LiFS
An Attribute-Rich File System for Storage-Class Memories

Sasha Ames, Nikhil Bobb, Kevin M. Greenan,
Owen S. Hofmann, Mark W. Storer,
Carlos Maltzahn, Ethan L. Miller, Scott A. Brandt
SSRC, UC Santa Cruz

Problem

- Explosion in number & variety of files
- Directories insufficient mechanism
- Applications forced to manage own metadata

A long time ago ...

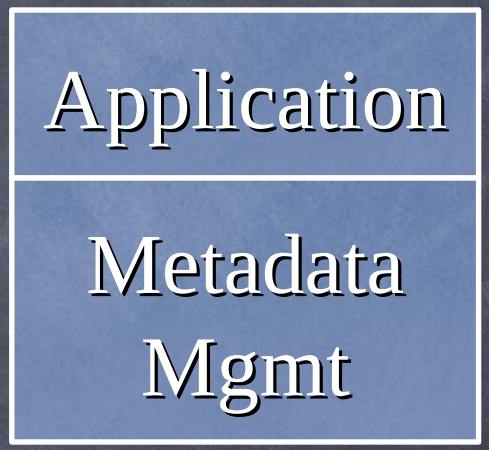
- Few files and simple conventions

Application

Few Files

... Today

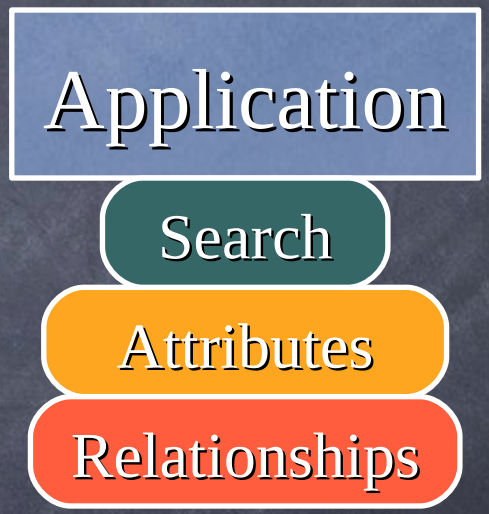
- Few files and simple conventions
- Many files require complex management



