Journaled File System (JFS) for Linux
UT, Texas
4/08/2002

Steve Best
sbest@us.ibm.com
Linux Technology Center - JFS for Linux
IBM Austin

http://oss.software.ibm.com/developer/opensource/jfs/project/pub/jfs040802.pdf

Overview of Talk

- Linux Filesystems
- Features of JFS
  - Why log/journal
  - Performance
- 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

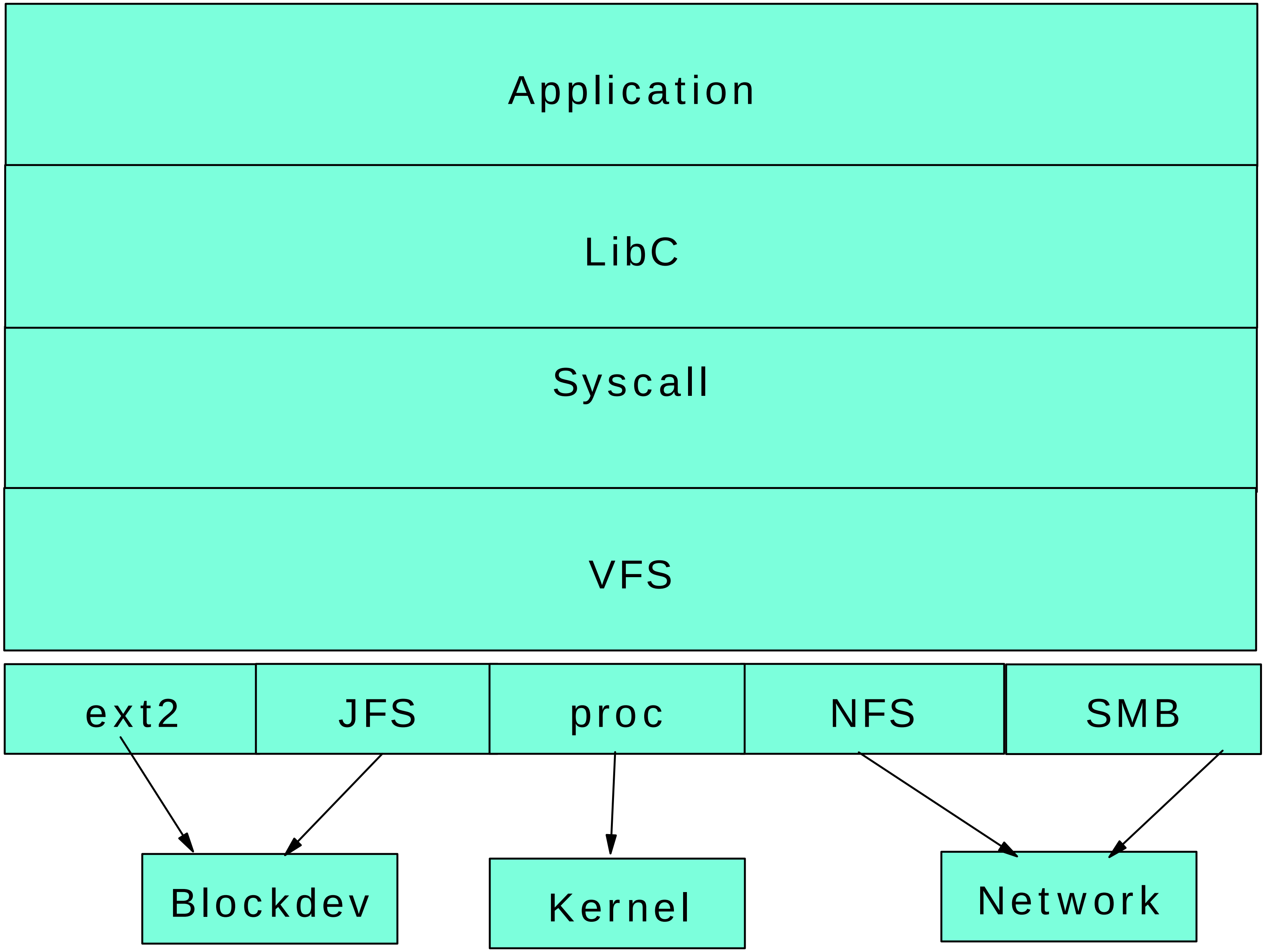
Linux Filesystems

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

Virtual Filesystem Layer

- abstraction layer above file systems
- Filesystems modular, except boot fs
  - ►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



Application

LibC

Syscall

VFS

ext2

JFS

proc

NFS

SMB

Blockdev

Kernel

Network

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.

