Richer File System Metadata Using Links and Attributes

Alexander Ames, Nikhil Bobb, Scott A. Brandt, Adam Hiatt, Carlos Maltzahn, Ethan L. Miller, Alisa Neeman, Deepa Tuteja

Computer Science Department
Storage **S**ystems **R**esearch **C**enter
University of California, Santa Cruz

Motivation

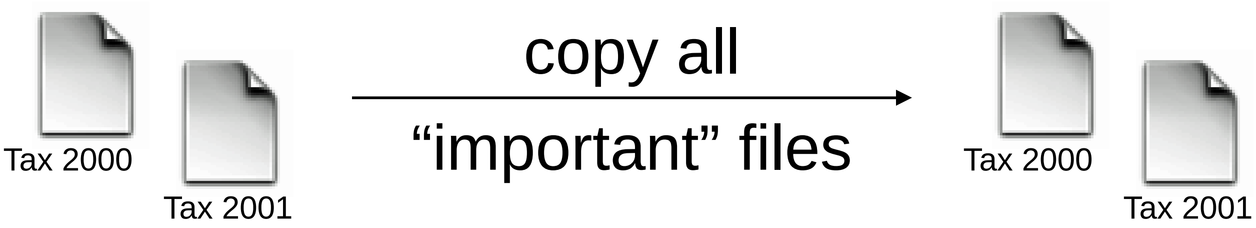
- The Problem:
  - File systems don’t keep up with today’s information management needs
- So what?
  - Massive data loss due to obsolescence
  - Easier to find stuff on the web than in home directories
  - New opportunity: storage-class memory technologies
- Our Approach: the **L**ink**i**ng **F**ile **S**ystem (LiFS)
  - Store file context in file system metadata
  - Use relational links, attributes, and triggers
  - Use storage-class memory to store metadata (e.g. MRAM)

Driving Problem: Obsolescence

Example: Using Software to do taxes

Tax Year	Platform	Comment
2000	PC	
2001	PC	
		Switch: copy all “important” files
2002	Mac	
2003	Mac	Got rid of PC
2004	Mac	Need to amend 2001: Mac Tax SW does not read PC files. Need PC?

