Journaled File System (JFS) for Linux
UT, Texas
April 25, 2003

Dave Kleikamp
shaggy@austin.ibm.com
Linux Technology Center - JFS for Linux
IBM Austin

http://oss.software.ibm.com/developer/opensource/jfs/project/pub/jfs042503.pdf

Overview of Talk

- Linux Filesystems
- Features of JFS
- JFS project
  - GPL Licensed
  - Source of the port
  - Goal to run on all architectures
    - (x86, PowerPC, S/390, ARM)
  - Goal to get into kernel.org source 2.4.x & 2.5.x
  - New features being added
- Other Journaling File Systems
  - Ext3, ReiserFS, XFS

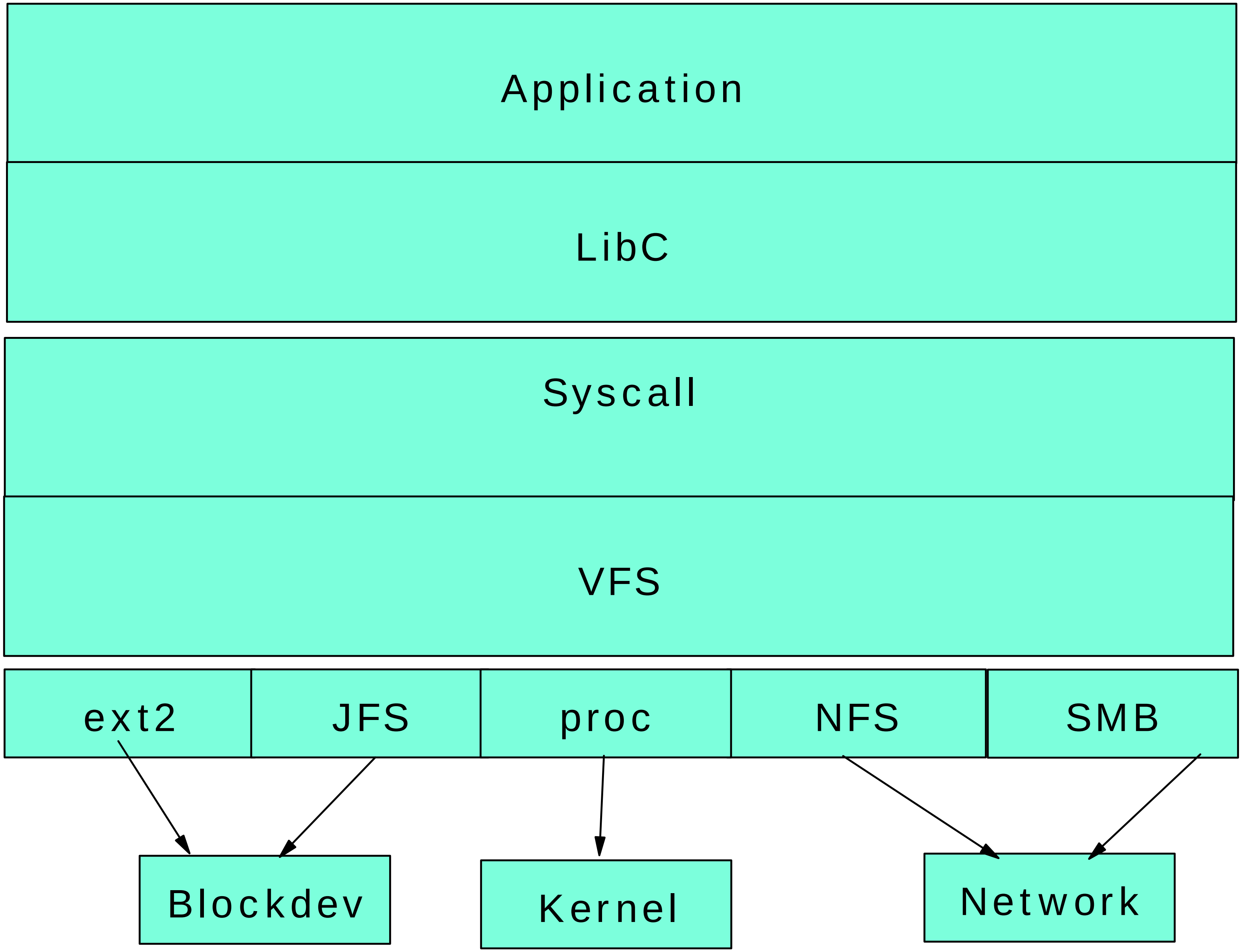
Linux Filesystems

- Local disk filesystems
  - Ext2, msdos/vfat, isofs/udf, ntfs/hpfs,ufs, ...
- Newer journaling filesystems
  - Ext3, ReiserFS, XFS, JFS
- Network filesystems
  - NFS, AFS, SMBFS, CIFS
- Distributed filesystems
  - Coda, InterMezzo, GFS, GPFS
- Others
  - procfs, devfs, shmfs, ramfs, sysfs

Virtual Filesystem Layer

- abstraction layer above file systems
- Filesystems may be modular
  - Module name = fs type in /etc/fstab
- VFS does not know fs specifics
- VFS works with generic superblock & Inode
  - Superblock/inode hold pointers to fs data/functions
  - VFS calls method in inode by name

Virtual and Filesystem



VFS & FS

- Mount of FS checks /etc/fstab for type
- Kernel loads module for filesystem
- Filesystem registers itself with kernel
  - VFS only knows fs type, fs read_super method
- VFS calls read_super
  - Reads superblock from disk, initializes generic sb
  - Superblock points to fs-specific operations
    - Read/write/update/delete inode
    - Write superblock
    - Statfs(returns used & free space, etc.)

VFS & FS

- read_super loads root inode
- inode has fs-specific data, operations
- Inode operations
  - Create/lookup/link/unlink file
  - mkdir/rmdir
  - rename
- File operations
  - Seek/read/write/sync
  - mmap/ioctl

VFS Role Summary

- Keep track of available file system types.
- Associate (and disassociate) devices with instances of the appropriate filesystem.
- Do any reasonable generic processing for operations involving files.
