Kudos A Modular File System Architecture

Andrew de los Reyes, Chris Frost, Eddie Kohler, Mike Mammarella, Lei Zhang
http://kudos.cs.ucla.edu kudos@lists.ucla.edu

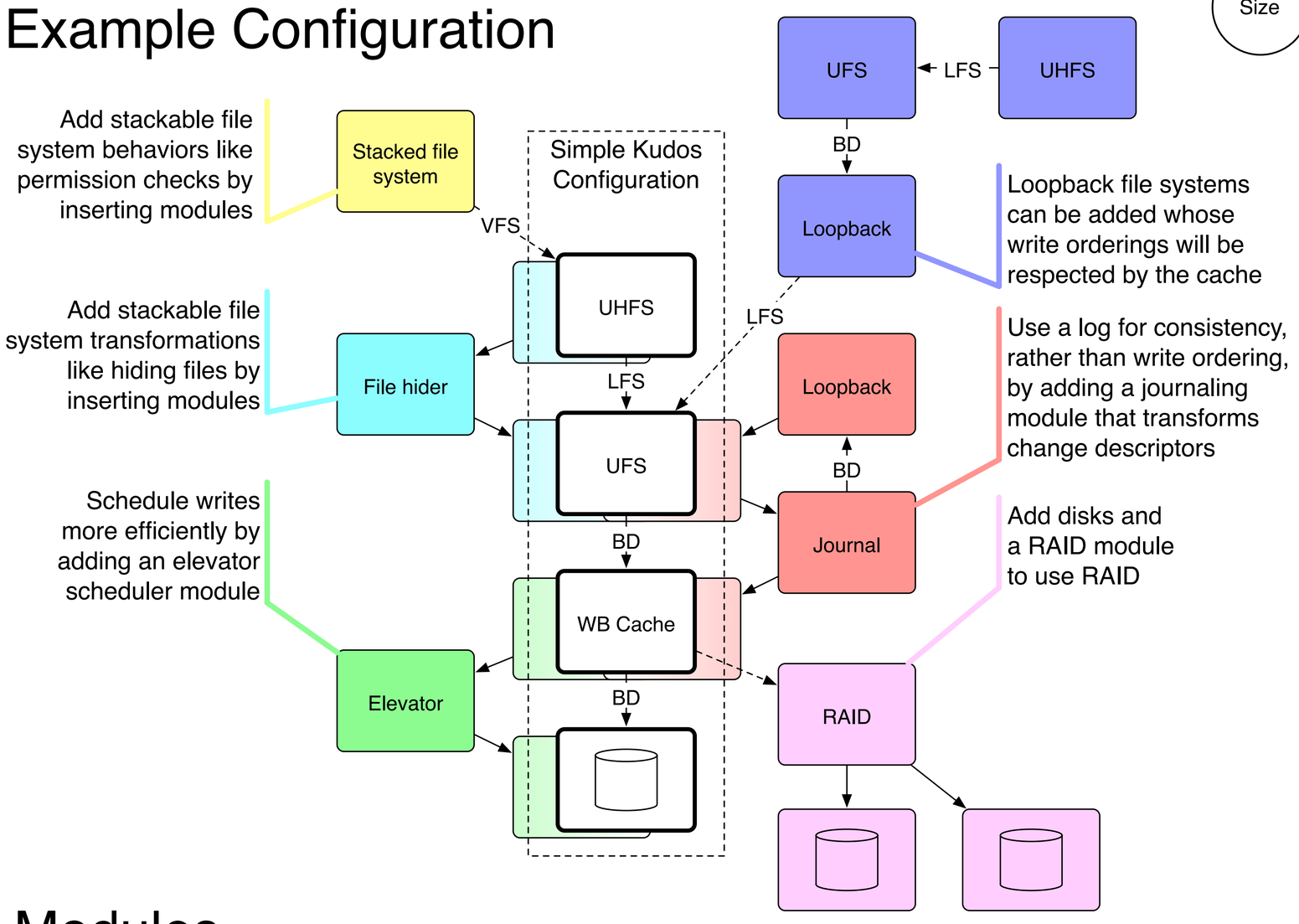
UCLA Computer Science

Motivation

- Disks and applications' storage requirements are growing exponentially
- Result: Increased application demands on file systems, both functionality (encryption, extensible metadata, snapshots...) and robustness (surviving failure: *consistency*)
- Existing "stackable" file system architectures simplify some feature development, but ignore consistency maintenance
- File system interfaces limit application behavior: Robust applications must either fully synchronize (expensive) or use the raw disk (difficult)

The Kudos modular file system architecture makes file system features easy to develop, simplifies consistency maintenance, and supports new robust application semantics.