Many Files

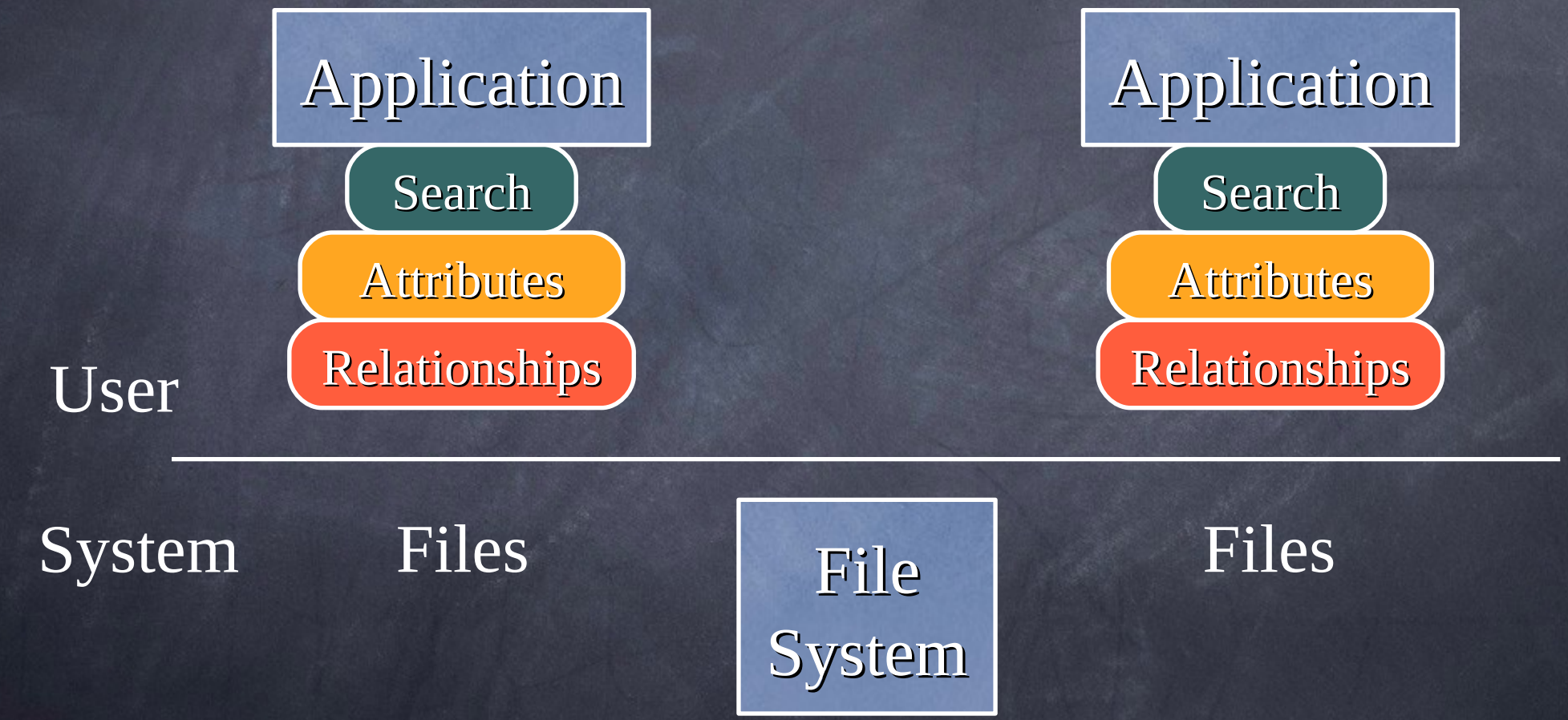
Metadata Management

- Few files and simple conventions
- Many files require complex management

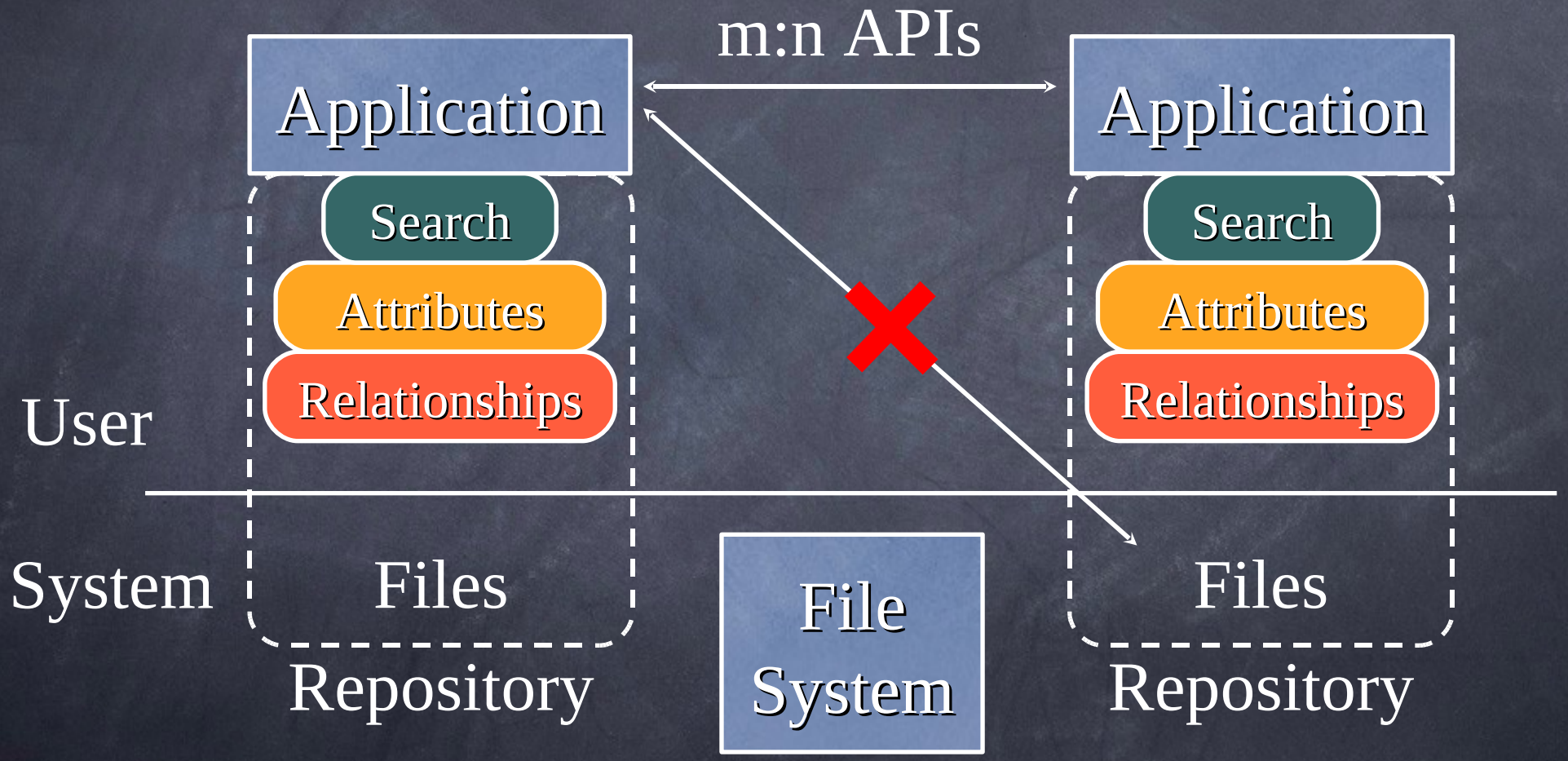


Many Files

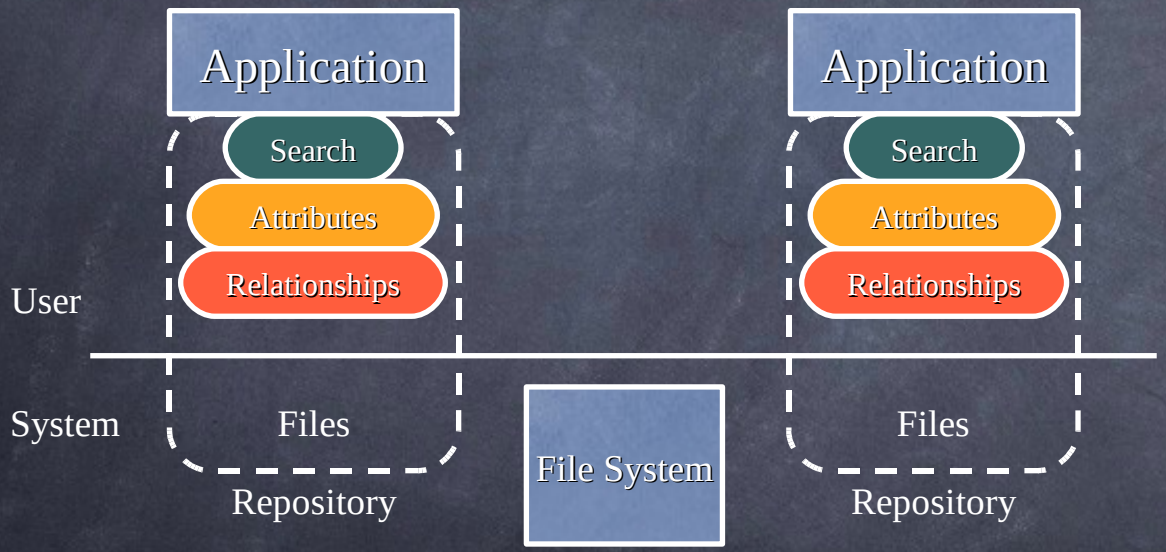
So what?


Application
Search
Attributes
Relationships
Application
Search
Attributes
Relationships
User
System
Files
File System
Files

Difficult to share

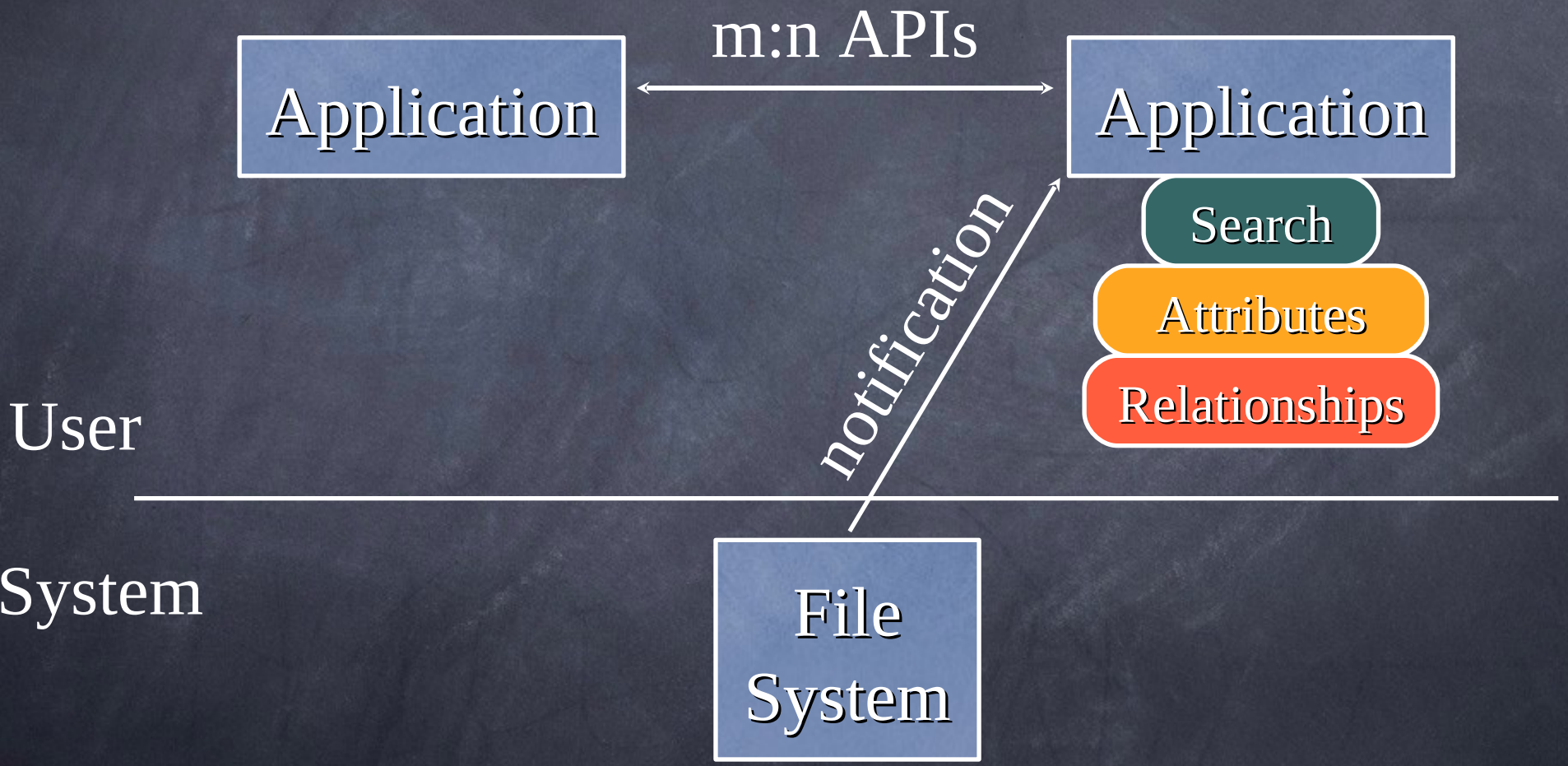


So what?

