The Associative File System: First Class File Relationships

Tony Mason (PhD Student)
The University of British Columbia
fsgeek@cs.ubc.ca

Margo Seltzer
The University of British Columbia
mseltzer@cs.ubc.ca

Abstract

File systems have evolved tremendously over the past six decades. However, the *storage management* of file systems has evolved but not the name space.

The hierarchical file system name space was proposed as part of Multics in 1965 and is fundamentally what is used today. However, storage capacity has exploded. The vast majority of this is stored in general purpose file systems.

The namespace's lack of evolution makes that data difficult to access. The systems community has failed to expand the capabilities of file systems to enable the development of viable alternatives to an ever-growing pool of files.

Focusing on file relationships permits us to enhance the system by exploiting those relationships to create communities of interest much like social networks do. I realize this by constructing a graph file system as an alternative to the traditional hierarchical file system.

1 INTRODUCTION

The hierarchical model was introduced in Multics [1, 2]. At the same time they introduced *links*, a tacit admission of the model's weakness.

Almost all systems that followed Multics adopted the hierarchical namespace, e.g., AT&T [4] and Digital [5]. Early UNIX work also suggests model limitations; they describe the importance of the *permuted index server*. [4].

Since then, storage capacity has exploded — we expect to have 163 zettabytes (ZB) of data in 2025 [3] — but the hierarchical name space has not evolved. This has led to numerous "work-arounds", including vast collections of similarly named files, limiting directory sizes, hard links, and soft links to enhance findability, as well as building specialized applications to capture cross-file relationships, treating the hierarchical file system as an ill-designed key-value store.

To address this we propose file system that supports relationships as a fundamental feature. We *already* manage a tiny set of relationships, e.g., the *contains* relationship. This evolves the namespace from a *tree* to a *graph*.

2 ASSOCIATIVE FILE SYSTEM

Other approaches *other than* a new file system and do not address our specific concerns. Thus, we propose the development of the *associative file system*. It will provide support for an extensible set of relationships. While we will be using some ideas from graph databases, we will adhere to the view that a file system is a general mechanism for storing arbitrary files but extend it to also support the ability to store additional relationships.

Relationships are between two files and are either unidirectional or bi-directional. Examples include: *similar* — similarity between two files, *precedes* and *succeeds* for *versioning*, and *contains* for the classic directory.

In addition to basic file system functions such as *create*, the new file system will also support operations to manage the relationships and labels that are part of my model, such as *relate*, *set*, and *label*.

This model is simple, yet powerful. It enables versioning, provenance, and application specfic relationships. Further, we can create clusters of related files by querying for all vertices having a specific relationship.

Relationships can be created from a variety of sources, including the system, tools that extract meta-data, applications, users, and operating system extensions.

3 CONCLUSION

The associative file system is my attempt to address these concerns in a novel way. While prior work in this area has looked at adapting databases to the problem, we choose to ask how to evolve the file system to support relationships. By elevating arbitrary and generalized relationships as first class file system elements, we can provide a better user experience.

Some will avoid this work, because it touches upon the unpredictable and messy area of human users. However, it is time to realize that our systems are no longer our personal playground, but a critical service to humanity.

REFERENCES

[1] Corbató, F. J., and Vyssotsky, V. A. Introduction and overview of the multics system. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I* (1965), ACM, pp. 185–196.

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

[3] Reinsel, D., Gantz, J., and Rydning, J. Data age 2025: The digitization of the world from edge to core. Tech. rep., Seagate, November 2018. (accessed January 6, 2019).

[4] Ritchie, D. M., and Thompson, K. The unix time-sharing system. In *ACM SIGOPS Operating Systems Review* (1973), vol. 7, ACM, p. 27.

[5] Spier, M. J., Hastings, T. N., and Cutler, D. N. An experimental implementation of the kernel/domain architecture. In *Proceedings of the Fourth ACM Symposium on Operating System Principles* (New York, NY, USA, 1973), SOSP '73, ACM, pp. 8–21.