Introduction to File Systems

Stewart Smith
(stewart@flamingspork.com)

Also known as:
Vice President, Linux Australia (stewart@linux.org.au)

Software Engineer, MySQL AB (stewart@mysql.com)

The importance of speed

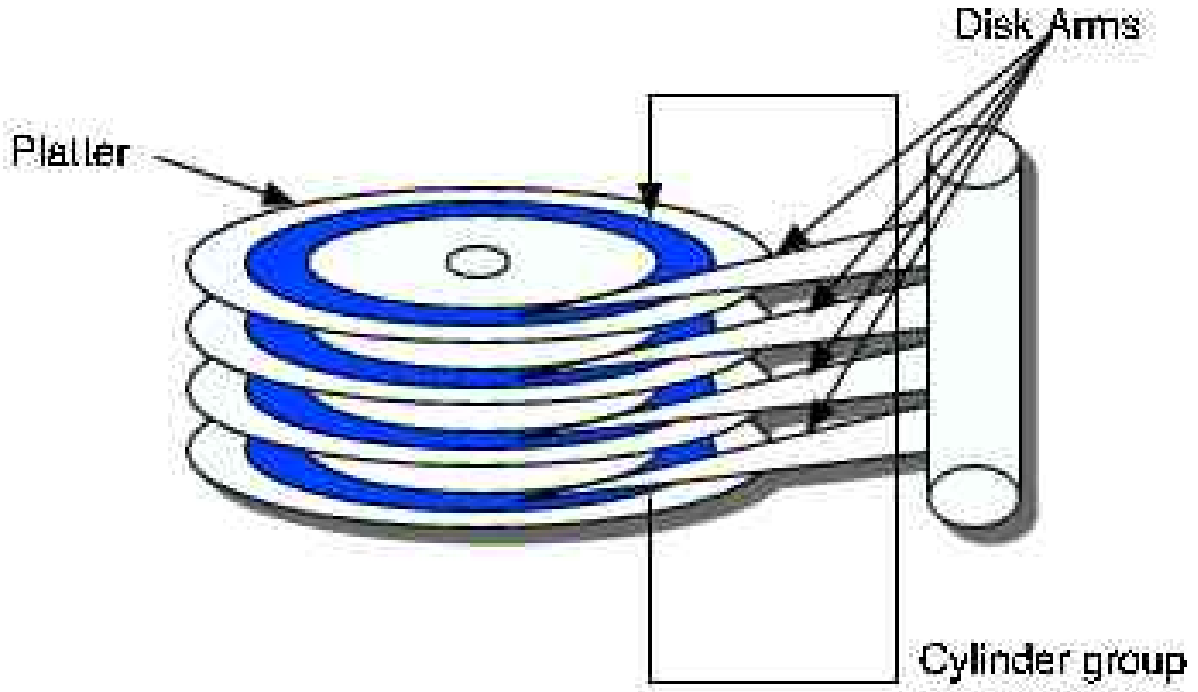
- How fast is a CPU?
- How fast is CPU cache?
- How fast is RAM?
- How fast is disk?

Relative Disk Speed

- Laptop: 24MB/sec
- Desktop: 50MB/sec

Most expensive operation?

- In an entire computer, the most expensive operation is:
  - A disk seek



Reads and writes

- Since seeking is expensive, when we do seek
  - we want to read (or write) as much data as possible.
- Big block sizes are good for this
  - BSD FFS got a lot of its speed from this.

Got the physics?

- Good...
  - now for more fancy stuff

Many File Systems to choose from

- FFS, LFS, UFS, FAT(12,16,32), NTFS, ext, ext2e ext3, reiserfs, reiser4, xfs, jfs, HFS, HFS+, BeFS, WinFS, LFFS

- Similar ideas in other systems:
  - Databases
  - Data files

The i-node

- An i-node describes a file
- A directory is a special case of a file
  - Contains a list of name,i-node number pairs
- Superblock contains the i-node number of the root directory