Driving Problem: Obsolescence



Tax 2000

Tax 2001

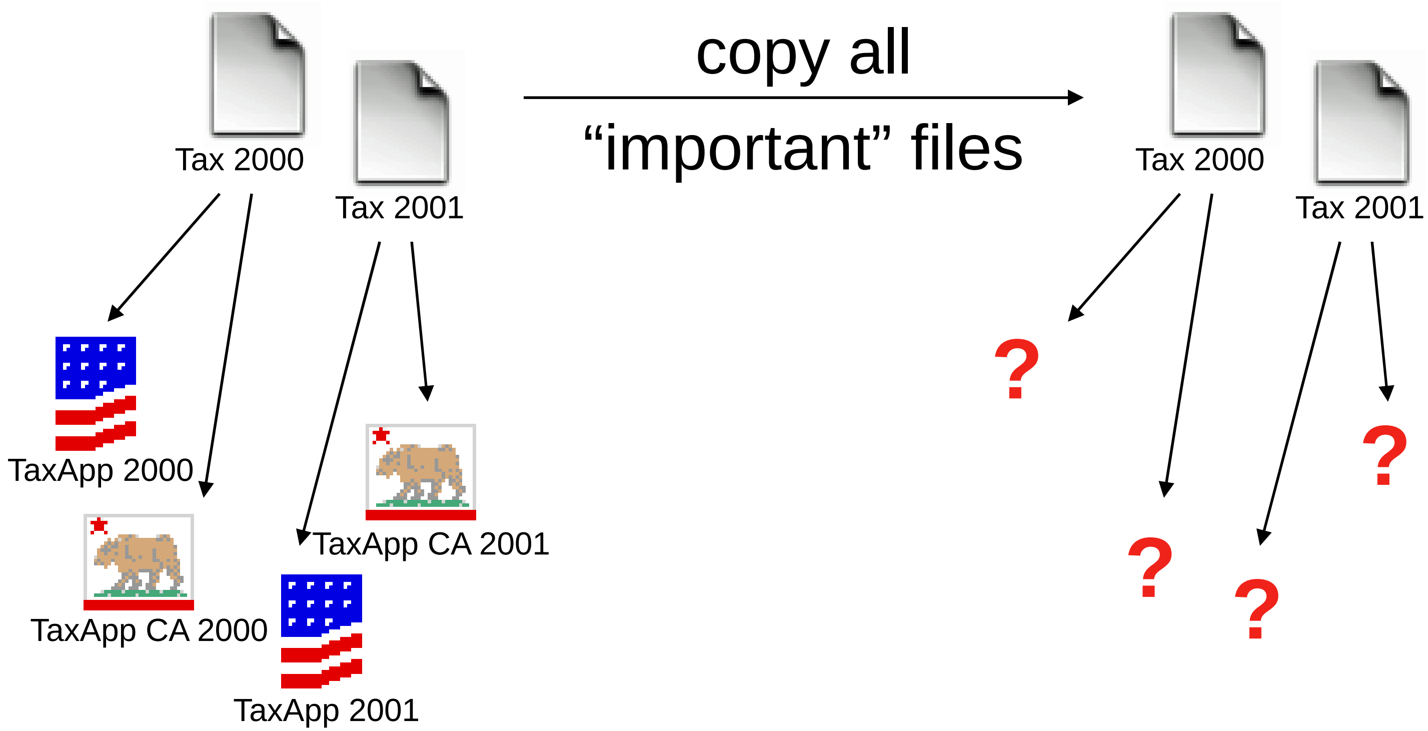
copy all

“important” files

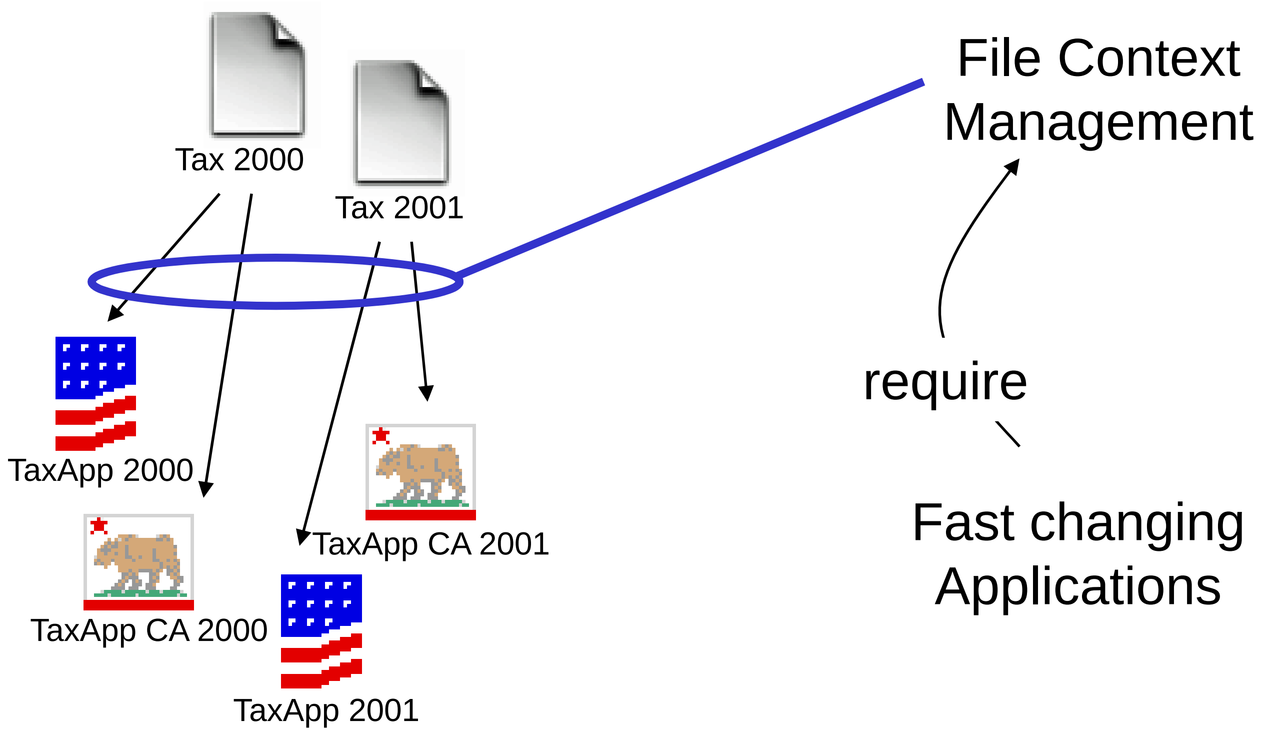
Tax 2000

Tax 2001

Driving Problem: Obsolescence



Driving Problem: Obsolescence

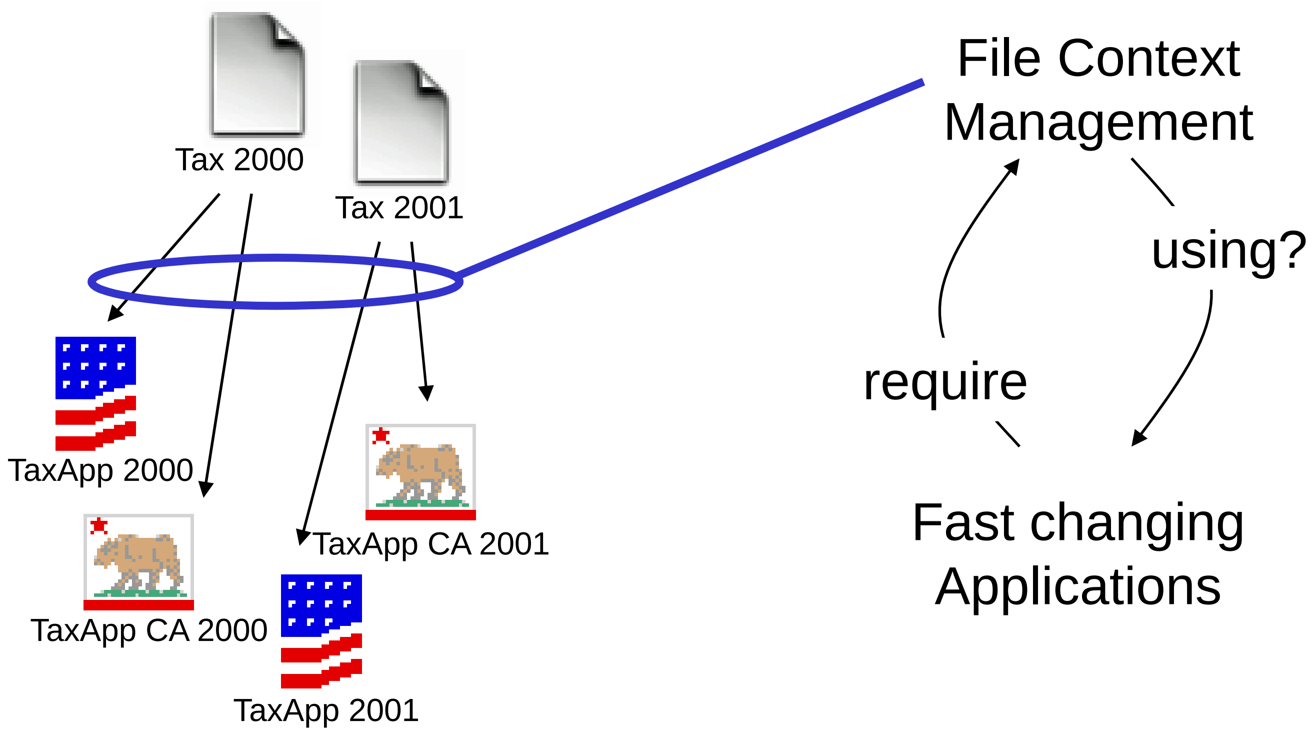


File Context Management

require

Fast changing Applications

Driving Problem: Obsolescence



Tax 2000

Tax 2001

TaxApp 2000

TaxApp CA 2000

TaxApp CA 2001

TaxApp 2001

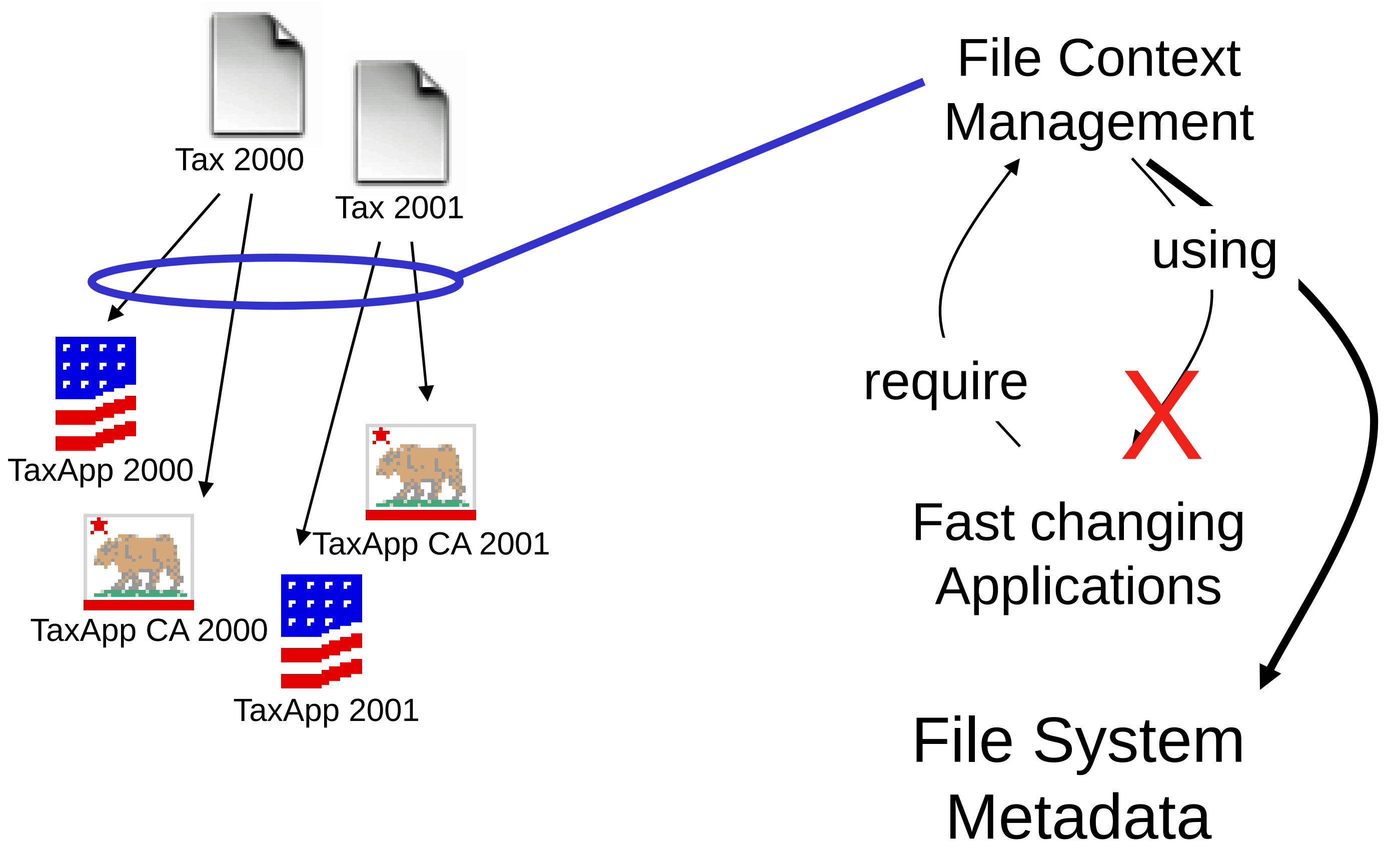
File Context Management

using?

Fast changing Applications

require

Driving Problem: Obsolescence



Tax 2000

Tax 2001

TaxApp 2000

TaxApp CA 2000

TaxApp CA 2001

TaxApp 2001