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

JFS Team members

IBM:

- Barry Arndt (barndt@us.ibm.com)
- Steve Best (sbest@us.ibm.com)
- Dave Kleikamp (shaggy@us.ibm.com)

Community:

- Christoph Hellwig (hch@caldera.de)
- ....others

Why use JFS ?

- Highly Scalable 64 bit file system:
  - scalable from very 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 < 1 sec

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 64-bit file system:

- **File size max 512 terabytes w/ 512 block size**
- **File size max 4 petabytes w/ 4K block size**
- **Max aggregate 4 PB w/512 block size**
- **Max aggregate 32 PB w/4k block size**

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

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

JFS Features

Journaling of meta-data only

- Restarts after crash immediately
- Design included journaling from the start
- 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
  - Directory Slot free list

JFS Features

Variable block size

- 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 (methods)

- 1st method stores up to 8 entries directly into directory's inode (used for small directories)
- 2nd method B+tree keyed on name (used for larger directories)

JFS Features

Support for Sparse and Dense files

- Sparse files reduce blocks written to disk
- Dense files disk allocation covers the complete file size

Capability to increase the file system size *

- LVM or EVMS and then use file system utility
  - LVM -> Logical Volume Manager
    - http://www.sistina.com/products_lvm_download.htm
  - EVMS ->Enterprise Volume Management System
    - http://sourceforge.net/projects/evms/

