whose data must be deleted. Starting from these documents, and by using the relationship of documents, Dagon was able to find all relevant objects and delete them, including the e-mail and Slack messages.

Aki would have configured their personal Namespace Server to allow sharing of the metadata associated with Fenix with their corporate Namespace Server, and Fenix would configure their personal Namespace Server similarly. As a result, when Fenix issues to the corporate Namespace Server a query asking to trace the origins of the data in the final report, the corporate Namespace Server is able to return all the history tracing back to the original CSV file.

Note that unlike existing systems, *Indaleko* is able to efficiently find related objects across storage silos. Operating systems already provide users with indexing services to accelerate search of local files. This search can be made cross-silo by mounting and enabling indexing on network shares (e.g., Windows Desktop Search), or by interfacing with specific applications such as e-mail (e.g., MacOS Spotlight, or Android search). The problems with indexing a large remote storage repository are resource limitations such as bandwidth. In contrast, *Indaleko* addresses these limitations by delegating indexing and storage to one or more services.

Namespace Servers are responsible for providing efficient search functionality. *Indaleko* uses Attribute Services to keep attributes up to date with object modifications. Lastly, *Indaleko* supports coordinated search among one or more local and remote Namespace Server, allowing, for example, a user to search across both their local Namespace Server as well as their employer's Namespace Server.

Chapter 5

Evaluation

For there is nothing lost, that may be found, if sought. —
Edmund Spenser, *Finding the Faerie Queene*, 1590.

An important aspect of supporting my thesis is to evaluate the system that I have proposed in Chapter 4 and ensure the proposed system provides the information necessary to answer my research questions (Chapter 3.)

5.1 Useful Events

Research question 1 asks what constitutes an activity context. Research question 2 asks how to capture this information. Research question 5 asks how applications can leverage activity context. These three questions all rely upon identifying which events are both useful and practical to collect and aggregate. While prior work has identified events that are of interest I expect to find additional potentially useful events to collect. Thus, evaluating the overhead of adding new events to the activity context is useful. Such an evaluation of the overhead associated with adding activity context would include: how difficult is it to add a new activity context provider to the model and how difficult is it to add support for the new activity context in an existing tool.

I suggest these metrics because they reflect upon the performance of the architectural model that I set out in Chapter 4.

The prior work, notably the personal information trace work [18], has publicly available tools that could ease the collection of data. I can then use my implementation against my own architecture to ensure that resource cost of collection demonstrates the low overhead that I expect for collecting such data.

Once I have demonstrated that my own tools implemented against my architecture do not have substantial overhead, I can then look at the complexity of adding additional data collection, using the suggestions in Table 2.2 as well as additional information that I can identify as potentially useful. Identifying such potentially useful activity context data is an area in which I would expect collaboration could be quite beneficial but I am not relying upon such collaboration to conduct my own research.

An important metric in evaluating my architecture will be to consider both the performance cost of adding additional data collection (measured in performance impact) as well as the development effort (measured in code size). Thus, I propose collecting that information while I develop the tools and build extensions to collect additional data.

5.2 Usefuless of Activity Context

Research question 5 asks a critical question underlying my thesis: that *activity context* is itself useful. There is at least one prior work outside the systems field that indicates it is [37] and thus I reasonably expect that I will be able to reproduce their results. It seems logical to consider their evaluation methodology as one way to measure the effectiveness of our activity context driven model. This, however, is not an ideal fit as the personal digital traces work was evaluated against synthetic existing benchmarks. Thus, other prior work that suggests other potential metrics including the time it takes for a user to perform their search and whether or not the search itself was successful (using the *abandonment rate*).

Assuming that activity context is useful, a more traditional systems evaluation seems justified: what is the cost of collecting and disseminating the activity context, what is the time to process queries [58], how difficult is it to add additional activity context providers, and what is the potential added complexity for applications to utilize *activity context*.

Note: While I expect there will be substantial benefit to collaborating with others interested in the human-computer interface (HCI) and information retrieval (IR) potential for using my work, based upon consultation with my supervisors I do not assume that this will be the case. Thus, the possibility of collaboration has the potential to provide considerable impact if my research supports my thesis, my thesis proposal is not dependent upon such collaborative work. Thus, I intend on being sufficiently flexible to take advantage of collaborative opportunities that do arise, but also realistic in completing my own work in order to complete my thesis.