File Context Management

using

require

X

Fast changing Applications

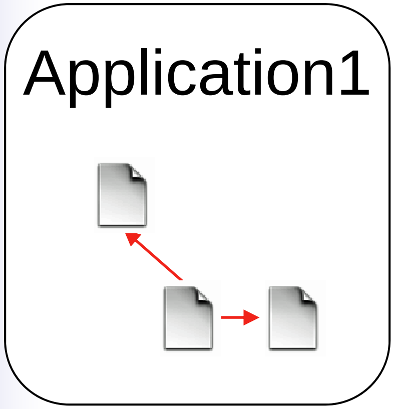
File System Metadata

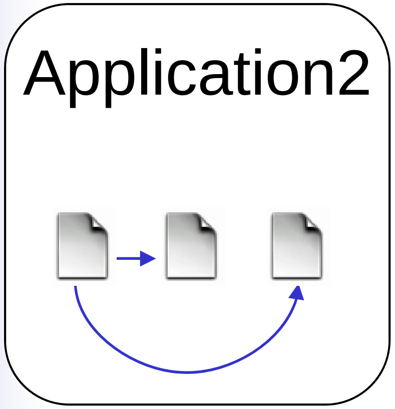
Driving Problem: Finding Stuff

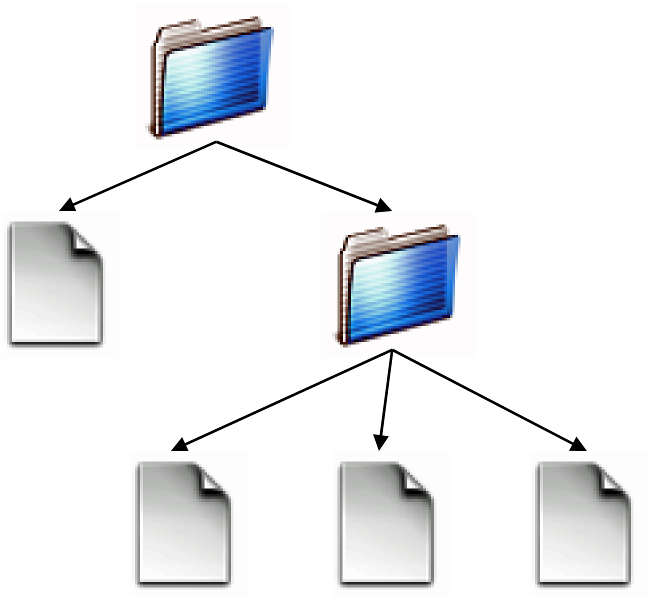
- It's easier to find stuff on the Web than in a home directory!
  - Lots of relationships on the Web: Google's PageRank
  - Few explicit relationships between files: weak ranking

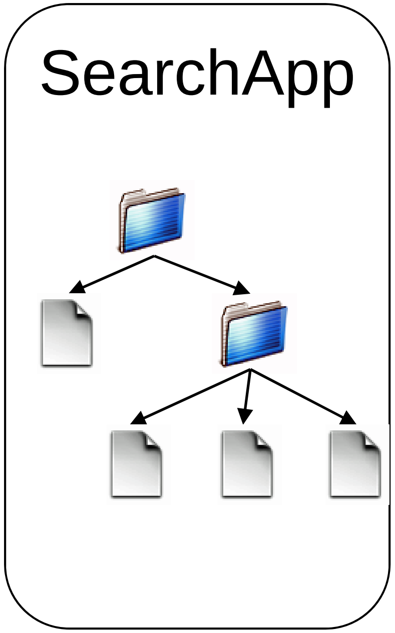
- Observations:
  - Lots of implicit relationships between files
    - Provenance, dependencies, contexts
  - Applications maintain own relations between files
    - make, email, Management of Photos, MP3, etc