- When filesystem-specific operations become necessary, vector them to the filesystem in charge of the file, directory, or inode in question.

Why journal?

The problem is that FS must update multiple structures during logical operation.

- Using logical write file operation example
  - ► it takes multiple media I/Os to accomplish
  - ► if the crash happens between these I/Os the FS isn't in consistent state
- Non-journaled FS have to examine all of the file system's meta-data using fsck
- Journaled file systems uses atomic transactions to keep track of meta-data changes.
  - ► replay log by applying log records for appropriate transactions

Journal File Systems

- Ext3
  - Compatible with Ext2
  - Both meta-data & user data journaling
  - Block type journaling
- ReiserFS
  - New file layout
  - Balanced trees
  - Block type journaling
- XFS
  - Ported from IRIX
  - Transaction type journaling

Why use JFS ?

- Highly Scalable 52 bit file system:
  - scalable from small to huge (up to 4 PB)
  - algorithms designed for performance of very large systems
- Performance tuned for Linux
- Designed around Transaction/Log
  - (not an add-on)
- Restarts after a system failure in seconds

JFS Port

- Proven Journaling FS technology (10+ years in AIX)
- New "ground-up" scalable design started in 1995
  - Design goals: Performance, Robustness, SMP
  - Team members from original JFS Designed/Developed this File System
- JFS for Linux
  - OS2 parent source base
  - OS/2 compatible option
- Where has the source base shipped?
  - OS/2 Warp Server for e-business 4/99
  - OS/2 Warp Client (fixpack 10/00)
  - AIX 5L called JFS2 4/01

JFS Community

Building JFS community

- Mailing list
- Written white papers
- Articles written about JFS
  - Interview With People Behind JFS,ReiserFS & XFS 8/2001
  - JFS tutorial 12/2000
  - LinuxWorld 10/2000
  - Linux Magazine 8/2000
  - Linux Gazette 7/2000
  - Byte 5/2000
  - Journal of Linux Technology 4/2000

JFS Features

Scalable 52-bit file system:

- **File size max 4 PB w/ 4k block size**
- **Max aggregate 4 PB w/4k block size**

Note: above values are limited by Linux I/O structures not being 64-bit in size.