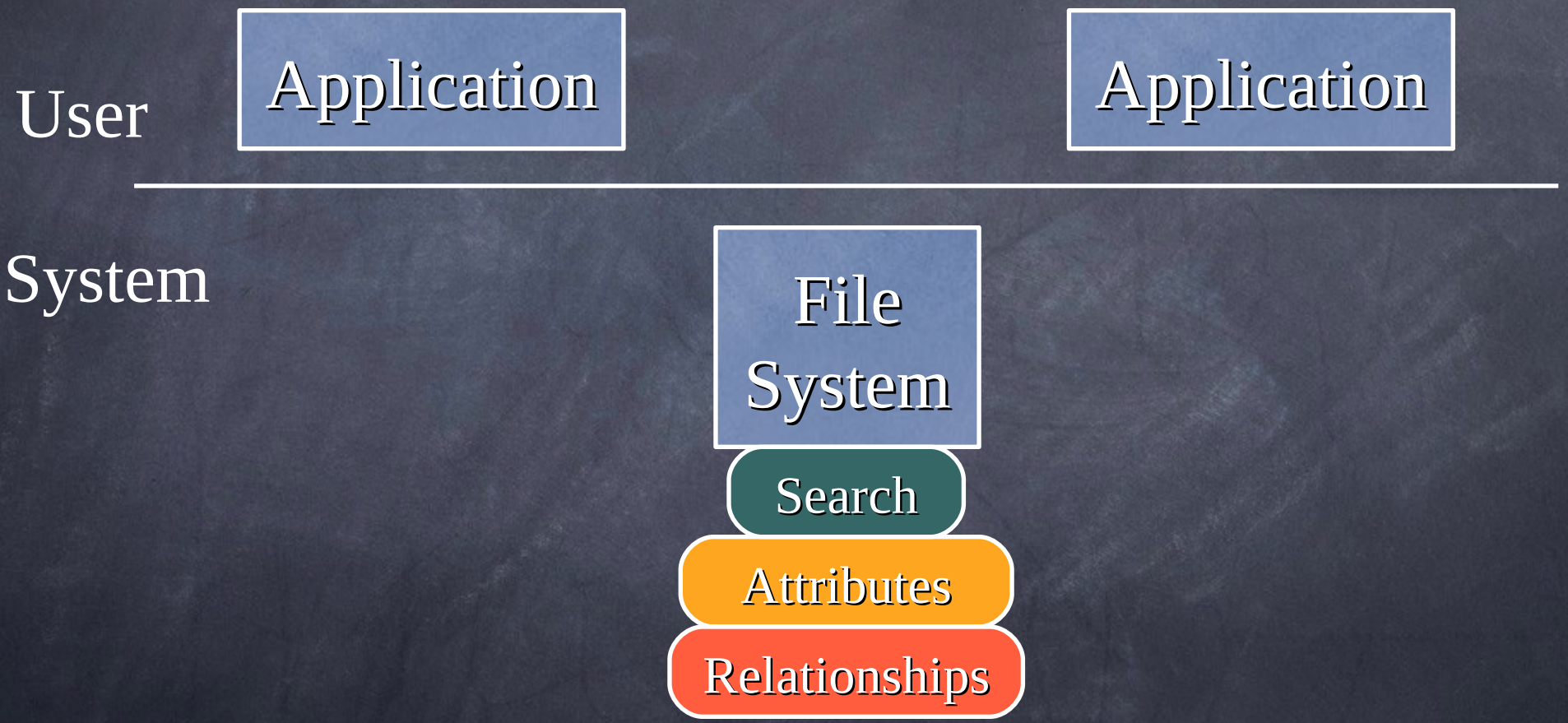


- No standards
- Not portable
- Difficult to share
- Inefficient
- Duplicate effort

Metadata Application?



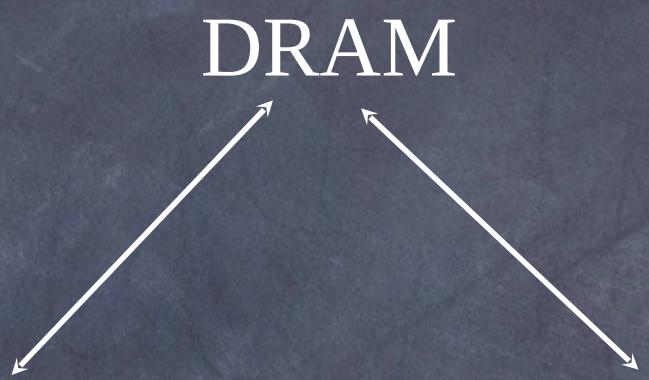
LiFS - Linking File System



What follows ...