5.3 Backwards Compatibility

Prior work, such as with semantic file systems [14], has been realized by using indexing services. Similarly, personal digital traces have been used to augment indexing services [37], [59]. These works have evaluations for the effectiveness of their solutions. Thus, virtual directory solutions and indexing solutions have prior evaluations that can be leveraged to develop a more extensive evaluation model.

With respect to my own thesis related work, the availability of this type of indexing and/or virtual directory mechanism would be helpful in understanding the costs and performance of my own architecture to ensure that it can adequately meet the needs of my target tool-building community.

5.4 Access to Activity Context

Research question 4 asks how to provide activity context to enable users and application developers to exploit the enhanced rich meta-data of *Indaleko*. Much of this work relates to performing efficient meta-data queries against a potentially large collection of such data. There is a strong body of prior work regarding meta-data queries of both static and dynamic sources [26], [56], [60]–[70]. The prior work includes a model for providing evaluation. In addition, file system meta-data query specific research has also created a framework for evaluation that relates to the performance speed of meta-data queries [58].

In addition to the performance of such queries, I expect it will also be useful to determine the generality and ability to form specific queries. My expectation is that these queries will not ordinarily be initiated directly by users but it may be useful, as part of evaluating the interface, to determine if human-provided queries are viable and the level of ease with which they can be constructed. I expect further refinement on how to evaluate the flexibility of the query mechanism may be avoided by adopting an existing query language [71], [72].

5.5 Privacy

Research question 6 asks about how to ensure the privacy of users is preserved when capturing detailed personal information in the *activity context* I propose recording. Because I have explicitly stated that for my own thesis work I will be assuming the user maintains complete control of their activity context, I do not propose any specific model for evaluating this security because it is *by design* as secure as the user's own data.

Ensuring privacy of meta-data is an active research area, in terms of extraction, dissemination, and sharing [73]–[75]. Thus, future work that is not envisioned as part of my thesis should be done in a context where emerging work is used to evaluate privacy concerns of a more general meta-data service.

Chapter 6

Conclusion

> *Collecting, storing, and disseminating information about system activity ("activity context") enables the use of Information Retrieval (IR) and Human Computer Interface (HCI) research to build powerful tools to improve human* finding *of pertinent digital data.*

My proposal sets out a plan to mine existing information that is available for collection on our own computer systems in order to solve the myriad obstacles that modern storage systems present to human users when attempting to find specific content.

I will achieve this by creating a flexible systems infrastructure for capturing a broad range of information about the user's interactions with the wider world in which the user operates. This will then allow that information to be used to facilitate the core user objective of *finding*, regardless of device, storage location, or name.

While there is strong evidence for my thesis, it remains to be proven. Using the system that I develop, I will be able to defend my thesis and improve the usefulness of computer systems to human users.

Bibliography

[1] V. Bush *et al.*, "As we may think," *The atlantic monthly*, vol. 176, no. 1, pp. 101–108, 1945 (cit. on pp. 1, 13).

[2] P. J. Guo and M. Seltzer, "Burrito: Wrapping your lab notebook in computational infrastructure," in *TaPP'12 Proceedings of the 4th USENIX conference on Theory and Practice of Provenance*, 2012, pp. 7–7 (cit. on pp. 2, 10, 30).

[3] J. C. Mogul, "Representing information about files," Ph.D. dissertation, Stanford University, 1986 (cit. on pp. 3, 14).

[4] G. A. I. Barnard and L. Fein, "An information filing and retrieval system for the engineering and management records of a large-scale computer development project," English, *American Documentation (pre-1986)*, vol. 9, no. 3, p. 208, Jul. 1958, Copyright - Copyright Wiley Periodicals Inc. Jul 1958; Last updated - 2010-08-27. [Online]. Available: https://search.proquest.com/docview/195441816 (cit. on p. 3).

[5] R. Daley and P. Neumann, "A general-purpose file system for secondary storage," in *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*, ACM, 1965, pp. 213–229 (cit. on pp. 3, 4).

[6] D. M. Ritchie and K. Thompson, "The unix time-sharing system," in *ACM SIGOPS Operating Systems Review*, ACM, vol. 7, 1973, p. 27 (cit. on p. 3).