2.4 Limits

- ►**Signed 32 bit 2^31 limit 1 TB max.**
- ►**2 TB limit is the max.**

2.5 Limits

- ►**16 TB limit caused by page cache**

JFS Features

Journaling of meta-data only

- Restarts after crash immediately
- Extensive use of B+tree's throughout JFS
- Extent-based allocation
- Unicode (UTF16)
- Built to scale. In memory and on-disk data structures are designed to scale without practical limits.
- Designed to operate on SMP hardware, with code optimized for at least an 4-way SMP machine

JFS Features

Performance:

- An extent is a sequence of contiguous aggregate blocks allocated to JFS object.
- JFS uses 24-bit value for the length of an extent
  - ► Extent range in size from 1 to 2(24) -1 blocks
  - ► Maximum extent is 512 * 2(24)-1 bytes (~8G)
  - ► Maximum extent is 4k * 2(24)-1 bytes (~64G)
    - – Note: these limits only apply to single extent; in no way limit the overall file size.
- Extent-based addressing structures
  - ► Produces compact, efficient mapping logical offsets within files to physical addresses on disk
  - ► B+tree populated with extent descriptors

JFS Features

Performance:

- B+tree use is extensive throughout JFS
  - ► File layout (inode containing the root of a B+tree which describes the extents containing user data)
  - ► Reading and writing extents
  - ► Traversal
  - ► Directory entries sorted by name

JFS Features

Variable block size (Not yet implemented)

- **Block sizes 512*, 1024*, 2048*, 4096**

Dynamic disk inode allocation

- **Allocate/free disk inodes as required**
- **Decouples disk inodes from fixed disk locations**

Directory organization

- **B+tree keyed on name**
- **Up to 8 entries may reside in B+tree root in inode (smaller directories are entirely within inode)**

JFS Features

Allocation Groups

- **Partitions the File System into regions**
- **Primary purpose of AGs is provide locality & parallelism within the FS**

JFS Features

Support for Sparse and Dense files (not yet)

- Sparse files reduce blocks written to disk
- Dense files disk allocation covers the complete file size (not yet)

Capability to increase the file system size

- LVM or EVMS and then remount the FS
  - LVM -> Logical Volume Manager
    - http://www.sistina.com/products_lvm_download.htm
  - EVMS -> Enterprise Volume Management System
    - http://sourceforge.net/projects/evms/
  - Support on-line re-sizing (1.0.21)
    - mount -o remount,resize /mount_point

JFS Features

Support for Snapshot

- Use LVM or EVMS
  - Setup the volume to use as the snapshot
  - Stop the File System operations (VFS operation)
  - Take the snapshot
  - Restart the File System operations (VFS operation)
  - Mount the snapshot volume
  - Create your backup using the snapshot volume
  - Remove the snapshot volume

JFS Features

Support for Extended Attributes (EA)

- Arbitrary name/value pairs that are associated with files or directories
- EA can be stored directly in the inode