- Design
- Implementation
- Evaluation
- Related Work
- Conclusions, Future Work

Design Assumption



Disk

Data

high latency
high bandwidth

NVRAM

Metadata

low latency
medium bandwidth

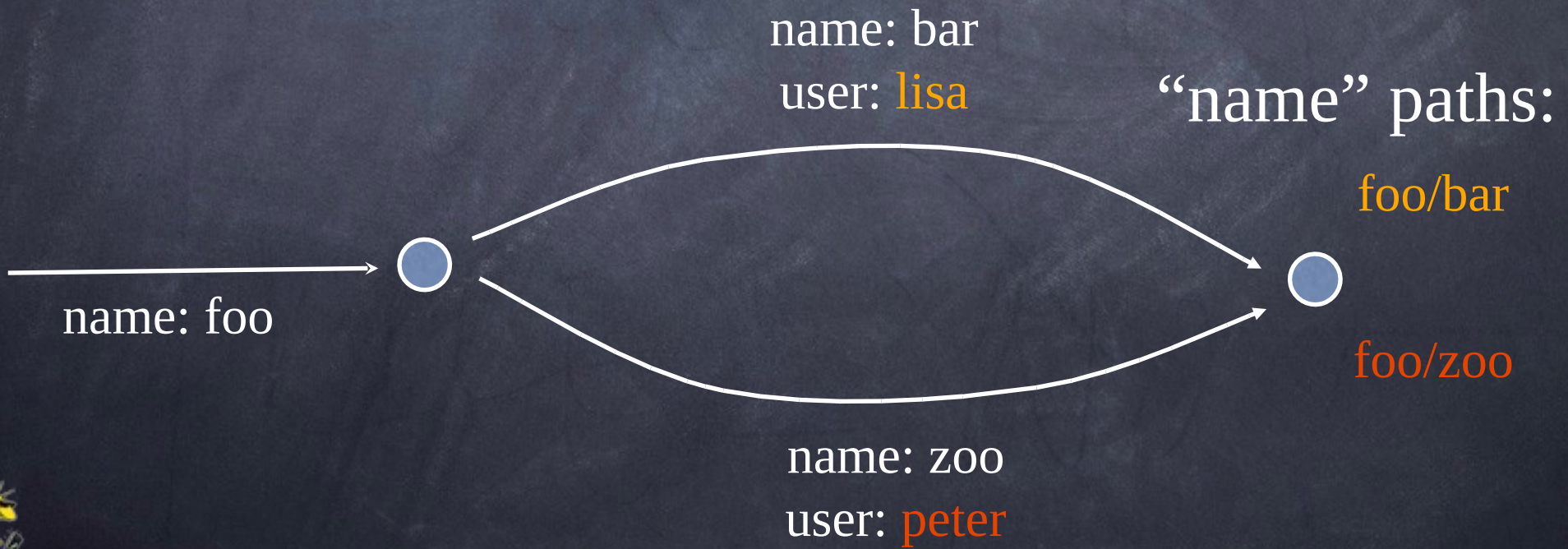
LiFS Design Features

- Files: extended attributes
- Directed links between files
- Links: attributes

LiFS: Linking File System

Naming in LiFS

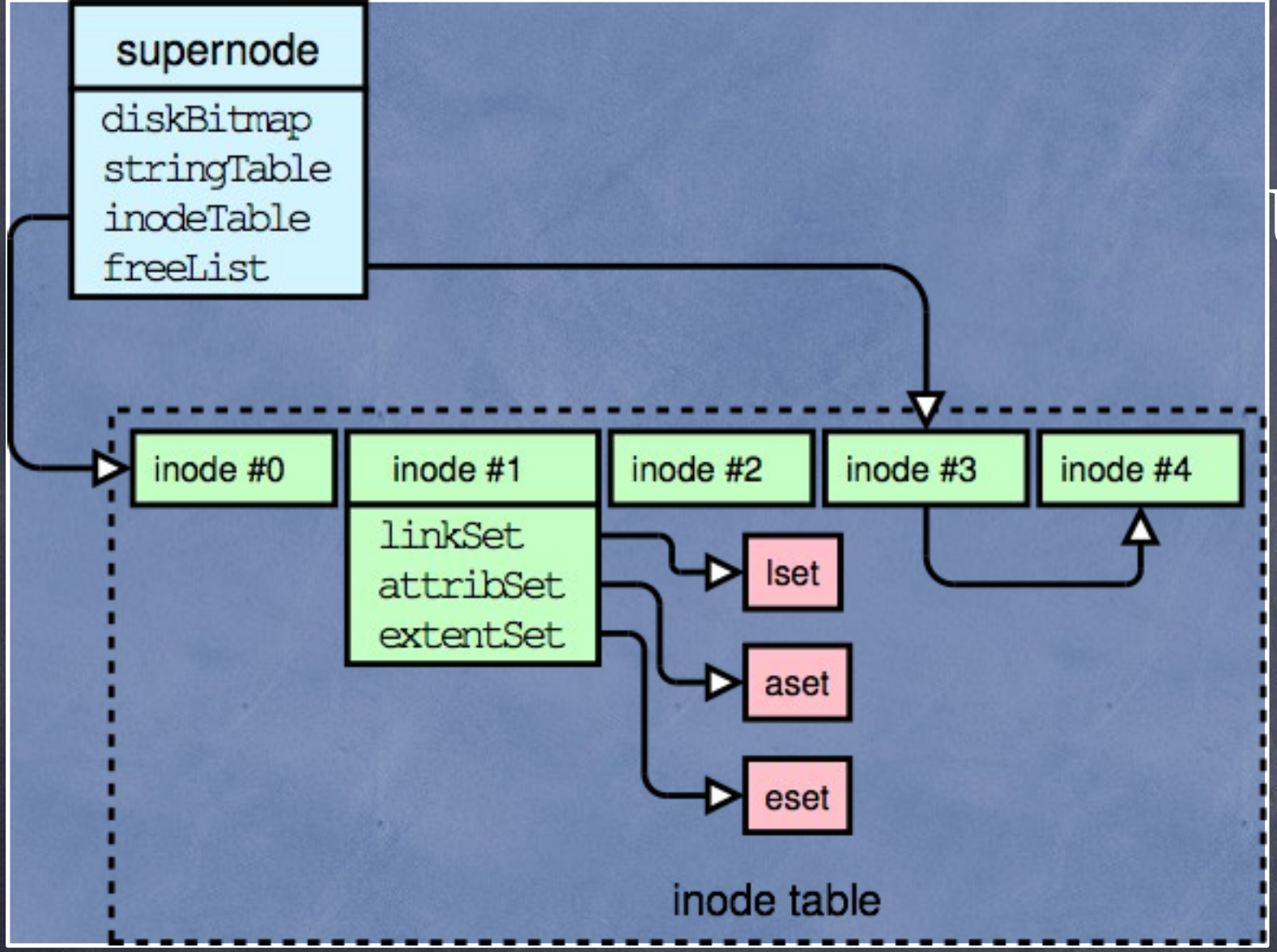
- Directed links between files
- Links: attributes



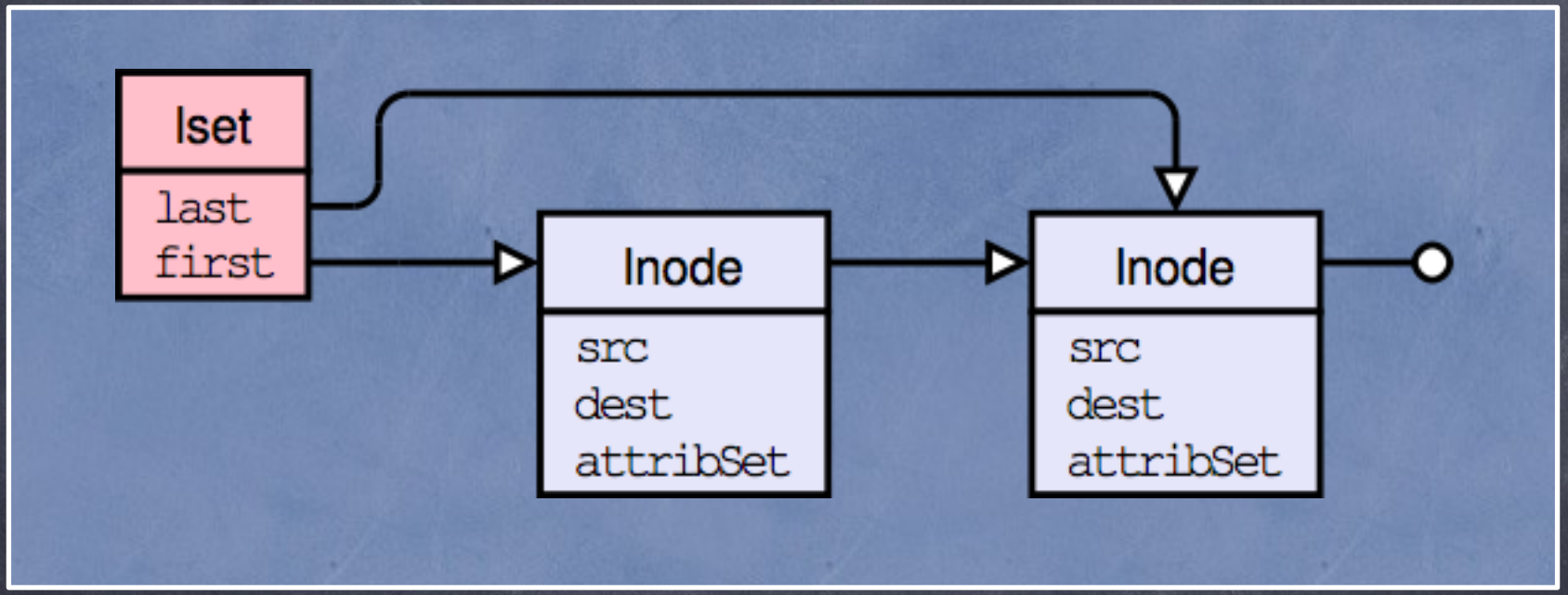
New System Calls

System Call	Function
rellink	create relational link
rmlink	remove relational link
setlinkattr	set attr on relational link
openlinkset	return handle of a source file's link set
readlinkset	get link name and attrs of next link in a link-set