[7] J. H. Saltzer, "Naming and binding of objects," in *Operating Systems: An Advanced Course*, R. Bayer, R. M. Graham, and G. Seegmüller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 99–208, ISBN: 978-3-540-35880-0. DOI: 10.1007/3-540-08755-9_4. [Online]. Available: https://doi.org/10.1007/3-540-08755-9_4 (cit. on pp. 3, 14).

[8] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the sun network filesystem," in *Proceedings of the Summer USENIX conference*, 1985, pp. 119–130 (cit. on pp. 4, 9).

[9] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West, "Scale and performance in a distributed file system," *ACM Transactions on Computer Systems*, vol. 6, no. 1, pp. 51–81, 1988 (cit. on pp. 4, 9).

[10] R. Levin and M. D. Schroeder, "Transport of electronic messages through a network," Xerox. Palo Alto Research Center, Tech. Rep., 1979 (cit. on p. 4).

[11] A. Z. Spector, "Performing remote operations efficiently on a local computer network," in *Proceedings of the Eighth ACM Symposium on Operating Systems Principles*, ser. SOSP '81, Pacific Grove, California, USA: Association for Computing Machinery, 1981, pp. 76–77, ISBN: 0897910621. DOI: 10.1145/800216.806594. [Online]. Available: https://doi.org/10.1145/800216.806594 (cit. on p. 4).

[12] A. D. Birrell, R. Levin, M. D. Schroeder, and R. M. Needham, "Grapevine: An exercise in distributed computing," *Communications of The ACM*, vol. 25, no. 4, pp. 260–274, 1982 (cit. on p. 4).

[13] K. J. Vicente, B. C. Hayes, and R. C. Williges, "Assaying and isolating individual differences in searching a hierarchical file system," *Human factors*, vol. 29, no. 3, pp. 349–359, 1987 (cit. on pp. 4, 10).

[14] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, "Semantic file systems," in *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '91, Pacific Grove, California, USA: Association for Computing Machinery, 1991, pp. 16–25, ISBN: 0897914473. DOI: 10.1145/121132.121138. [Online]. Available: https://doi.org/10.1145/121132.121138 (cit. on pp. 4, 14, 18, 33, 41).

[15] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999 (cit. on p. 4).

[16] A. Bosch, T. Bogers, and M. Kunder, "Estimating search engine index size variability: A 9-year longitudinal study," *Scientometrics*, vol. 107, no. 2, pp. 839–856, 2016 (cit. on p. 5).

[17] R. Sandberg, “The sun network file system: Design, implementation and experience,” in *in Proceedings of the Summer 1986 USENIX Technical Conference and Exhibition*, Citeseer, 1986 (cit. on p. 6).

[18] D. Q. de Campos Vianna, “Searching heterogenous personal data,” Ph.D. dissertation, Rutgers University, 2019 (cit. on pp. 6, 17, 39).

[19] J. Chen, H. Guo, W. Wu, and C. Xie, “Search your memory ! - an associative memory based desktop search system,” in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’09, Providence, Rhode Island, USA: Association for Computing Machinery, 2009, pp. 1099–1102, ISBN: 9781605585512. DOI: 10.1145/1559845.1559992. [Online]. Available: https://doi.org/10.1145/1559845.1559992 (cit. on pp. 6, 10).

[20] S. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins, “Stuff i’ve seen: A system for personal information retrieval and re-use,” in *ACM SIGIR Forum*, ACM, vol. 49, 2016, pp. 28–35 (cit. on p. 6).

[21] P. Chakriswaran, D. R. Vincent, K. Srinivasan, V. Sharma, C.-Y. Chang, and D. G. Reina, “Emotion ai-driven sentiment analysis: A survey, future research directions, and open issues,” *Applied Sciences*, vol. 9, no. 24, p. 5462, 2019 (cit. on p. 7).

[22] Y. Li, Y. Jiang, D. Tian, L. Hu, H. Lu, and Z. Yuan, “Ai-enabled emotion communication,” *IEEE Network*, vol. 33, no. 6, pp. 15–21, 2019. DOI: 10.1109/MNET.001.1900070 (cit. on p. 7).