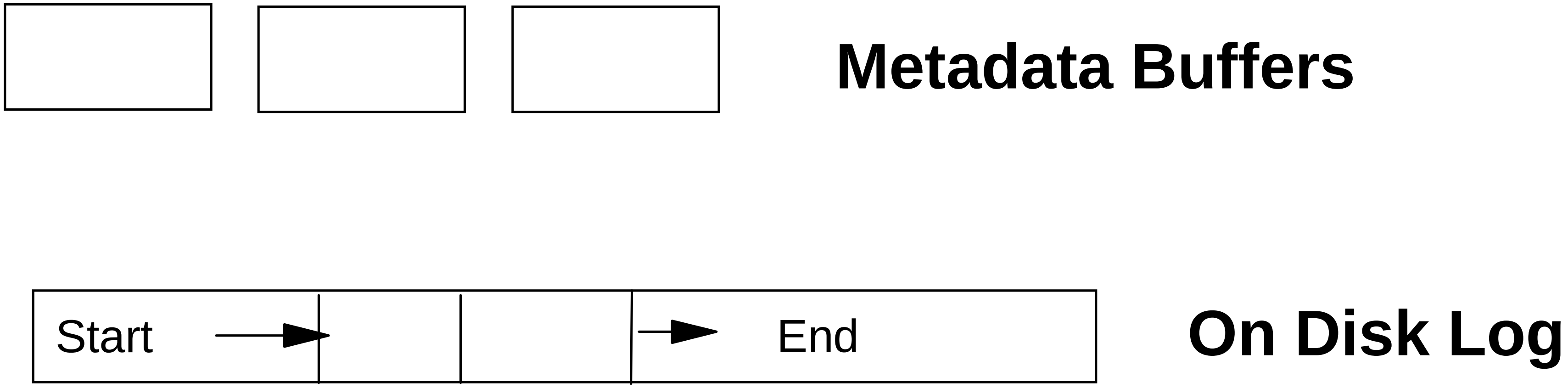
Support for Access Control Lists (ACLs)