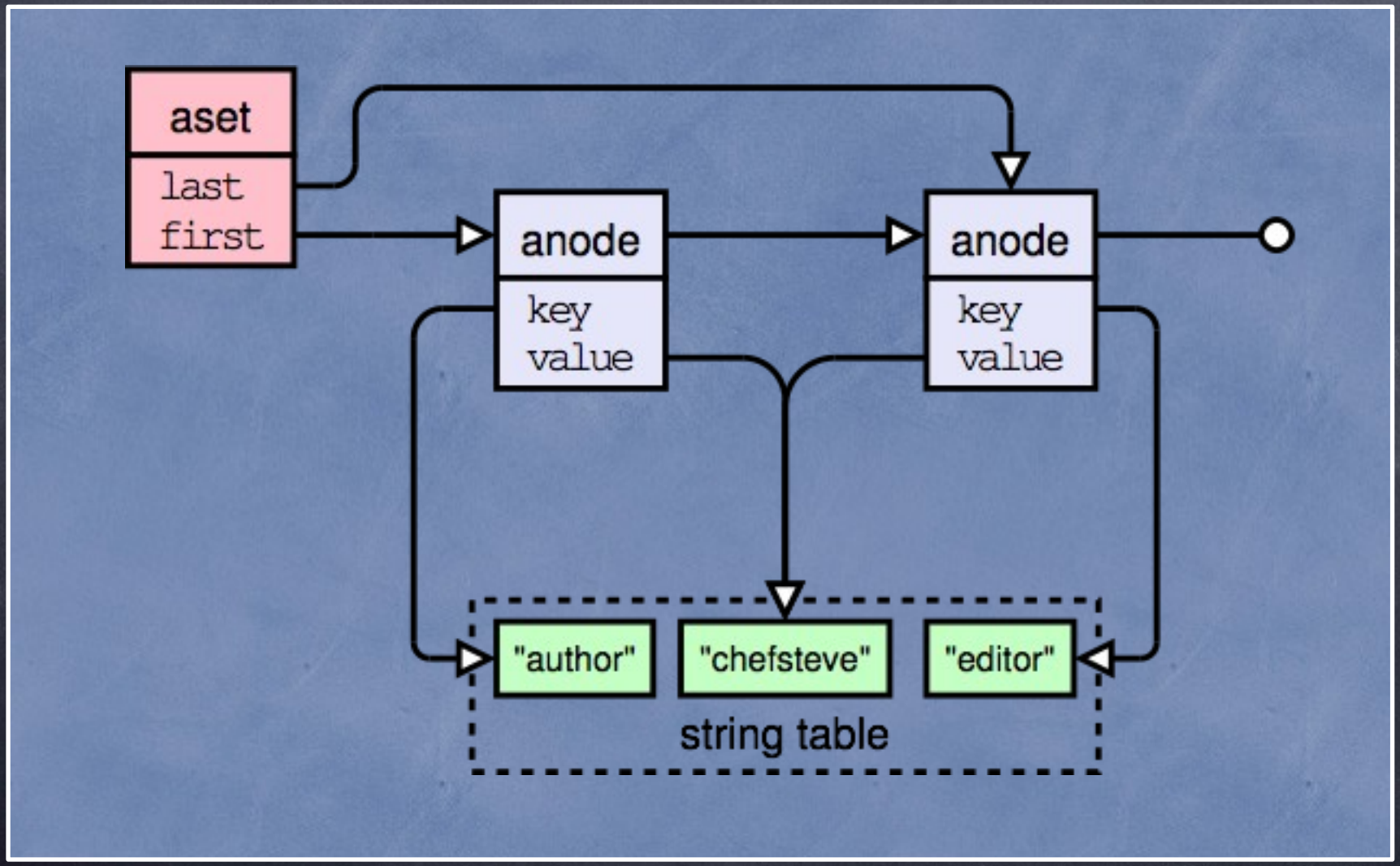
ures


supernode
diskBitmap
stringTable
inodeTable
freeList
inode #0
inode #1
inode #2
inode #3
inode #4
linkSet
attribSet
extentSet
lset
aset
eset
inode table