[23] D. M. Ritchie and K. Thompson, “The unix time-sharing system,” *Commun. ACM*, vol. 17, no. 7, pp. 365–375, Jul. 1974, ISSN: 0001-0782. DOI: 10.1145/361011.361061. [Online]. Available: https://doi.org/10.1145/361011.361061 (cit. on p. 9).

[24] I. Digital and Xerox, “The ethernet, a local area network. data link layer and physical layer specifications,” 1980. [Online]. Available: http://decnet.ipv7.net/docs/dundas/aa-k759b-tk.pdf (cit. on p. 9).

[25] A. N. Bessani, R. Mendes, T. Oliveira, N. F. Neves, M. Correia, M. Pasin, and P. Verissimo, “Scfs: A shared cloud-backed file system.,” in *USENIX Annual Technical Conference*, Citeseer, 2014, pp. 169–180 (cit. on p. 9).

[26] C. Wang, K. Thareja, M. Stealey, P. Ruth, and I. Baldin, “Comet: A distributed metadata service for federated cloud infrastructures,” in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–7. DOI: 10.1109/HPEC.2019.8916536 (cit. on pp. 9, 10, 33, 41).

[27] K. J. Vicente and R. C. Williges, “Accommodating individual differences in searching a hierarchical file system,” *International Journal of Human-computer Studies International Journal of Man-machine Studies*, vol. 29, no. 6, pp. 647–668, 1988 (cit. on p. 10).

[28] J. Benet, “Ipfs - content addressed, versioned, p2p file system.,” *arXiv preprint arXiv:1407.3561*, 2014 (cit. on p. 10).

[29] D. Bermbach, M. Klems, S. Tai, and M. Menzel, “Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs,” in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 452–459. DOI: 10.1109/CLOUD.2011.62 (cit. on p. 10).

[30] S. Sivasubramanian, “Amazon dynamodb: A seamlessly scalable non-relational database service,” in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’12, Scottsdale, Arizona, USA: Association for Computing Machinery, 2012, pp. 729–730, ISBN: 9781450312479. DOI: 10.1145/2213836.2213945. [Online]. Available: https://doi.org/10.1145/2213836.2213945 (cit. on p. 10).

[31] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, “Farsite: Federated, available, and reliable storage for an incompletely trusted environment,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 1–14, Dec. 2003, ISSN: 0163-5980. DOI: 10.1145/844128.844130. [Online]. Available: https://doi.org/10.1145/844128.844130 (cit. on p. 10).

[32] P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton, “Extending document management systems with user-specific active properties,” *ACM Trans. Inf. Syst.*, vol. 18, no. 2, pp. 140–170, Apr. 2000, ISSN: 1046-8188. DOI: 10.1145/348751.348758. [Online]. Available: https://doi.org/10.1145/348751.348758 (cit. on p. 10).

[33] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Noble, "Using provenance to aid in personal file search," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ser. ATC'07, Santa Clara, CA: USENIX Association, 2007 (cit. on pp. 10, 17).

[34] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, "Automatic database management system tuning through large-scale machine learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17, Chicago, Illinois, USA: Association for Computing Machinery, 2017, pp. 1009–1024, ISBN: 9781450341974. DOI: 10.1145/3035918.3064029. [Online]. Available: https://doi.org/10.1145/3035918.3064029 (cit. on p. 10).

[35] K. Briney, *Data Management for Researchers: Organize, maintain and share your data for research success.* 2015 (cit. on p. 14).

[36] D. Vianna, A.-M. Yong, C. Xia, A. Marian, and T. Nguyen, "A tool for personal data extraction," in *2014 IEEE 30th International Conference on Data Engineering Workshops*, 2014, pp. 80–83 (cit. on pp. 17, 23).

[37] D. Vianna, V. Kalokyri, A. Borgida, T. D. Nguyen, and A. Marian, "Searching heterogeneous personal digital traces," in *Proceedings of the Association for Information Science and Technology*, vol. 56, 2019, pp. 276–285 (cit. on pp. 16–18, 20, 40, 41).

[38] J. Kim and W. B. Croft, "Retrieval experiments using pseudo-desktop collections," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 1297–1306 (cit. on p. 17).

[39] C. A. Soules and G. R. Ganger, "Connections: Using context to enhance file search," in *Proceedings of the twentieth ACM symposium on operating systems principles*, 2005, pp. 119–132 (cit. on p. 17).