Driving Problem: Finding Stuff



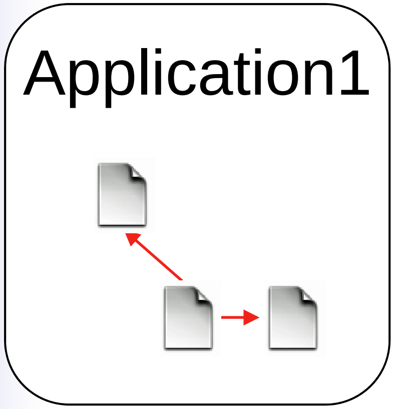


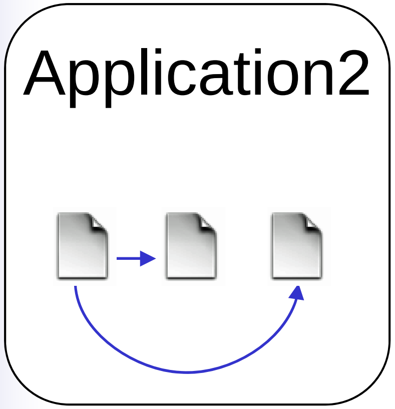


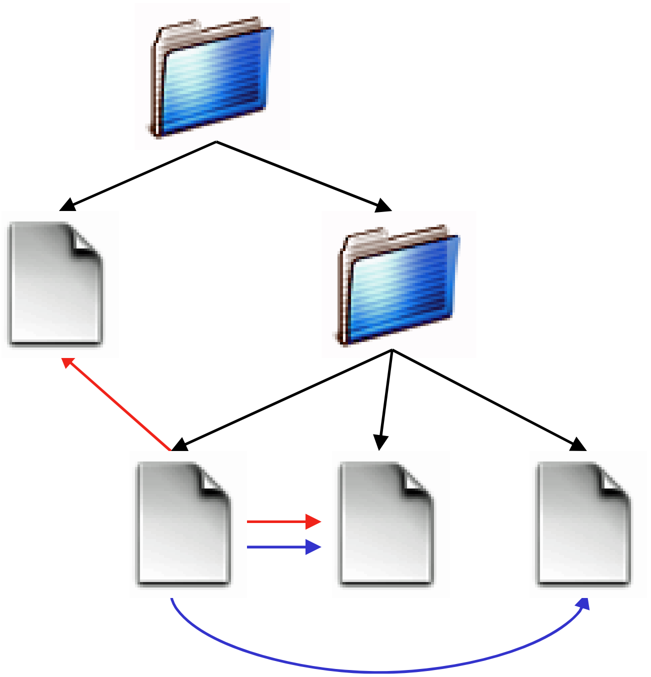


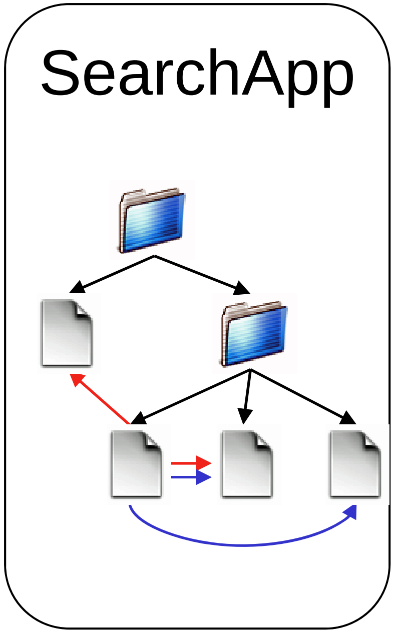
So far:
Many Apps maintain their own metadata

Driving Problem: Finding Stuff









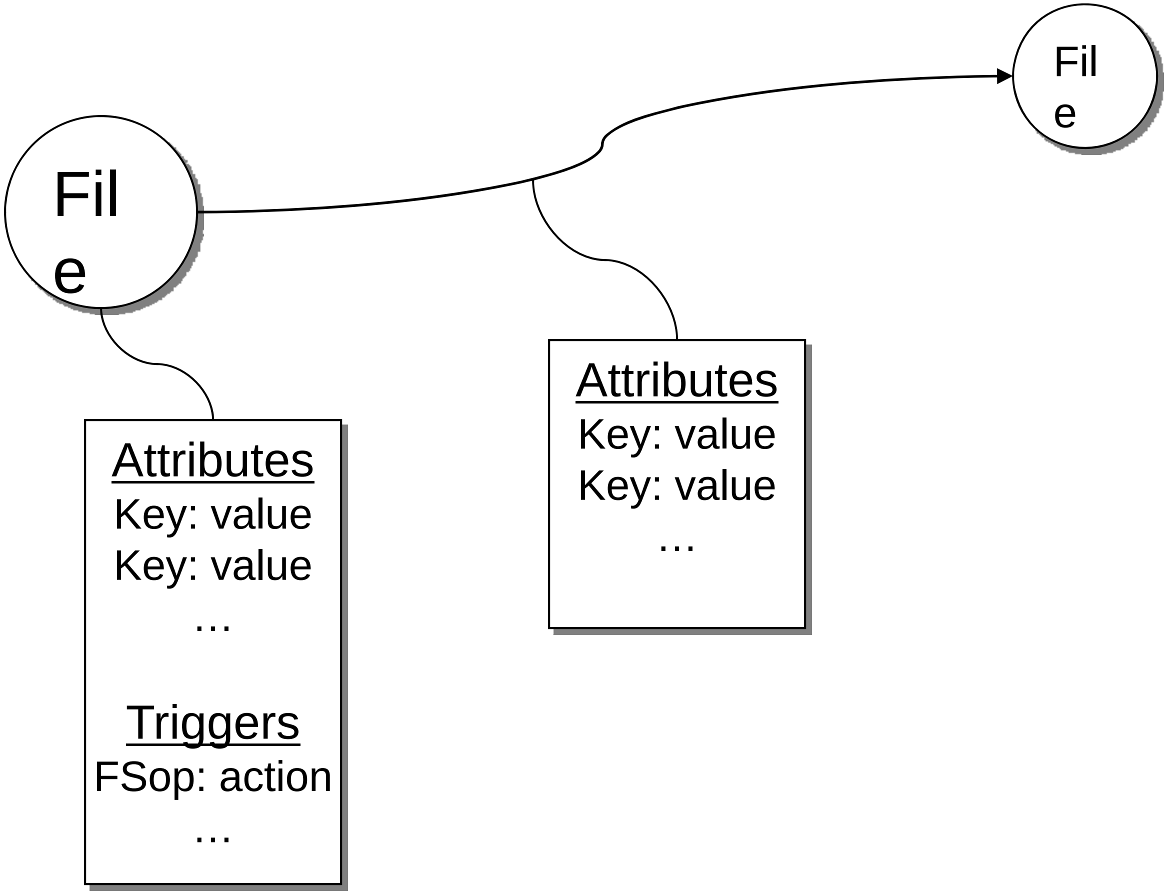
LiFS:

- Uniform metadata store
- “Local” Semantic Web

File Context is Surprisingly Useful

- Infrastructure dependencies
- Search
- Views
- Provenance and history
- Workflow
- …