- Support more fine-grained permissions
- Store ACLs as Extended Attributes

Extended Attributes and ACLs

- http://acl.bestbits.at/

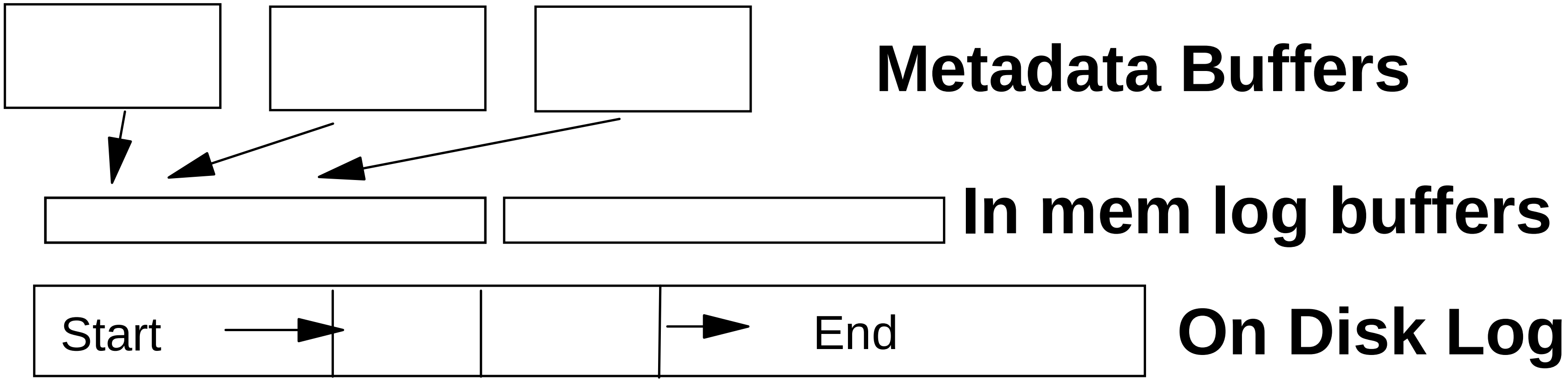
Journaling Basics



Reserve log space

Allocate transaction block, lock modified metadata

Journaling Basics



Metadata Buffers

In mem log buffers

Start

End

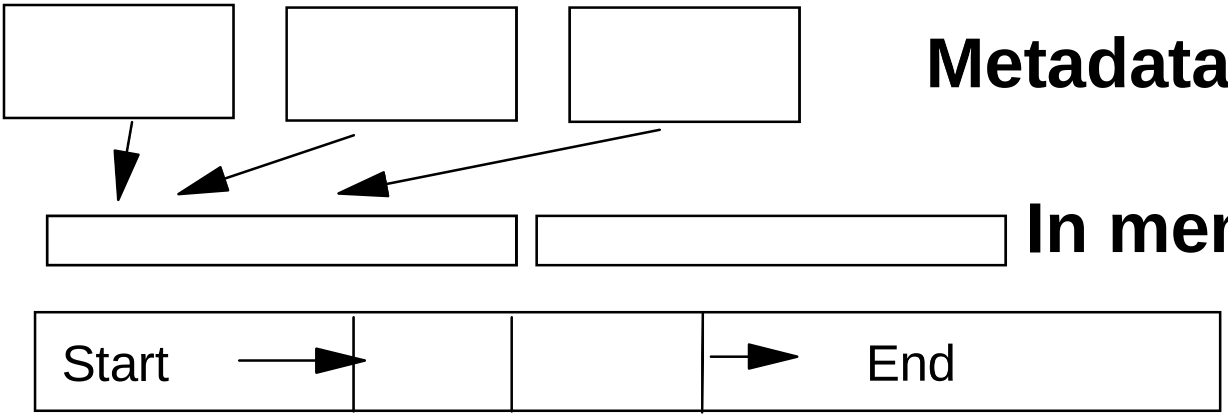
On Disk Log

Transaction Commit

Copy modified metadata into in memory log buffers

Pin buffers in memory and unlock

Journaling Basics



Metadata Buffers

In memory log buffers

Start

End

On Disk Log

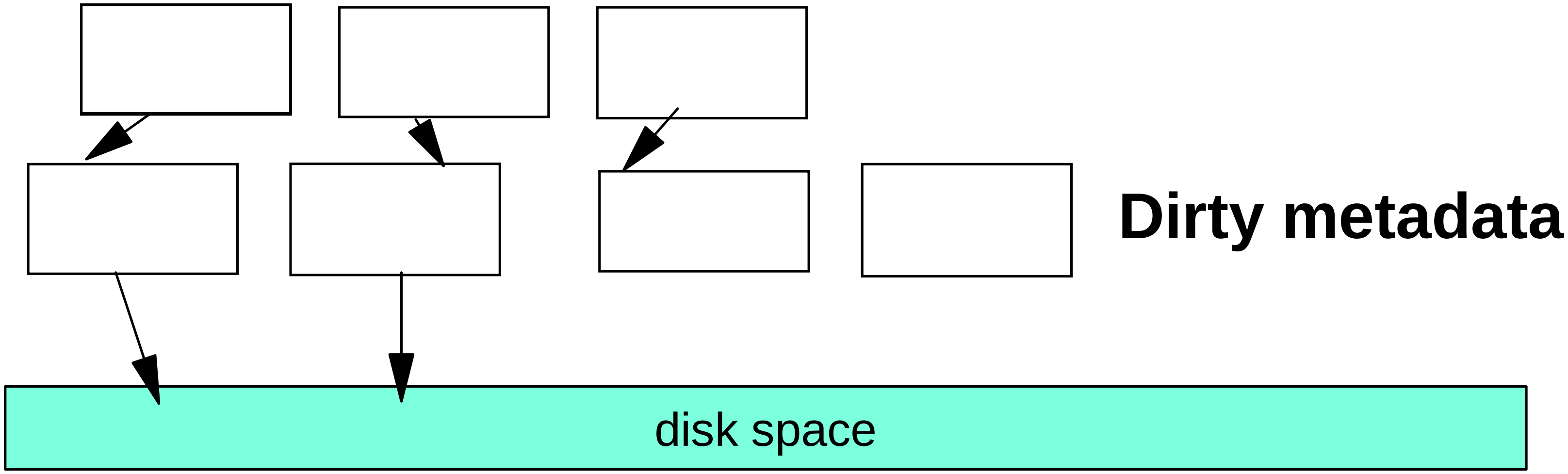
Write in memory log out to log device

Unlock metadata

Triggered by:

- log buffer full
- synchronous transaction (O_SYNC write)
- sync activity

Journaling Basics

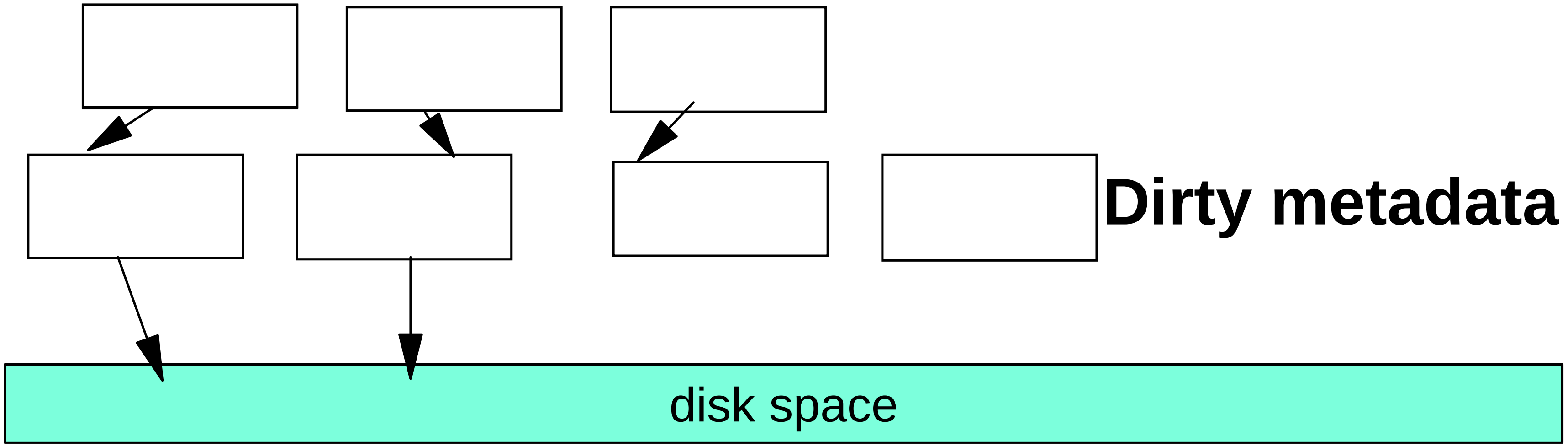


Write metadata out to the disk

Triggered by:

- Flush activity
- Memory pressure
- log space pressure

Journaling Basics



Metadata write completes

What operations are logged

Only meta-data changes:

- File creation (create)
- Linking (link)
- Making directory (mkdir)
- Making node (mknod)
- Removing file (unlink)
- Symbolic link (symlink)
- Create/modify/delete EA/ACL (setacl)
- Grow regular file
- Truncate regular file

Layout of Log

- Circular link list of transaction "block"
  - in memory
  - written to disk
    - location of log is found by superblock
- Log file
  - create by mkfs.jfs (internal or external)
    - Internal log size
    - default 0.4% of the aggregate size
    - maximum size 32M
    - External log size
    - maximum size 128M

Logging create example

Brief explanation of the create transaction flow:

```
    tid = txBegin(dip->i_sb, 0);

    tblk = tid_to_tblock(tid);
    tblk->xflag |= COMMIT_CREATE;
    tblk->ip = ip;
    iplist[0] = dip;
    iplist[1] = ip;

/*  work is done to create file */

    rc = txCommit(tid, 2, &iplist[0], 0);
    txEnd(tid);
```