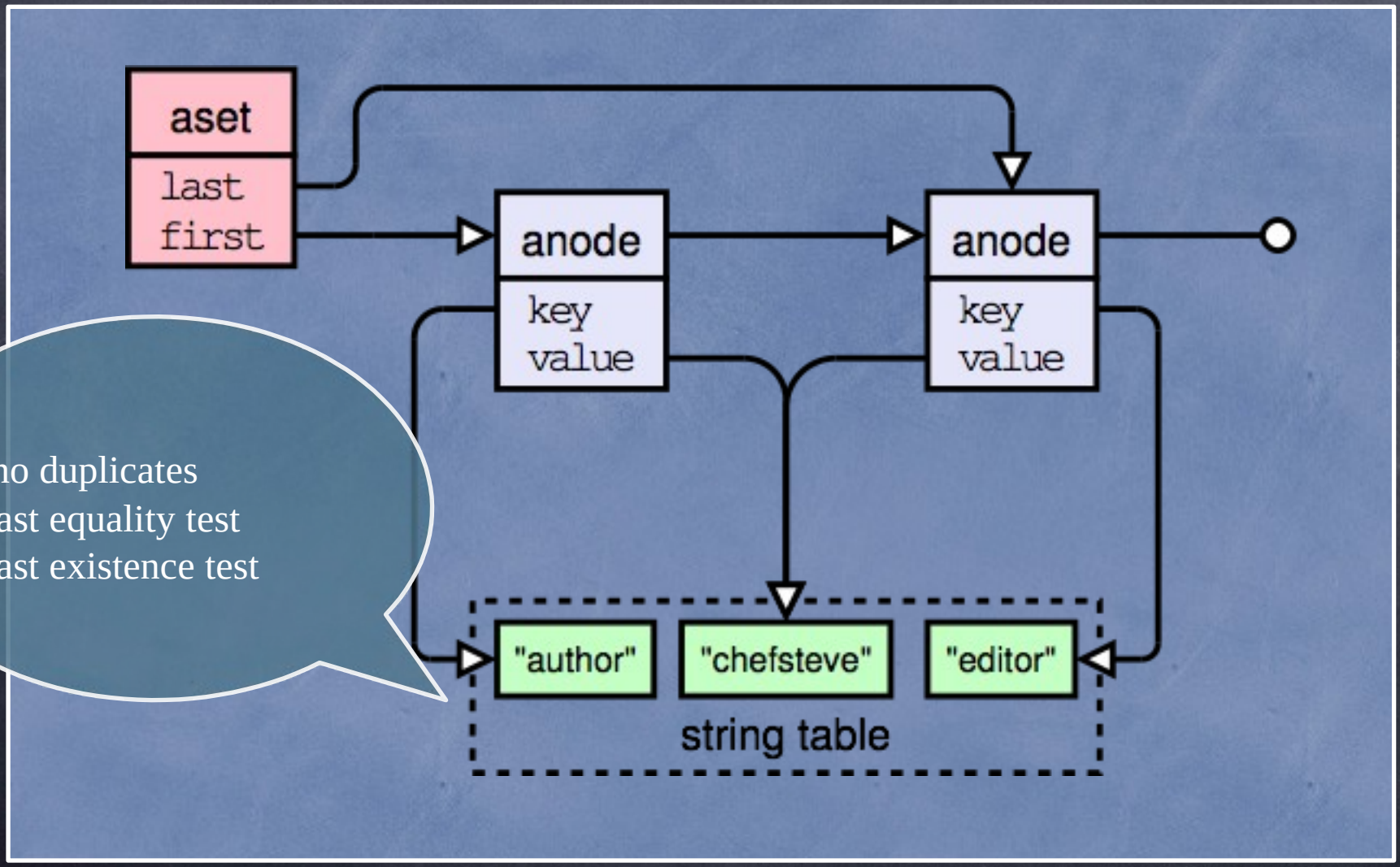
Link Set


lset
last
first
lnode
src
dest
attribSet
lnode
src
dest
attribSet

Attribute Set



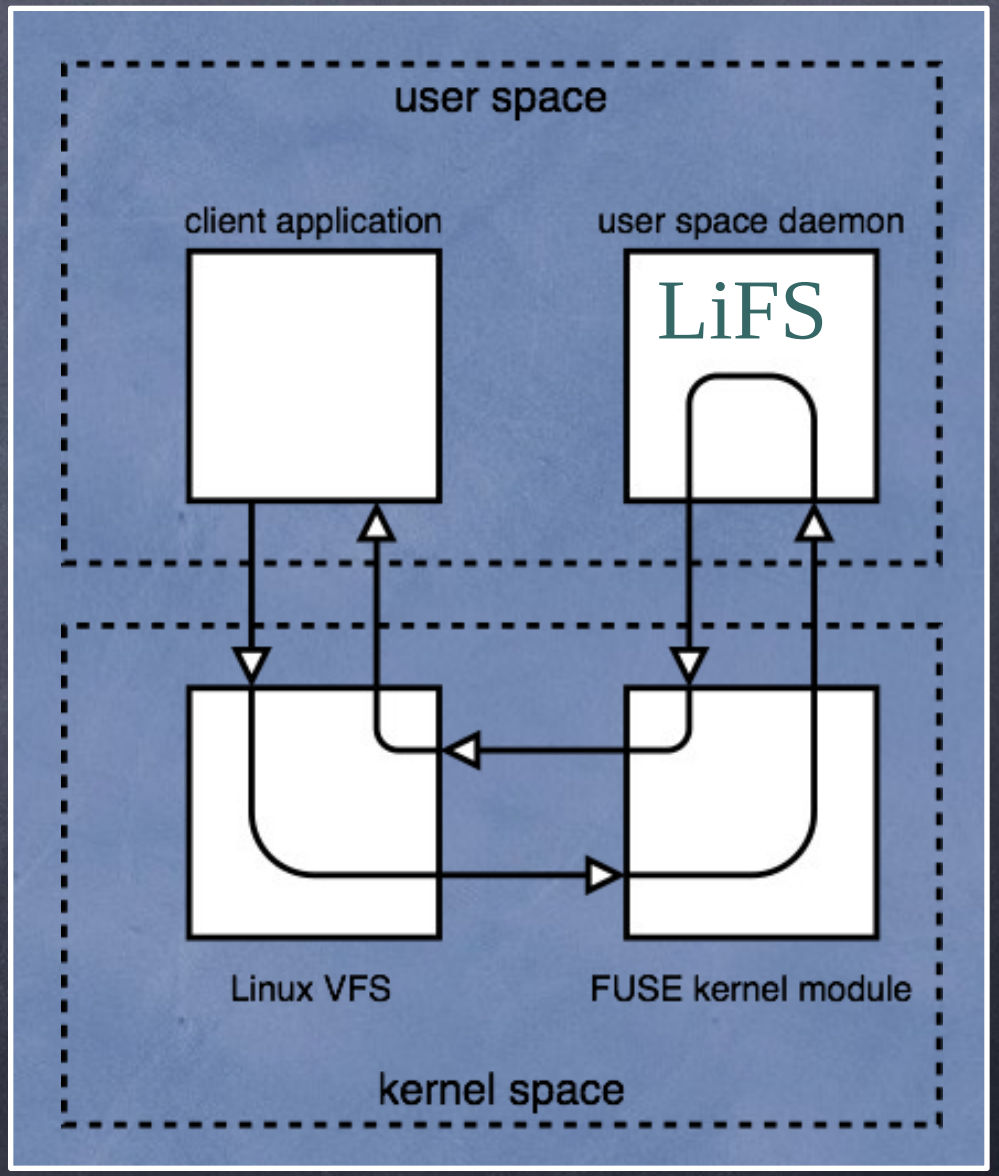
String Table



- no duplicates
- fast equality test
- fast existence test

Implementation

- FUSE: maps VFS calls back to user space
- NVRAM: Locked system memory in DRAM
- Custom NVRAM allocator with fixed-sized pools
- Lookup optimizations:
  - String table
  - Full path name cache



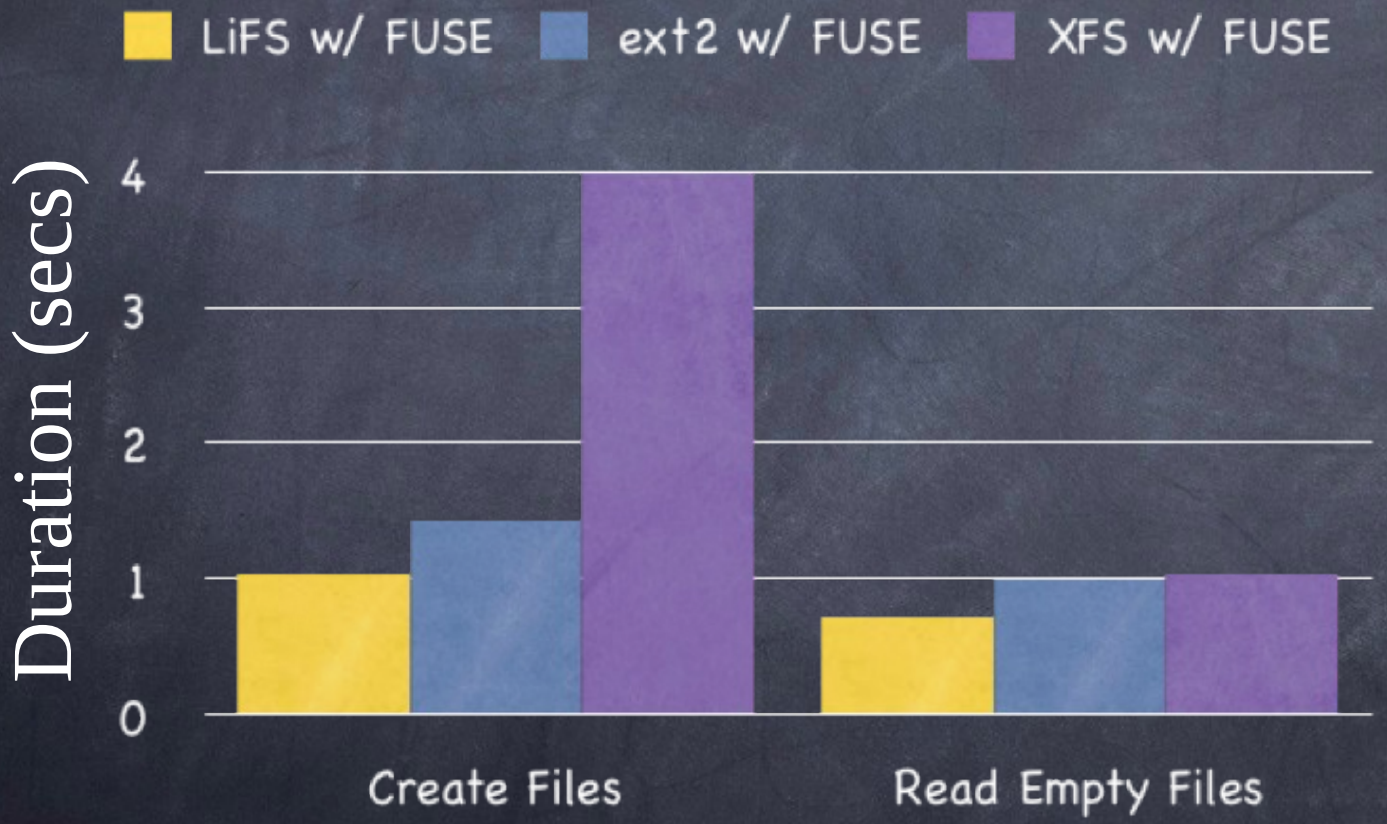
Evaluation

- Goals:
- Traditional FS Ops: speed & scalability compared to other file systems
- New fs ops: scalability
- FUSE overhead

Setup:

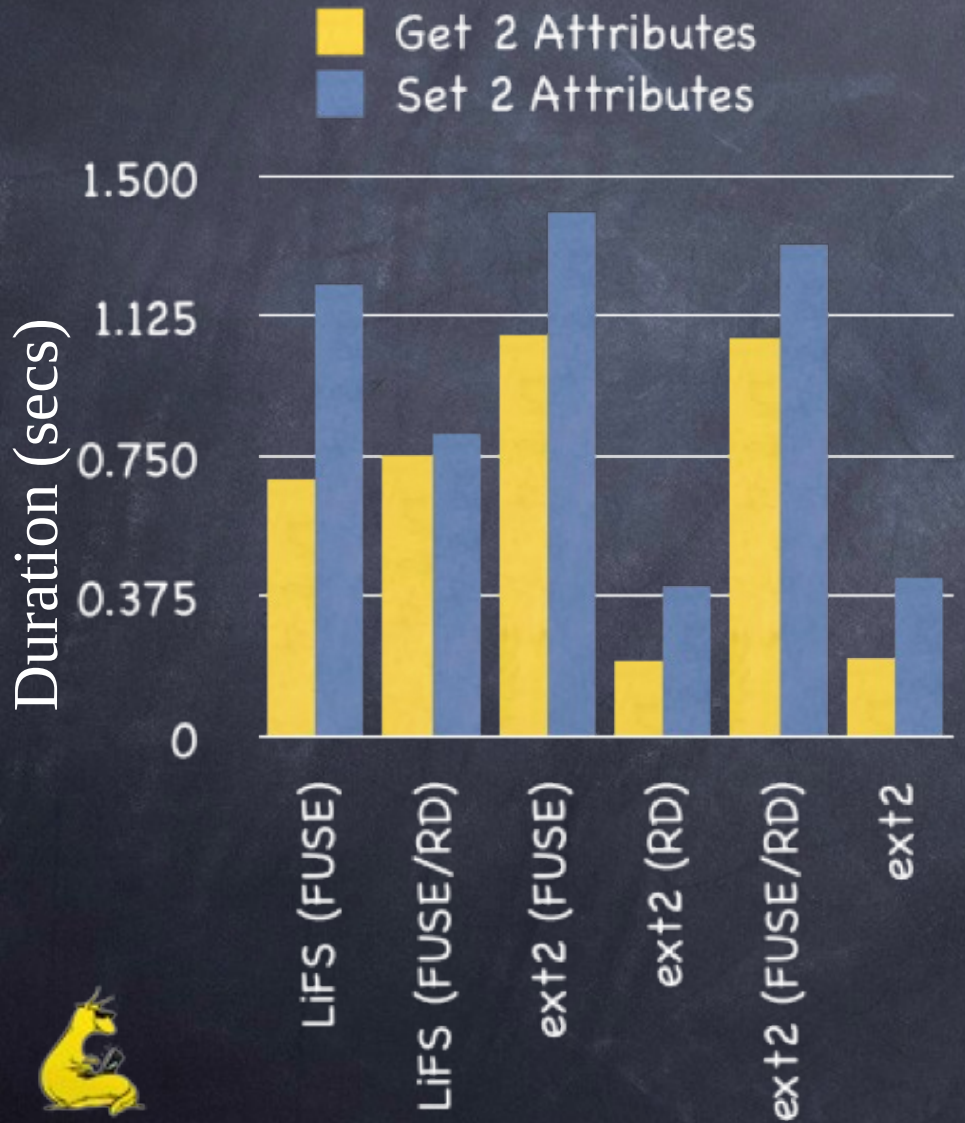
- Sun Workstation running Linux 2.6.9-ac11
- AMD Opteron 150, 2.4 GHz
- 1 GB DRAM