LiFS Key Concepts: Links and Attributes

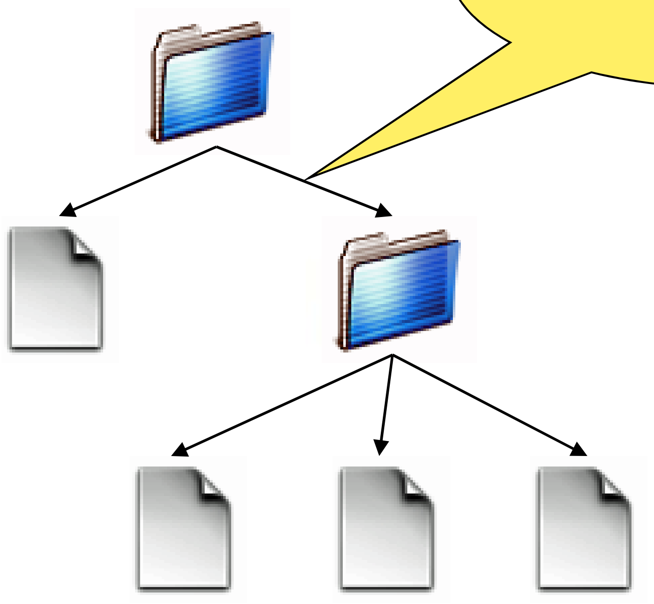


<u>Attributes</u>
Key: value
Key: value
…

<u>Triggers</u>
FSop: action
…

<u>Attributes</u>
Key: value
Key: value
…

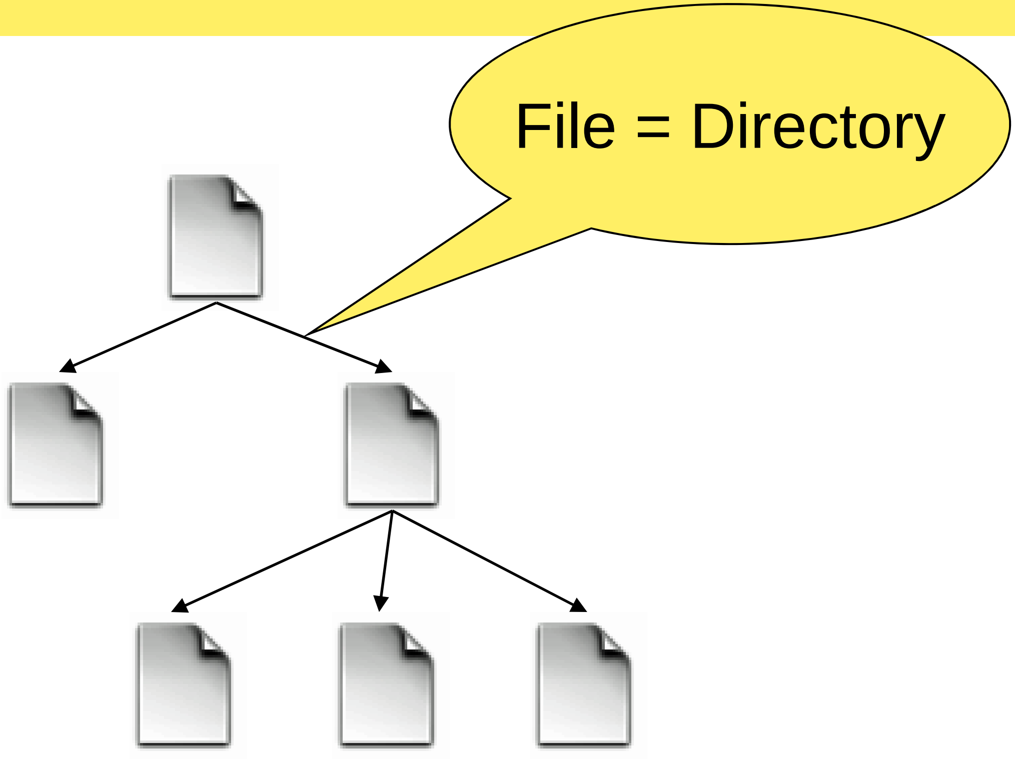
Links in Traditional File Systems



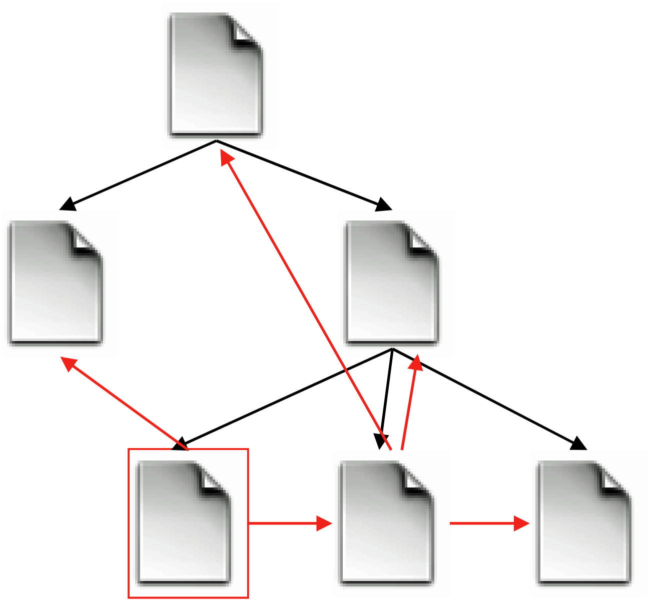
Containment Links

Important for POSIX compatibility

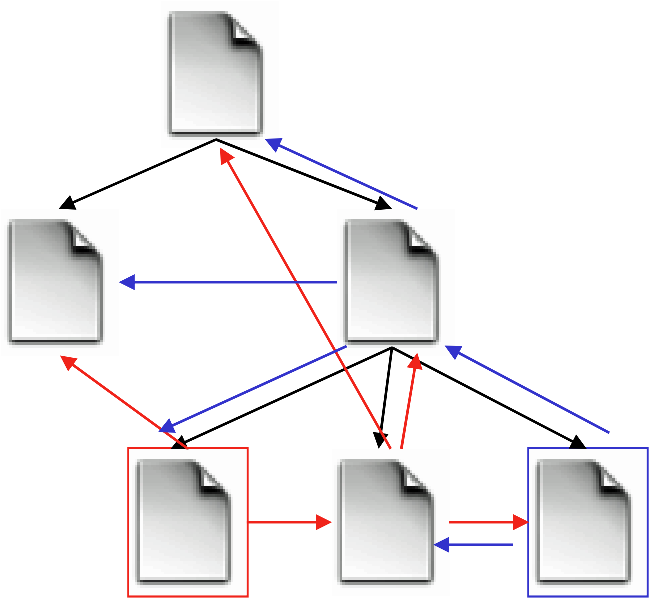
Relational Links