Logredo

Started by fsck.jfs

Logredo

- Replay all transactions committed since the most recent sync point
- Superblock is read first
- Log replay is one pass over log, reading backwards from logend to first sync point rec.
- Inodes, index trees, and directory trees
- Inode Allocation Map processing
- Handle 6 different logredo records
  - (LOG_COMMIT, LOG_MOUNT, LOG_SYNCPT, LOG_REDOPAGE, LOG_NOREDOINOEXT, LOG_UPDATEMAP)

Logredo

All records have been handled:

- Flush redo buffers
- If needed rebuild freelists
- Finalize file system
  - Update allocation map
  - Update superblock
- Finalize the log
  - Clear active list

Where is JFS today?

Announced & Shipped 2/2/2000 at LinuxWorld NYC

- What has been completed
  - ►65 code drops so far
  - ►JFS patch files to support multiple levels of the kernel (2.4.3-2.4.x), kernel & utility tarballs
  - ►Completely independent of any kernel changes (easy integration path)
  - ►Release 1.0.0 (production) 6/2001
  - ►Accepted by Alan Cox 2.4.18pre9-ac4 (2/14/02)
  - ►Accepted by Linus for 2.5.6-pre2 (2/28/02)
  - ►Accepted by Marcelo Tosatti 2-4.20-pre4(8/20/02)
  - ►Release 1.1.2 3/25/2003

JFS for Linux

Utility area:

jfs_mkfs -> Format

jfs_fsck -> Check and repair file system

- Replays the log

jfs_defrag * -> Defragmentation of file system

jfs_tune -> Configuration of the FS

jfs_debugfs -> View and modify JFS on-disk structures

jfs_logdump -> Service-only dumps contents of journal

jfs_fscklog -> Service-only extract/display log from fsck

Distros

Distributions shipping JFS

- Turbolinux 7.0 Workstation (8/01) was 1st
- Mandrake Linux 8.1, 8.2, 9.0
- SuSE Linux 7.3 , 8.0, 8.1, SLES 8.0
- Red Hat 7.3, 8.0, 9.0
- Slackware 8.1
- United Linux 1.0
- others......

JFS WIP

Near term:

- Performance improvements in FS
- Adding support for external log to be shared by more than one FS
- Adding defragmentation of FS
- Mount option for backup programs to restore without journaling

Longer term:

- Quota
- Data Management API (DMAPI)

File System & File Sizes

Filesystems limits on 32-bit architectures

	ReiserFS	Ext3	XFS	JFS
Max. files	4G	4G	4G	4G
Subdirs/dir	65K	32K	4G	65K
Max. filesize	16TB*	2TB	16TB*	16TB*
Max. FS size	16TB*	16TB	16TB*	16TB*

Notes:
Block device limit in 2.4 was 2TB
Block device limit in 2.5 has been raised
* Issue is page cache has limit 16TB

Journaling File Systems

Ext3 patches

on sourceforge as the ext3 module in the "gkernel" project

http://www.zipworld.com.au/~akpm/linux/ext3/

ReiserFS web page

http://www.namesys.com

XFS web page

http://oss.sgi.com/projects/xfs/

JFS web page

http://oss.software.ibm.com/jfs

Journaling File Systems Articles

"Journaled Filesystem" by Steve Best, David Gordon, and Ibrahim Haddad, Linux Journal January 2003

"Journaling File System" by Steve Best, Linux Magazine 10/2002

- http://www.linux-mag.com/2002-10/jfs_01.html

"Journaling Filesystems" by Moshe Bar, Linux Magazine 8/2000

- http://www.linux-mag.com/2000-08/journaling_01.html

"Journal File Systems" by Juan I. Santos Florido, Linux Gazette 7/2000

- http://www.linuxgazette.com/issue55/florido.html

"Journaling File Systems For Linux" by Moshe Bar, BYTE.com 5/2000

- http://www.byte.com/documents/s=365/byt20000524s0001/

Credits

Thanks to Steve Best for providing these presentation graphics.

Questions..........