Support for defragmentation of the FS *

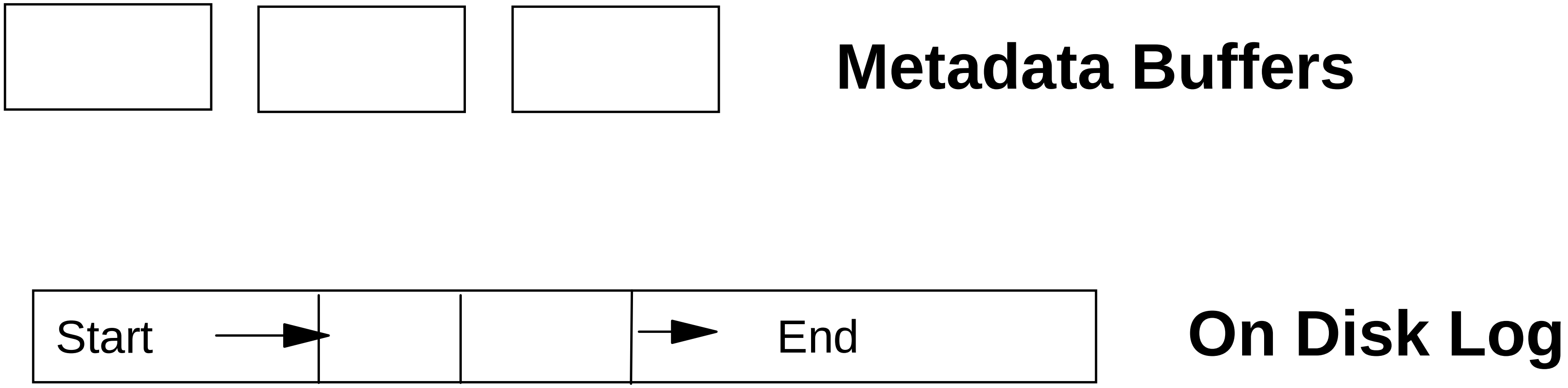
- Defragmentation utility

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

Journaling Basics



Metadata Buffers

Start

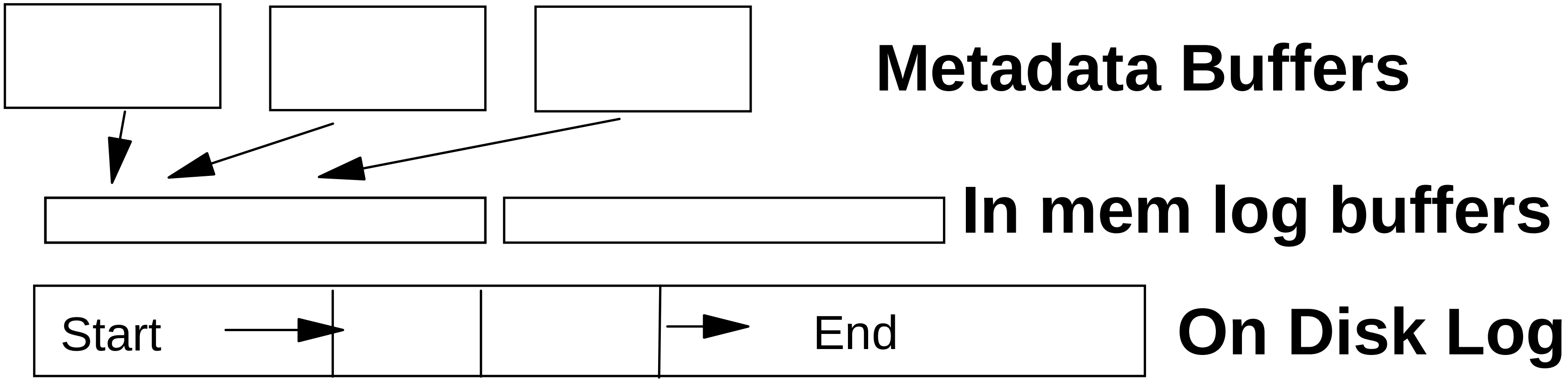
End

On Disk Log

Reserve log space

Allocate transaction block, lock modify metadata

Journaling Basics



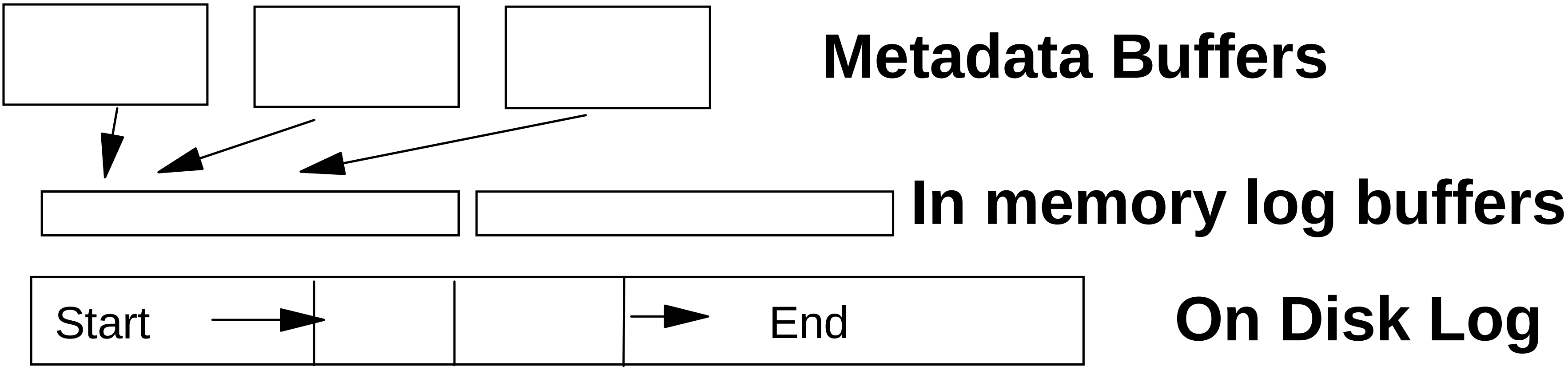
Transaction Commit

Copy modified metadata into in memory log buffers

Pin buffers in memory and unlock

Transaction is complete

Journaling Basics



Metadata Buffers

In memory log buffers

Start

End

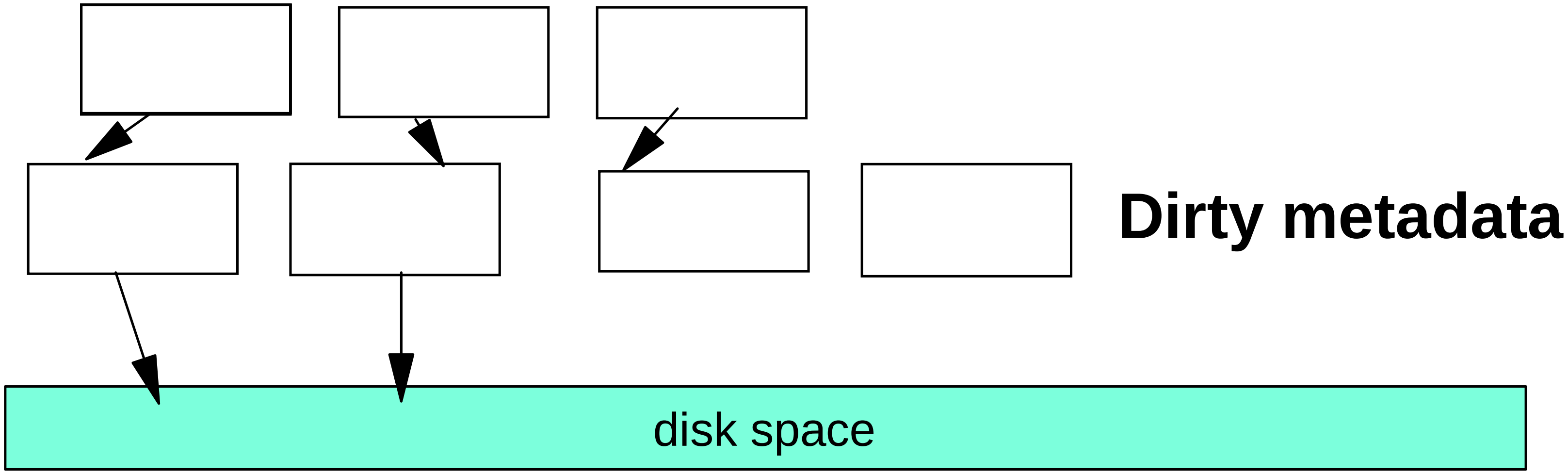
On Disk Log

Write in memory log out to log device