Relational Links: Multiple Views



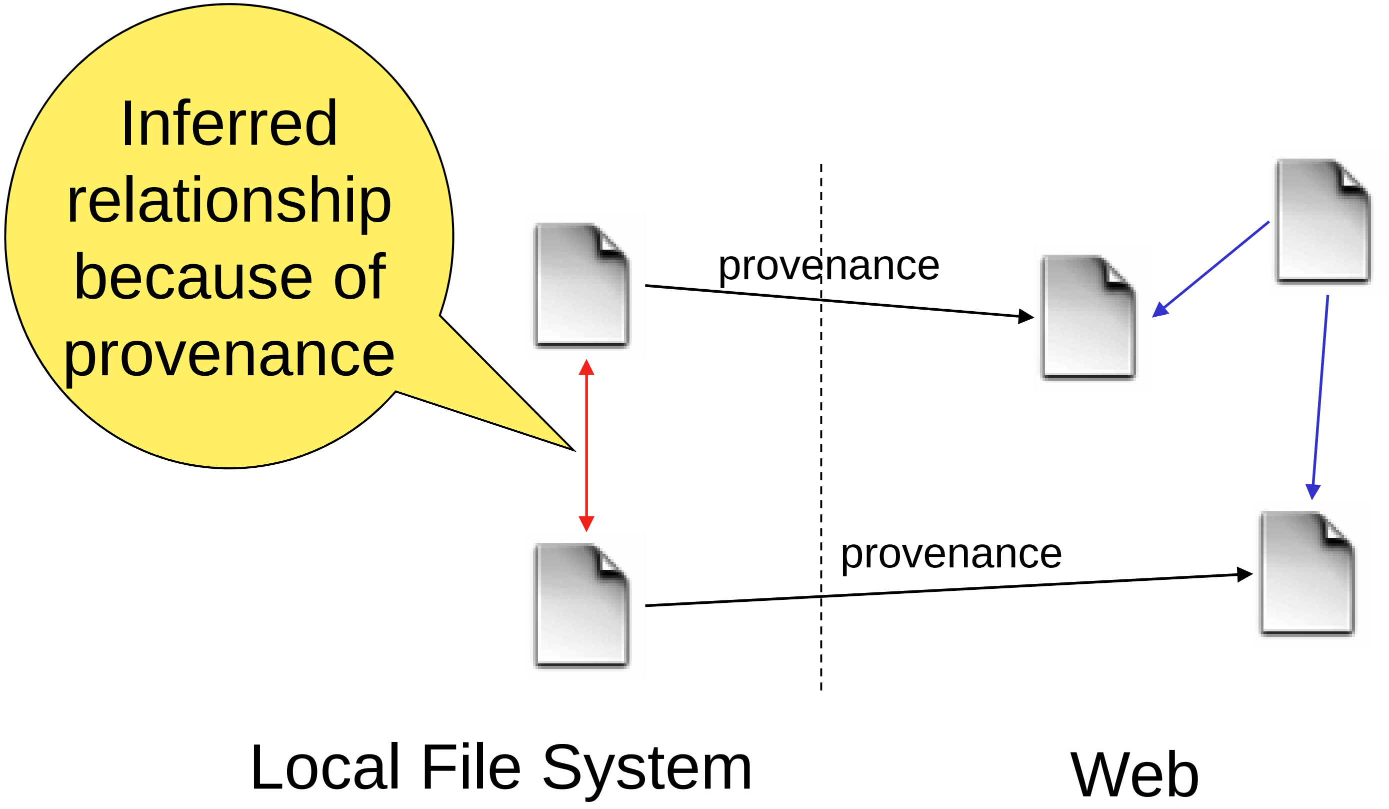
Relational Links: Multiple Views



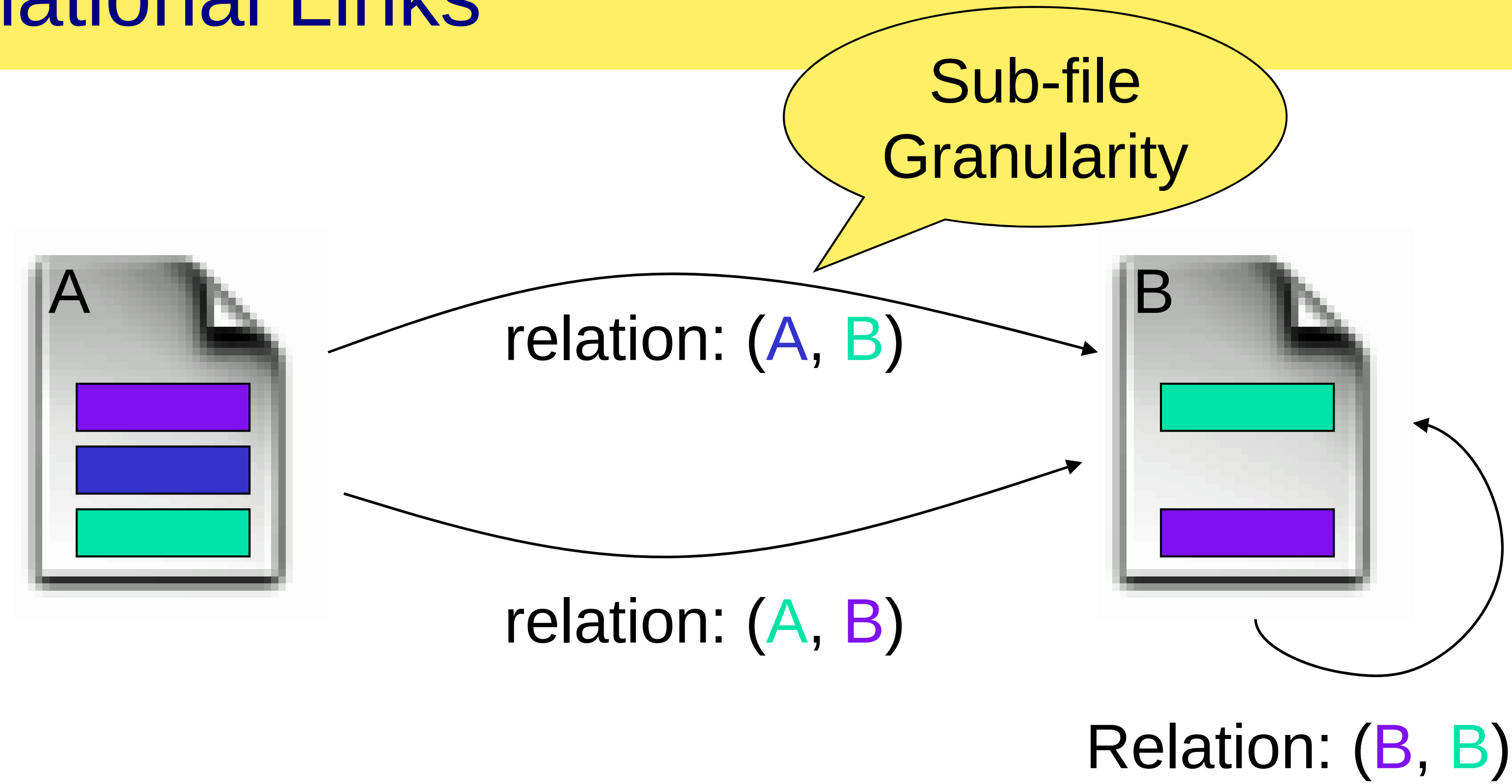
Examples:

- Multiple Concurrent Libraries
- Personal Information Management

Relational Links: Provenance (Web)



Relational Links



Sub-file Granularity

relation: (A, B)

relation: (A, B)

Relation: (B, B)

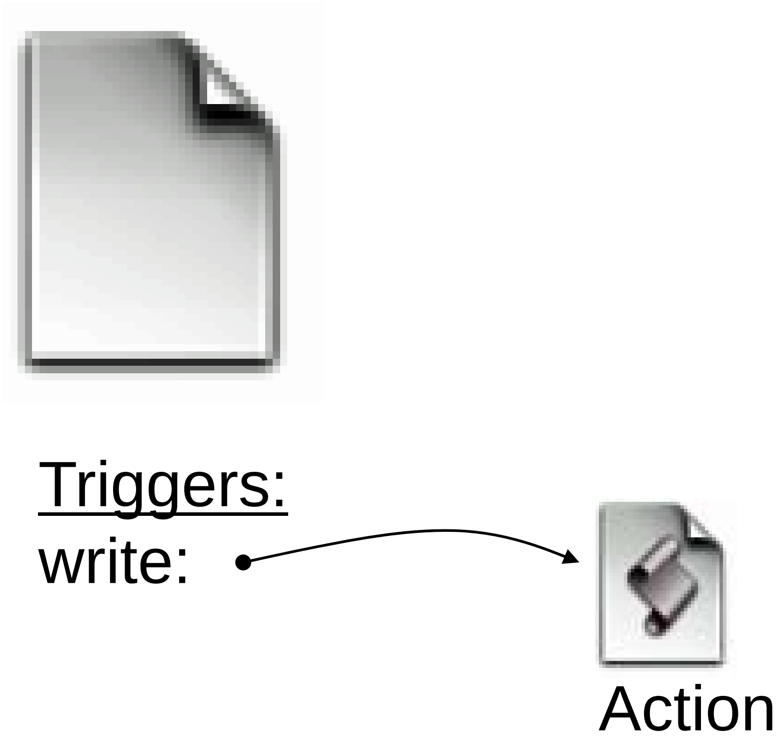
Example Relationships:

- Calendar entries to inbox file items
- Call graphs in a software project

File Triggers

Attribute: key = value

Trigger: fs/op = action

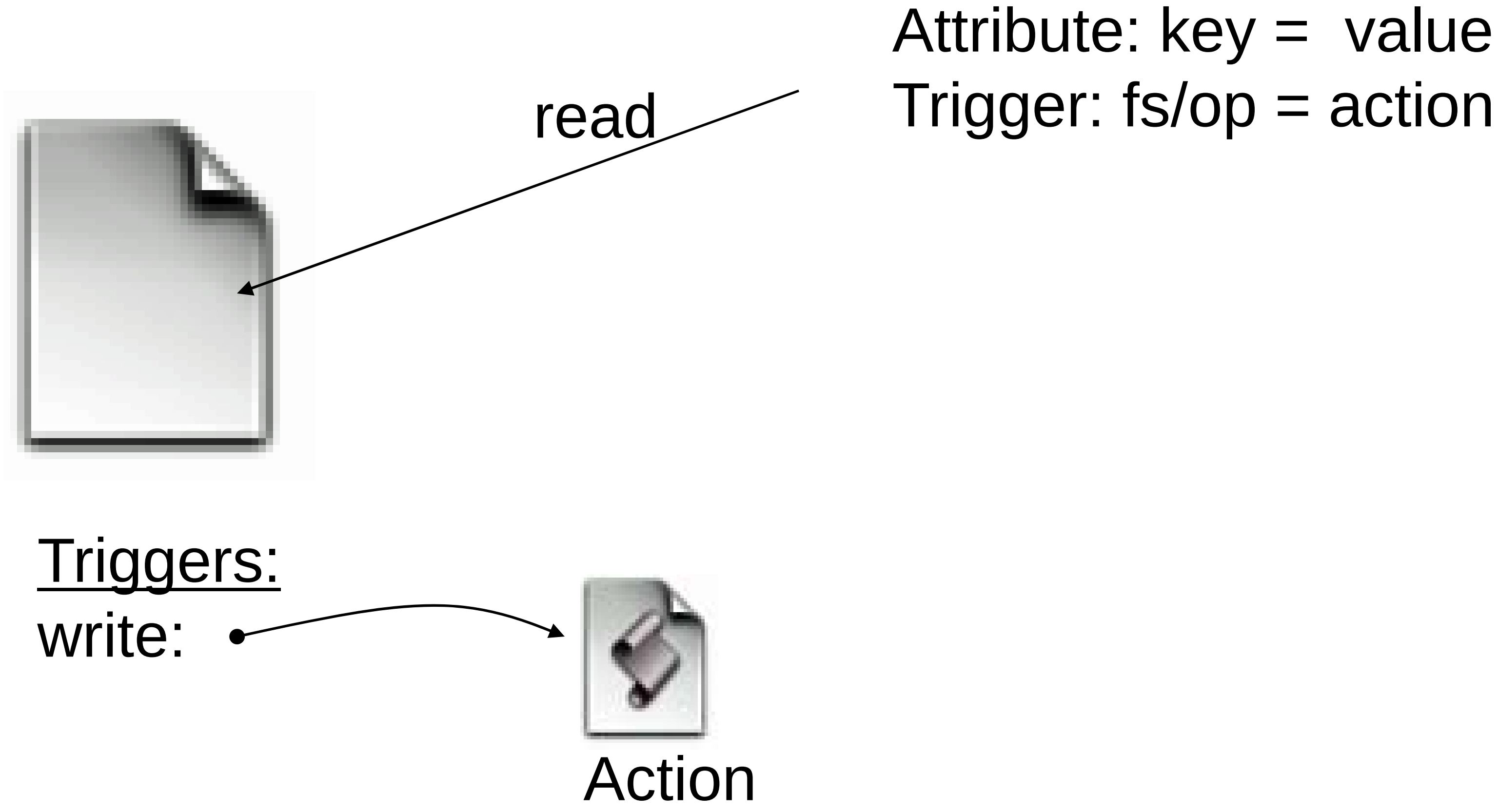


Triggers:

write:

Action

File Triggers



Attribute: key = value

Trigger: fs/op = action

read

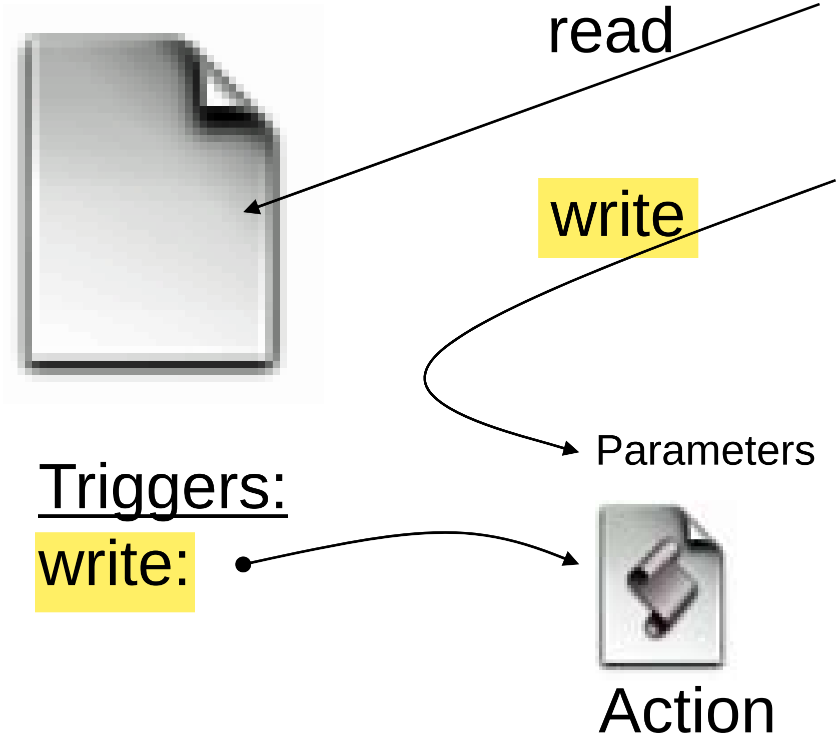
<u>Triggers:</u>

write:

Action

File Triggers

Attribute: key = value
Trigger: fs/op = action



read

write

Parameters

Triggers:
write:

Action

Action decides what to do with write

File Triggers: Examples

- Workflows
  - Notification upon checking in source code
  - Thumbnailing upon uploading photos

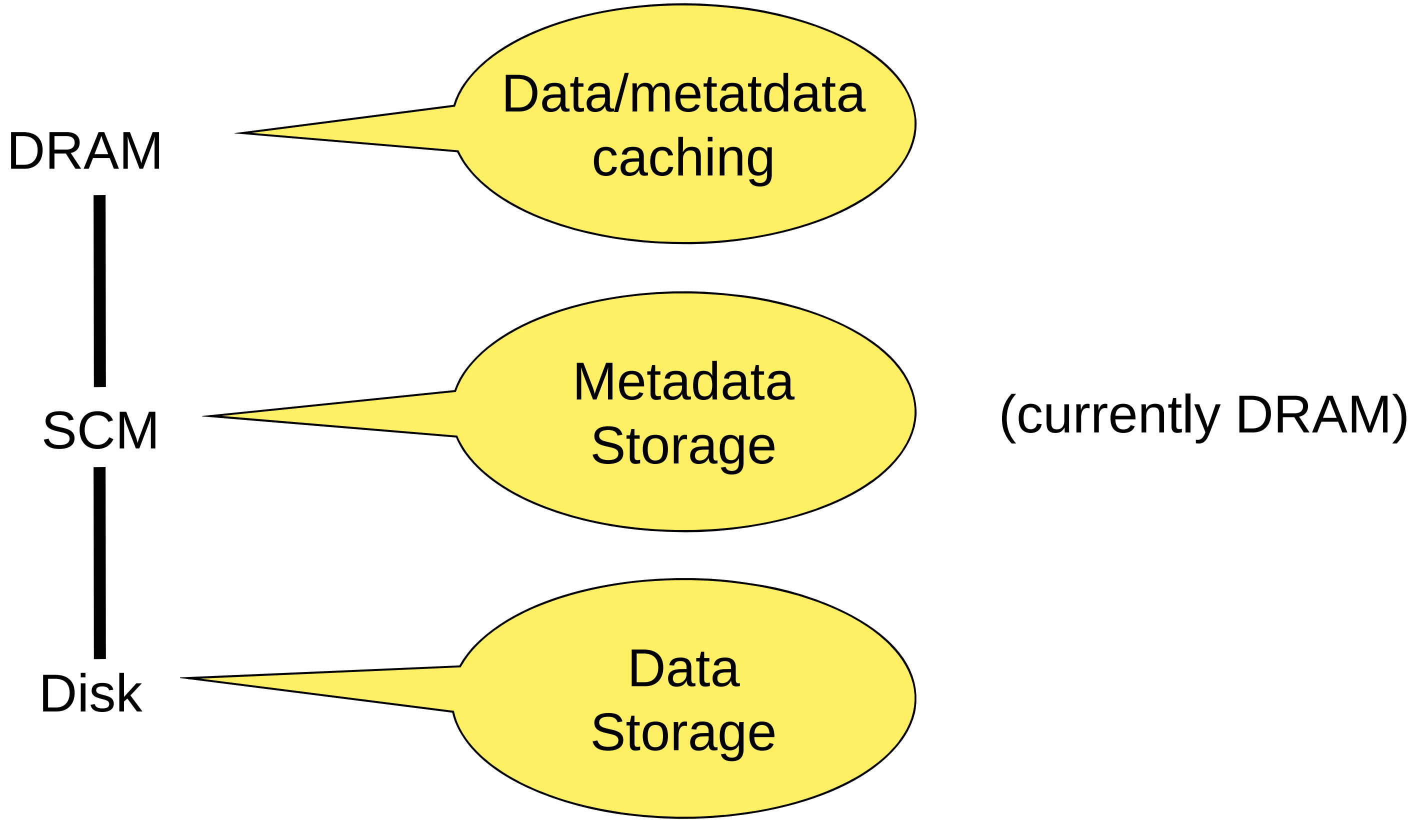
- File System Services
  - Copy-on-write after a snapshot
  - Versioning
  - Encryption
  - Mirroring