[40] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, *et al.*, "Ground: A data context service.," in *CIDR*, Citeseer, 2017 (cit. on pp. 17, 19).

[41] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong, "Mylifebits: Fulfilling the memex vision," in *Proceedings of the tenth ACM international conference on Multimedia*, 2002, pp. 235–238 (cit. on pp. 17, 19).

[42] V. Kalokyri, A. Borgida, A. Marian, and D. Vianna, "Integration and exploration of connected personal digital traces," in *Proceedings of the ExploreDB'17*, 2017, p. 3 (cit. on p. 17).

[43] O. Bergman, T. Israeli, and S. Whittaker, "Search is the future? the young search less for files," in *Proceedings of the Association for Information Science and Technology*, vol. 56, 2019, pp. 360–363 (cit. on pp. 16, 18).

[44] C. M. Brown, *Human-computer interface design guidelines.* Intellect Books, 1999 (cit. on p. 18).

[45] O. Bergman, T. Israeli, and S. Whittaker, "Factors hindering shared files retrieval," *Aslib Journal of Information Management*, vol. 72, no. 1, pp. 130–147, 2019 (cit. on p. 18).

[46] H. Ritzdorf, N. Karapanos, and S. Čapkun, "Assisted deletion of related content," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014, pp. 206–215 (cit. on p. 18).

[47] F. Vitale, "Personal data curation in the cloud age : Individual differences and design opportunities," Ph.D. dissertation, University of British Columbia, 2020. DOI: http://dx.doi.org/10.14288/1.0392427. [Online]. Available: https://open.library.ubc.ca/collections/ubctheses/24/items/1.0392427 (cit. on p. 18).

[48] C. A. Soules and G. R. Ganger, "Why can't i find my files?: New methods for automating attribute assignment," in *HotOS*, 2003, pp. 115–120 (cit. on p. 18).

[49] S. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins, "Stuff i've seen: A system for personal information retrieval and re-use," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, vol. 49, 2003, pp. 28–35 (cit. on p. 19).

[50] A. Collins, T. Apted, and J. Kay, "Tabletop file system access: Associative and hierarchical approaches," in *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*, IEEE, 2007, pp. 113–120 (cit. on p. 19).

[51] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca, "A data-oriented survey of context models," *ACM Sigmod Record*, vol. 36, no. 4, pp. 19–26, 2007 (cit. on p. 20).

[52] M. Greenberg, “Files-as-filesystems for posix shell data processing,” in *Proceedings of the 11th Workshop on Programming Languages and Operating Systems*, ser. PLOS ’21, Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 17–23, ISBN: 9781450387071. DOI: 10.1145/3477113.3487265. [Online]. Available: https://doi.org/10.1145/3477113.3487265 (cit. on p. 20).

[53] R. Kadekodi, S. Kadekodi, S. Ponnapalli, H. Shirwadkar, G. R. Ganger, A. Kolli, and V. Chidambaram, “Winefs: A hugepage-aware file system for persistent memory that ages gracefully,” in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles CD-ROM*, 2021, pp. 804–818 (cit. on p. 20).

[54] J. Locke, *Locke’s essays: An essay concerning human understanding, and A treatise on the conduct of the understanding (Complete in 1 volume with the author’s last additions and corrections)*. 1844 (cit. on p. 25).

[55] D. Devecsery, M. Chow, X. Dou, J. Flinn, and P. M. Chen, “Eidetic systems,” in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 525–540 (cit. on p. 26).

[56] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian, “Smartstore: A new metadata organization paradigm with semantic-awareness for next-generation file systems,” in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–12. DOI: 10.1145/1654059.1654070 (cit. on pp. 33, 41).

[57] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, “Practical whole-system provenance capture,” in *Symposium on Cloud Computing (SoCC’17)*, ACM, 2017 (cit. on p. 36).

[58] S. Ames, M. Gokhale, and C. Maltzahn, “Qmds: A file system metadata management service supporting a graph data model-based query language,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 28, no. 2, pp. 159–183, 2013 (cit. on pp. 40, 41).