Example Configuration



Modules

Uniform High-level File System (UHFS)
- translates VFS-level calls to sequences of LFS operations
- connects the resulting change descriptors
- can use any LFS module to create a full file system

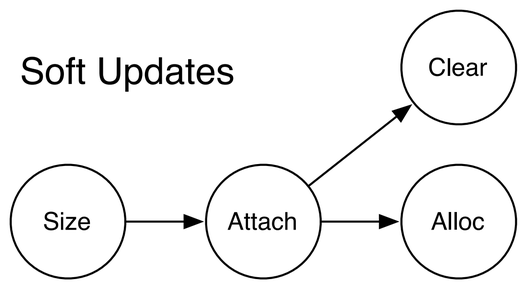
Unix File System (UFS)
- implements the on-disk format used by BSD, Solaris, and many other Unix-like operating systems
- LFS modularity and change descriptors help simplify UFS code

Write-Back Cache
- accepts change descriptors generated by other modules and sends them towards the disk in a valid order

Journal
- changes the arrangement of incoming change descriptors and adds new change descriptors to implement a different consistency protocol
- transformation illustrated at right

Change Descriptors

- File systems order disk writes so that in the event of a crash, data is recoverable (e.g. "Soft Updates" in UFS)
  - For example, allocating a disk block before referencing it ensures that a system crash can't result in doubly-allocated blocks.
- Existing consistency systems must be reimplemented for each file system
- In Kudos, **change descriptors** represent all in-memory changes to the disk, and express these ordering requirements in a file system independent manner



Soft updates change descriptor graph, including the dependencies for adding a newly allocated block to an inode. Writing the new block pointer to an inode (Attach) depends on initializing the block (Clear) and updating the free block map (Alloc). Updating the size of the inode (Size) depends on writing the block pointer.

Module Interfaces

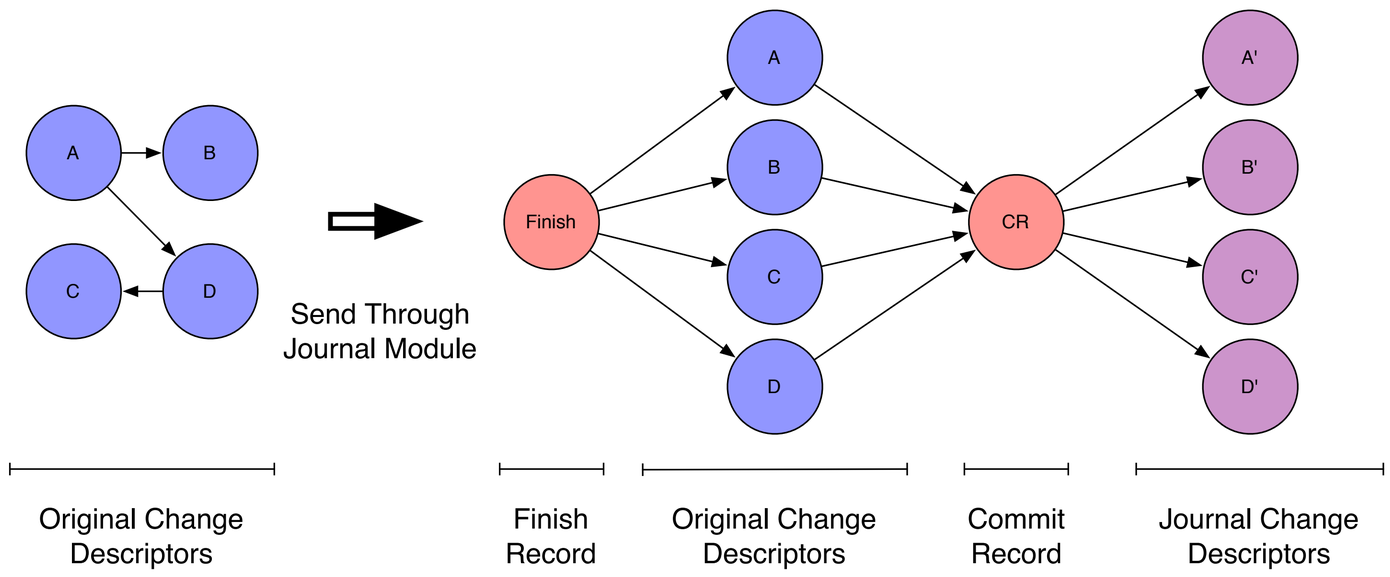
A complete Kudos configuration is composed of many modules with three major interfaces:

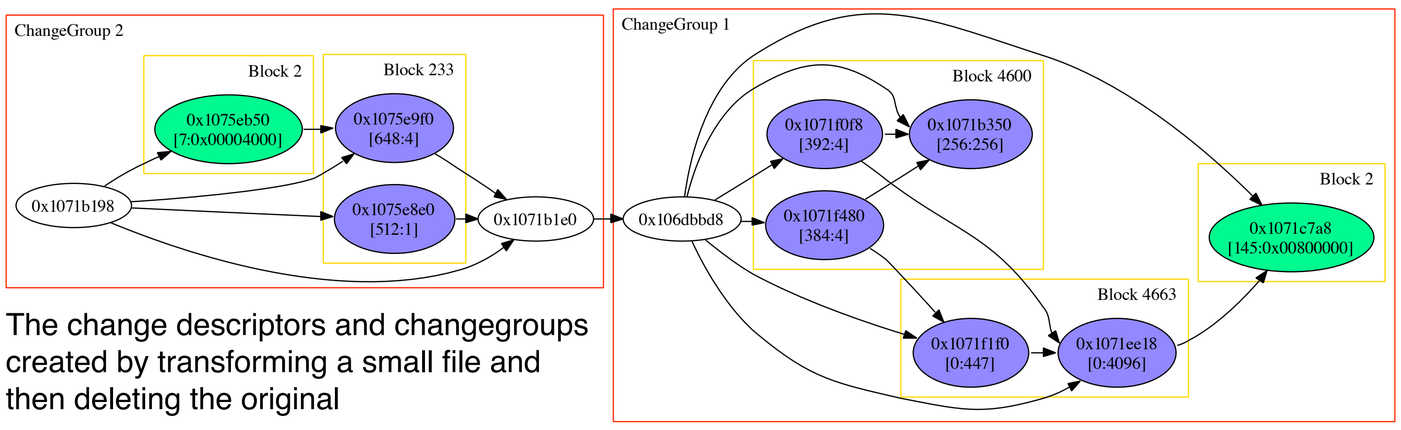
- Virtual File System (VFS)
  - Abstract file and folder operations (e.g. open/write files)
- Block Devices (BD)
  - Abstract disk block operations (e.g. read/write blocks)
- Low-level File System (LFS)
  - Abstract file system structure operations (e.g. allocate a block, append a block to a file)

The novel LFS interface separates high- and low-level file system operations, simplifying development of high-level modules (reading a file) and low-level modules (locating a block). High-level modules arrange change descriptors to preserve Unix consistency semantics across multiple LFS operations.

All modules in the system can create, inspect, and modify change descriptors, allowing them to specify or adjust disk write ordering requirements. Modules need not understand each other's consistency protocols.

Journaling Transformation





The change descriptors and changegroups created by transforming a small file and then deleting the original

Changegroups

- Give applications control over their change descriptors
- Identifies a set of change descriptors generated by file system operations
- Applications specify ordering requirements among changegroups
  - Example: Safely transform a file (at least one copy available no matter what)
  - Before: transform, then sync, then delete original
  - Changegroups: transform, *add dependency*, delete original
  - Result minimizes writes, should be faster

Evaluation

Extract, then delete Linux 2.6.15 source code:

	Untar	Delete
Kudos UFS (SU)	24.50 s	11.69 s
FreeBSD UFS (SU)	20.47 s	4.71 s
Linux ext3 (journal)	12.90 s	4.90 s

	Create 100 small files	Remove file, add file
Kudos UFS (SU)	122 writes	15 writes
FreeBSD UFS (SU)	135 writes	83 writes

Our prototype implementation is 20% to 2x slower than FreeBSD's UFS, but writes fewer blocks. We are currently optimizing the prototype.

Changegroup Case Study
- To assess the utility of changegroups in a real world application, we updated the University of Washington's IMAP mail server to take advantage of changegroups
  - Each IMAP command is performed within a changegroup and is made dependent on the preceding command
  - Mailbox checkpoints sync the current changegroup to disk
- Benefits:
  - Correctness is easier (changegroups capture all changes implicitly)
  - Fewer syncs are needed for correctness (e.g. message copies), resulting in fewer writes to disk

	Move 100 messages
Kudos (with changegroups)	919 writes
Kudos (no changegroups)	1537 writes
FreeBSD UFS	2200 writes

Benefits

- Easier to write crash-recoverable applications
- File system modules easier to understand and make correct
- Faster innovation, better file systems!