Querying, Navigation, Indexing

- Simple command line language
  - `ls2 /home@lattrs:a=b/file`
- GUI applications for display and navigation
  - LiFSBrowse: used for Software Evolution Study
- Crawler to discover file similarity
  - generates weighted links

Implementation: Target System

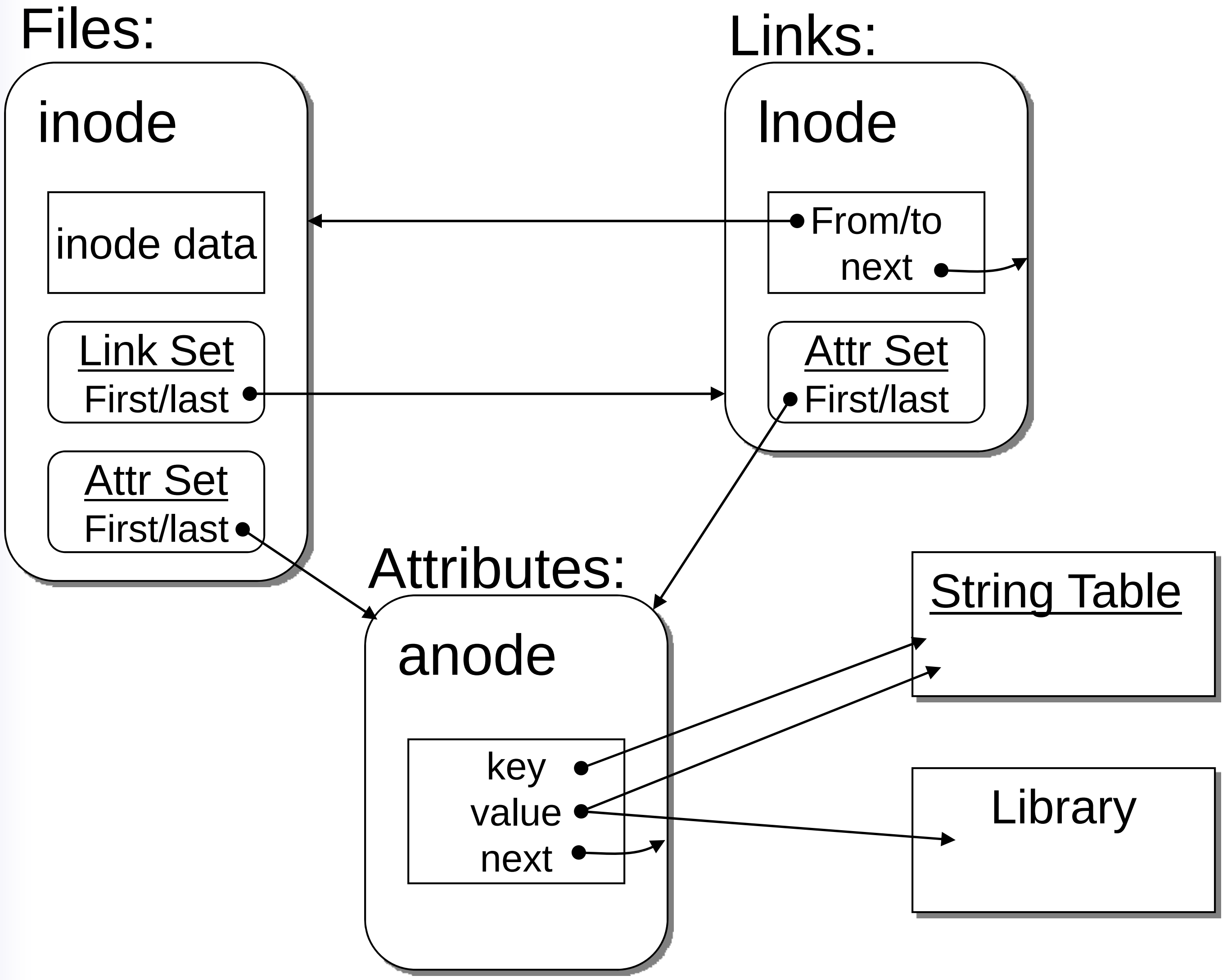
- Storage Class Memory (SCM):
  - Non-volatile, byte-read/writable (MRAM, FeRAM, …)



Implementation: Overview

- Linux + FUSE
  - FUSE redirects VFS calls to user space process
  - Initial proof of concept: Postgres database for metadata
  - Current prototype: No database
- Data structures optimized for link traversal
  - Most use cases call for context of a given file or set of files
  - Traversal is expensive on relational databases
  - Attribute key/value stored in string tables: small space, fast match!
- Fast chunk allocator for SCM
  - All SCM data structures are relocatable

Implementation: Data Structures



Status & Summary: LiFS

- Initial prototype state
- Number of new concepts & interesting challenges
- Links & attributes are surprisingly useful
- File triggers powerful mechanism for file system services
- Ongoing Work
  - High performance data structures for metadata
  - Search & traversal across local & distributed links
    - Metadata placement among nodes
  - Query language (looking at W3C's Sparql)
  - Protection & Composition of file triggers

Related Work

- Queryable File Systems:
  - Semantic File System, Inversion File System, Fan-out Unification File System, Logic File System
- In-Memory File Systems:
  - HeRMES, Conquest
- Advanced Commercial File Systems
  - Microsoft’s WinFS, Apple’s Spotlight
- Active Infrastructures
  - Placeless Documents
- The Semantic Web
- Digital Preservation
  - Dspace, Greenstone, Fedora