Triggered by:

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

Journaling Basics

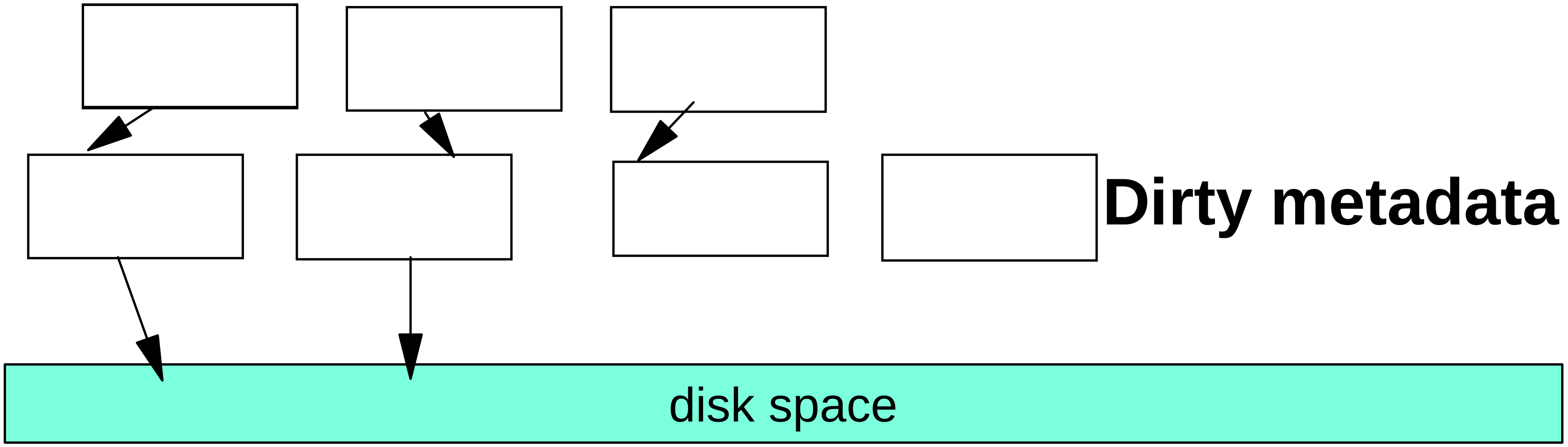


Write metadata out to the disk

Triggered by:

- Flush activity
- Memory pressure
- log space pressure

Journaling Basics



Metadata write completes

- Removes metadata locks

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)
- Set ACL (setacl)
- Truncate regular file

Layout of Log

- Circular link list of transaction "block"
  - in memory
  - written to disk
    - location of log is found by superblock
- Size of Log file
  - create by mkfs.jfs
    - default 0.4% of the aggregate size
    - maximum size 32M
    - 15G -> defaults 8192 aggregate blocks