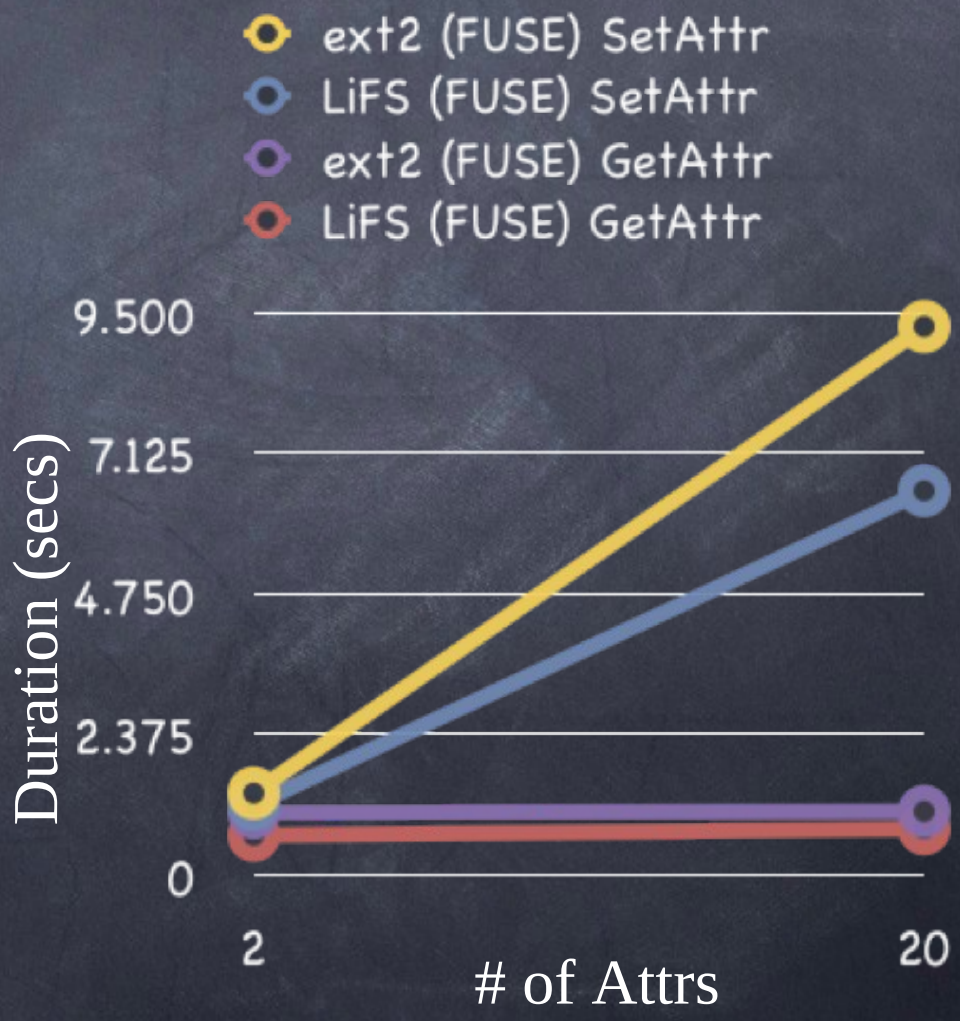
Files



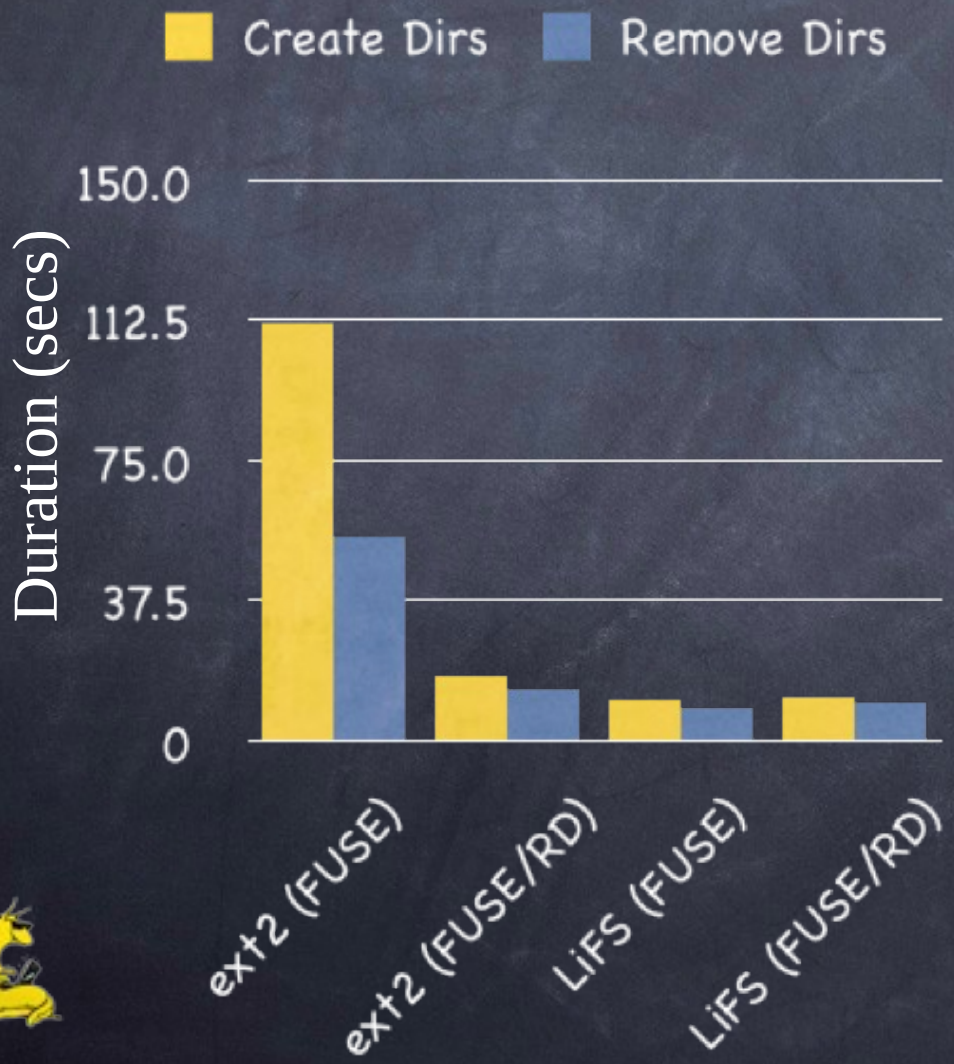
- Directory Tree: k=5, d=5, n=4, 15,620 files
- Freshly created file systems
- LiFS is competitive

File Attributes



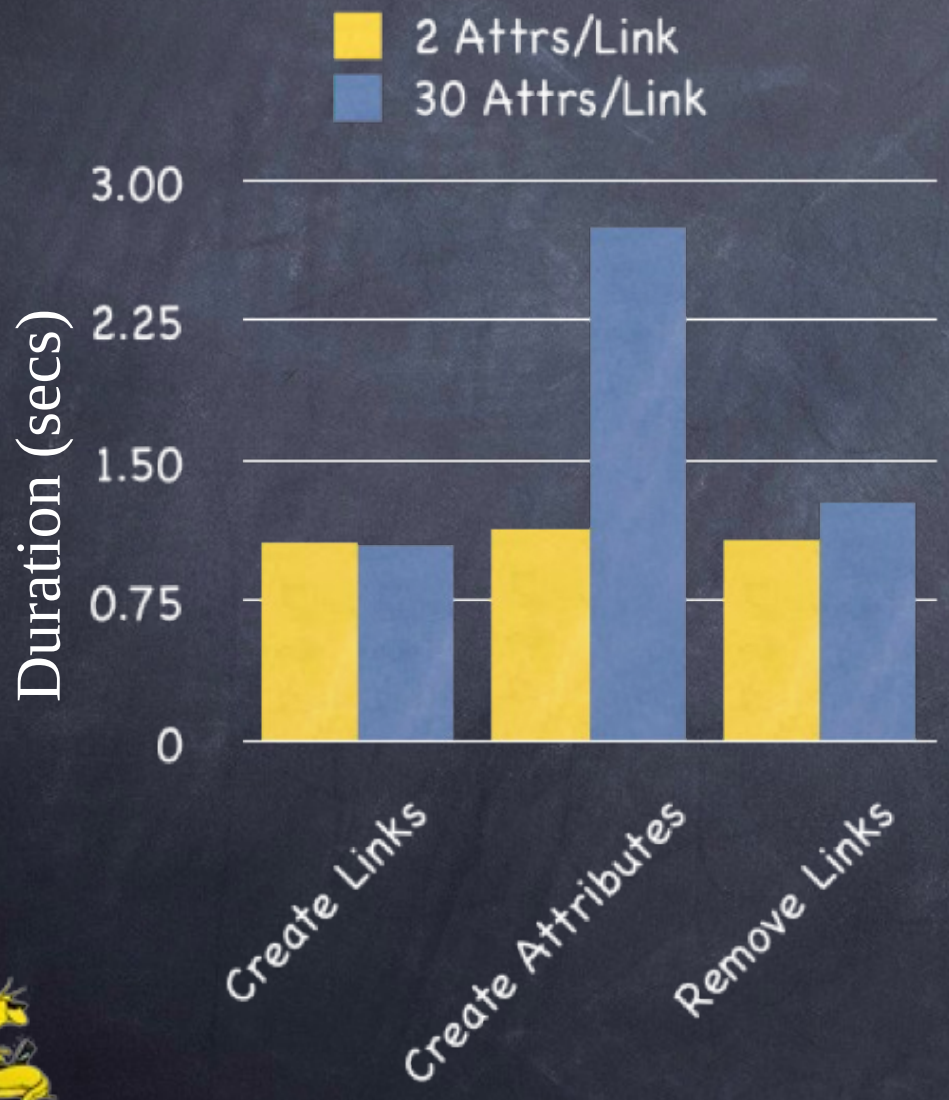


Create/Remove Dirs



- Directory tree: k=10, d=6, n=1, 111,110 dirs
- LiFS performs better than ext2 with FUSE & RAM-disk

Create/Delete Rel. Links



- Directory Tree: k=5, d=5, n=4, 15,620 files
- Duration of processing 15,620 random links
- More attributes slow down identifying link

Related Work

- Queryable File Systems
- In-Memory File Systems
- Advanced Commercial File Systems
- The Semantic Web
- Digital Preservation

Queryable File Systems

- Attributes allow expressive queries
- Use secondary storage only
- No linking mechanism with attributes

In-Memory File Systems

- Lack advanced file system features
- Lots of research to overcome challenges of persistent memory
- Database research on utilizing persistent memory

Advanced Commercial File Systems

- Microsoft’s WinFS, Apple’s Spotlight, Beagle (Linux with Inotify), Sun’s ZFS
- No attributed links
- No metadata management in NVRAM

The Semantic Web

- Links & Attributes same expressiveness
- LiFS as file system or storage layer

Digital Preservation

- Obsolescence by broken data relationships
- Large efforts on institutional level
- Need to also extent to file systems
- LiFS provides infrastructure

Future Work

- More efficient data structures (Workloads?)
- Fault tolerant data structures
- Online file system consistency checker
- Extend to distributed storage
- Explore use of rich metadata structures without NVRAM

Conclusions

- Contributions:
  - Rich file system metadata via links & attributes
  - Common high-performance metadata store for applications
- Advantages: performance, simplicity, expressiveness

Thanks to:

- Faculty and students of the SSRC
- NSF grant 0306650
- SSRC sponsors: Hewlett Packard Laboratories, Hitachi Global Storage Technologies, IBM Research, Intel, Microsoft Research, Network Appliance, Rocksoft, Symantec, and Yahoo.

Thank You!

UCSC Storage Systems Research Center

http://ssrc.cse.ucsc.edu

carlosm@soe.ucsc.edu