[59] L. Xu, Z. Huang, H. Jiang, L. Tian, and D. Swanson, “VSFS: A searchable distributed file system,” *Proceedings of PDSW 2014: 9th Parallel Data Storage Workshop - Held in Conjunction with SC 2014: The International Conference for High Performance*

Computing, Networking, Storage and Analysis, pp. 25–30, 2014. DOI: 10.1109/PDSW.2014.10 (cit. on p. 41).

[60] T. Strong and P. Akundi, "A Semantic Cloud for File System Annotation," pp. 1–4, (cit. on p. 41).

[61] F. Revol, "Universal file system extended attributes namespace," in *International Conference on Dublin Core and Metadata Applications*, 2011, pp. 69–73 (cit. on p. 41).

[62] J. Liu, D. Feng, Y. Hua, B. Peng, P. Zuo, and Y. Sun, "P-index: An efficient searchable metadata indexing scheme based on data provenance in cold storage," in *International Conference on Algorithms and Architectures for Parallel Processing*, Springer, 2015, pp. 597–611 (cit. on p. 41).

[63] Z. Huo, L. Xiao, Q. Zhong, S. Li, A. Li, L. Ruan, S. Wang, and L. Fu, "Mbfs: A parallel metadata search method based on bloomfilters using mapreduce for large-scale file systems," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 3006–3032, 2016 (cit. on p. 41).

[64] M. Suguna and T. Anand, "Dynamic Metadata Management in Semantic File Systems," vol. 5, no. 3, pp. 44–47, 2015 (cit. on p. 41).

[65] A. Parker-Wood, D. D. E. Long, E. Miller, P. Rigaux, and A. Isaacson, "A File By Any Other Name: Managing File Names with Metadata," *Proceedings of International Conference on Systems and Storage*, pp. 1–11, 2014 (cit. on p. 41).

[66] R. Watson, S. Dekeyser, and N. Albadri, "Exploring the design space of metadata-focused file management systems," in *Proceedings of the Australasian Computer Science Week Multiconference*, ACM, 2017, p. 20 (cit. on p. 41).

[67] A. Leung, I. Adams, and E. L. Miller, "Magellan: A searchable metadata architecture for large-scale file systems," *University of California, Santa Cruz, Tech. Rep. UCSC-SSRC-09-07*, 2009 (cit. on p. 41).

[68] A. W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller, "Spyglass: Fast, scalable metadata search for large-scale storage systems.," in *FAST*, vol. 9, 2009, pp. 153–166 (cit. on p. 41).

[69] S. Niazi, M. Ismail, S. Haridi, J. Dowling, S. Grohsschmiedt, and M. Ronström, "Hopsfs: Scaling hierarchical file system metadata using newsql databases.," in *FAST*, 2017, pp. 89–104 (cit. on p. 41).

[70] R. V. H. Van Staereling, R. Appuswamy, D. C. van Moolenbroek, and A. S. Tanenbaum, “Efficient, modular metadata management with loris,” in *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*, IEEE, 2011, pp. 278–287 (cit. on p. 41).

[71] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, “Cypher: An evolving query language for property graphs,” in *Proceedings of the 2018 International Conference on Management of Data*, ACM, 2018, pp. 1433–1445 (cit. on p. 41).

[72] O. van Rest, S. Hong, J. Kim, X. Meng, and H. Chafi, “Pgql: A property graph query language,” in *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, ACM, 2016, p. 7 (cit. on p. 41).

[73] M. Eviette and A. Simpson, “Towards models for privacy preservation in the face of metadata exploitation,” in *Privacy and Identity Management*, M. Friedewald, S. Schiffner, and S. Krenn, Eds., Cham: Springer International Publishing, 2021, pp. 247–264, ISBN: 978-3-030-72465-8 (cit. on p. 42).

[74] S. Eskandarian, H. Corrigan-Gibbs, M. Zaharia, and D. Boneh, “Express: Lowering the cost of metadata-hiding communication with cryptographic privacy,” in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021, pp. 1775–1792 (cit. on p. 42).

[75] V. I. Budzko, D. Melnikov, V. Korolev, V. G. Belenkov, and P. A. Keyer, “Architecture solutions for the metadata extraction toolkit, taking into account the built-in privacy extracts,” in *CEUR Workshop Proceedings*, 2019, pp. 3–9 (cit. on p. 42).