Logging create example

Brief explanation of the create transaction flow:

```
  txBegin(dip->i_ipmnt, &tid, 0);

   tblk = &TxBlock[tid];
   tblk->xflag |= COMMIT_CREATE;
   tblk->ip = 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 synch point
- Superblock is read first
- Log replay is one pass over log, reading backwards from logend to first synch 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_UPATEMP)

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
  - ►55 code drops so far
  - ►JFS patch files to support multi-levels of the kernel (2.4.3-2.4.x) kernel patch & utility patch file
  - ►Completely independent of any kernel changes (easy integration path)
  - ►Beta 1 12/2000
  - ►Beta 2 3/2001
  - ►Beta 3 4/2001
  - ►Release 1.0.0 (production) 6/2001
  - ►Accepted by Alan Cox 2.4.18pre9-ac4 (2/14/02)
  - ►Aceepted by Linus for 2.5.6-pre2 (2/28/02)
  - ►Release 1.0.17 4/2/2002

JFS for Linux

Utility area:

mkfs	-> Format
logredo	-> Replays the log
fsck.jfs	-> Check and repair file system
defrag *	-> Defragmentation of file system
extendfs *	-> Extend the file system
xchklog	-> Service-only extract log from fsck
xpeek	-> Peek and change JFS on-disk structures
xchkdmp	-> Service-only displays file created by
logdump	-> Service-only dumps contents of log file

Distros

Distributions shipping JFS

- Turbolinux 7.0 Workstation (8/01)
- Mandrake Linux 8.1 (9/01)
- SuSE Linux 7.3 Intel (10/01)
- SuSE Linux 7.3 PowerPC (11/01)
- SuSE Linux Enterprise Server 7 for zSeries (11/01)
- Turbolinux 7.0 Server(12/01)
- SuSE Linux Enterprise Server 7 for IBM eServer iSeries and pSeries (1/02)

Distros

Distributions shipping JFS

- Mandrake Linux 8.2 (3/02)
- SuSE Linux 8.0 IA32 (4/02)

Distributions in process of shipping JFS

- Debian (Woody release)

JFS WIP

Near term:

- Adding growing the FS support
- Adding defragmentation of FS
- Adding support for釆 externel log

Longer term:

- Quota
- ACL
- Extended Attributes

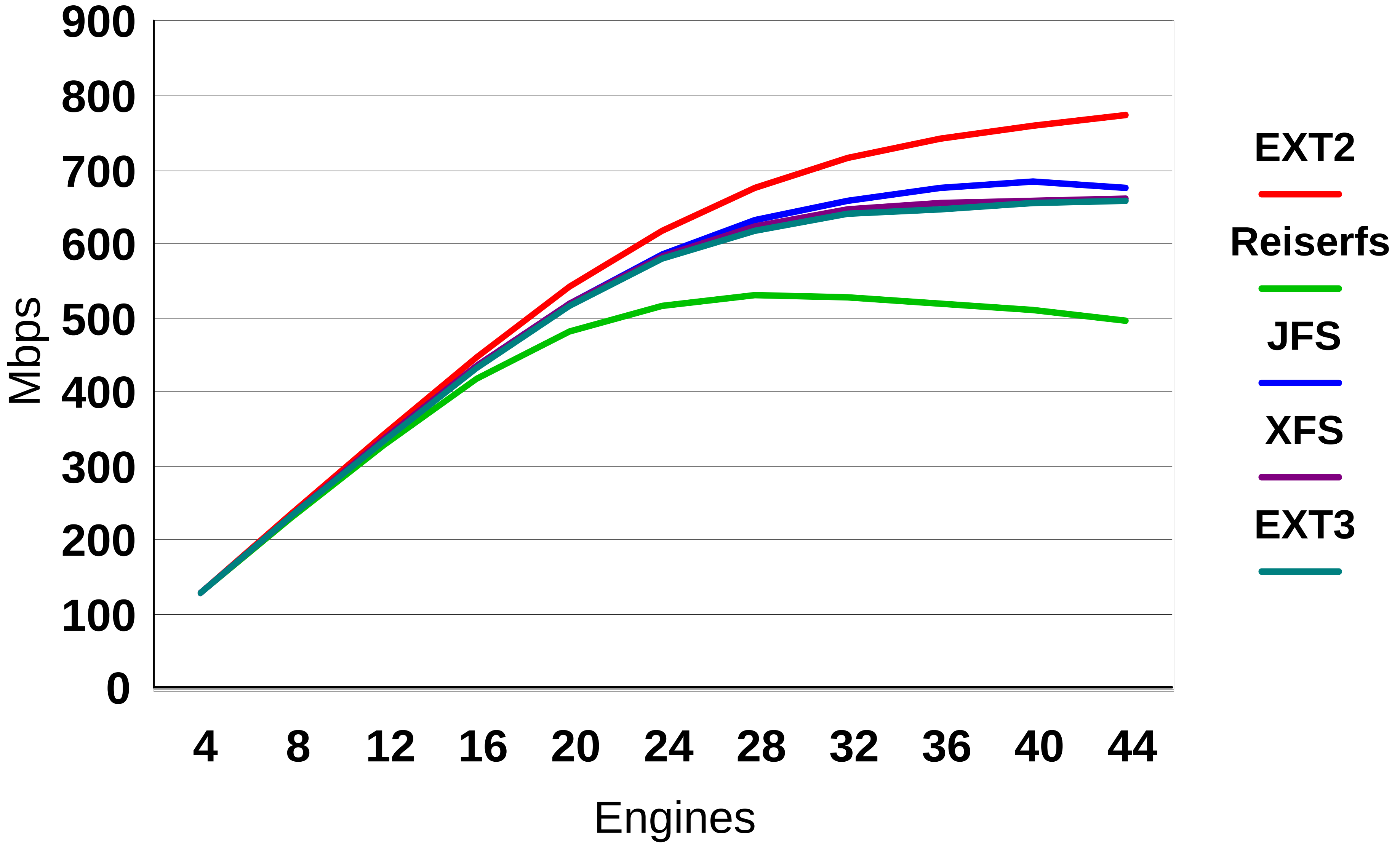
File System & File Sizes

	ext3	ReiserFS	XFS	JFS
Block sizes	4KB	4KB	4KB	4KB
Max. FS size	16TB	17592 GB	18 * thousand petabytes	32 petabytes
Max file size	2TB	1152921504 GB	9 * thousand petabytes	4 petabytes

Netbench Throughput: Filesystem Comparison

Linux 2.4.7, Samba 2.2.0, NetBench 7.0.1

4 x 700 MHz Intel Xeon, 1 MB L2, 4 x 1 Gbps Ethernet, RAID 1E



Journaling File Systems Benchmarks

Journal Filesystem Comparison on Netbench was published by Andrew Theurer on August 27, 2001

http://marc.theaimsgroup.com/?l=linux-kernel&m=99892482710604&w=2

Linux 2.4.x Journaling File Systems: Performance Data using IOzone was published by Randy Dunlap on August 29, 2001

http://www.osdlab.org/reports/journal_fs/

OpenBench Labs tests ext3FS, JFS, and ReiserFS on a fast RAID appliance February 2002 issue of Open Magazine

http://www.open-mag.com/42244203327.htm

2.5 Kernel changes FS I/O areas ....so far

- Block IO (bio) layer needs attention
  - Performance problems
    - Requests are broken down to individual blocks
    - Most of them start out larger
    - Elevator must merge them back together
    - Processing overhead, many buffer heads
  - Built-in limitations
    - global arrays
- Rewrite of block IO (bio) layer (Jens Axboe)
  - included in 2.5.1-2
- http://lwn.net/2001/1206/kernel.php3

Journaling File Systems

Ext3 patches

2.2.x series

ftp://ftp.us.kernel.org/pub/linux/kernel/people/sct/ext3/

2.4.x series (2.4.15)

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

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

ReiserFS web page

(2.4.1)

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

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

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

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

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

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

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

"JFS for Linux" by Joe Bar, LinuxWorld 10/2000

- http://www.linuxworld.com/linuxworld/lw-2000-10/lw-10-vcontrol_1.html

Interviews with developers of JFS, ReiserFS and XFS on OSNews

- http://www.osnews.com/story.php?news_id=69

JFS Project urls

JFS Web page

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

JFS Overview white paper

- http://www-4.ibm.com/software/developer/library/jfs.html

JFS Layout white paper

- http://www-4.ibm.com/software/developer/library/jfslayout/index.html

JFS Log white paper

- http://oss.software.ibm.com/jfs/project/pub/jfslog/jfslog.pdf

JFS Root Boot Howto

- http://oss.software.ibm.com//jfs/project/pub/jfsroot.html

JFS Mailing list

- http://oss.software.ibm.com/pipermail/jfs-discussion/

Questions..........