Data in an i-node

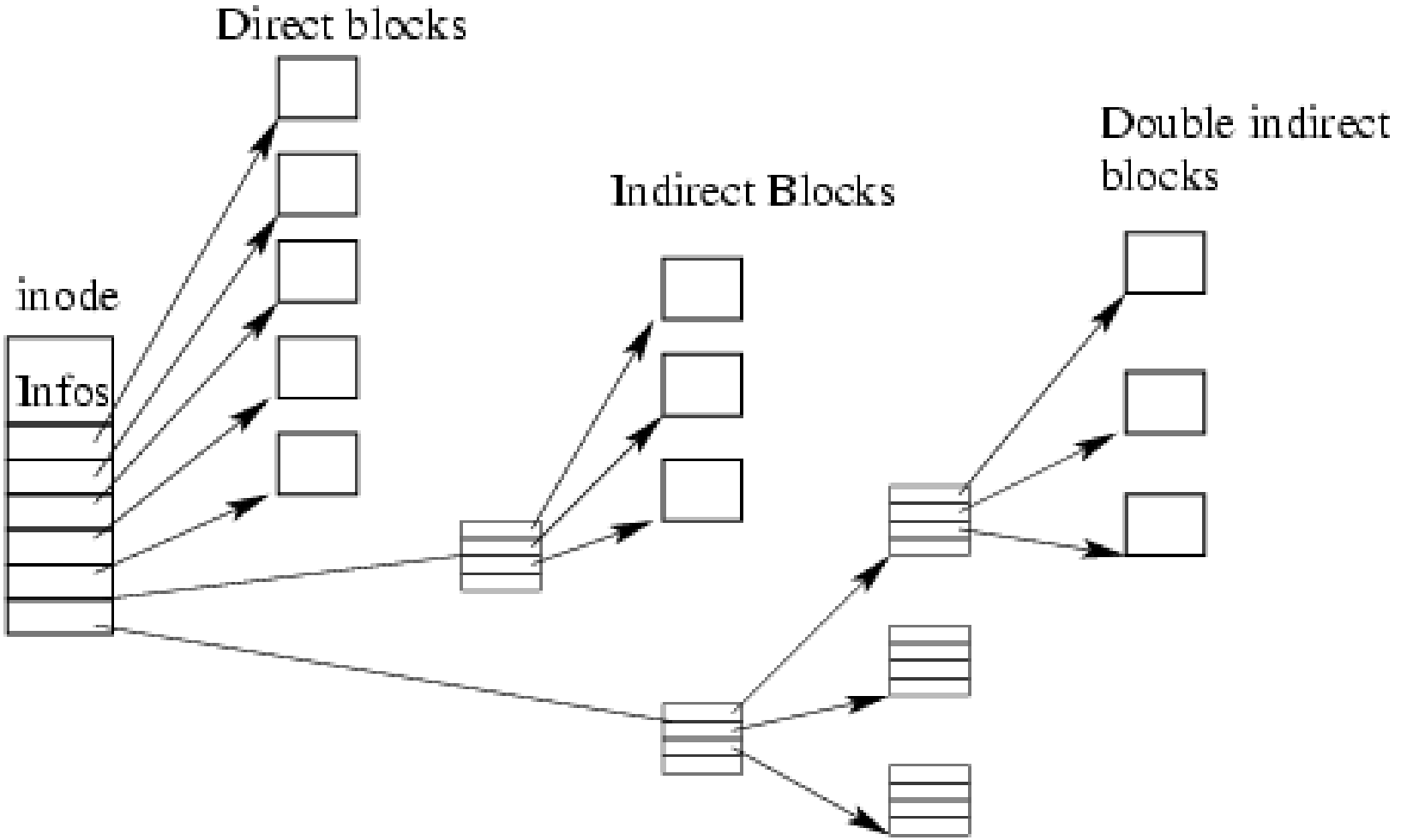
- Mode (chmod)
- owner, group
- timestamps
- size
- some directions to find out how to get the content of the inode
- extended attributes information

BSD FFS

- Where we learnt how to run
- Featured:
  - Larger block sizes (4096 bytes)
  - use of cylinder groups to exploit the physical properties of disks
  - improved reliability through careful ordering of meta-data writes.
- Paper published in 1984

Block Addressing

- Both FFS and ext2 do this:



On disk format

- Super block
- Cylinder groups
  - Redundant copy of super block
  - block bitmap
  - inode table
  - data area

/unix lookup

- Read superblock
- seek to location of root i-node (and read)
- seek to location of root i-node data blocks (and read)
- linear search for 'unix'
- seek to location of /unix i-node (and read)
- seek to location of /unix data blocks (and read)

Consistency (Reliability)

- FFS was synchronous
  - slow
- ext2 wasn't
  - fast, but easier to loose data
- Soft Updates fixed it for FreeBSD
  - carefully ordered meta data writes
- ext3 fixed it for ext2
  - addition of meta-data journalling

Downsides to reliability

- Soft Updates
  - still needs background fsck to reclaim lost disk blocks
- Journaling (ext3)
  - normally a performance penalty
    - although smart ordering of writes can increase performance

Downsides to FFS/ext[23]

- Each file (on average) wastes 0.5 disk blocks
  - really only ext2. FFS splits up
- Seek intensive
- Number of inodes is decided at mkfs time!
- Volume resizes are IO intensive
- Sucky performance on large files

Performance improvements

- Other people have solved some problems.
- Dynamically allocated inodes
  - e.g. XFS allocates inodes as you need them (in chunks)
- Put i-nodes and data together
- tail packing of files
  - reiserfs (esp reiser4) will pack small files into a single block.

Block Addressing

- Extents!
  - From block n, m blocks belong to this file
- XFS
  - store extents in the inode
  - if file has lots of extents, B+Tree
- reiser4
  - extents also

Block Allocation

- Extending a file
  - ideally you just add blocks to the end
- Searching a block bitmap is a bit tricky
- XFS
  - two B+Trees of free extents
    - Ordered by start block
    - Ordered by size
- Pre-allocation

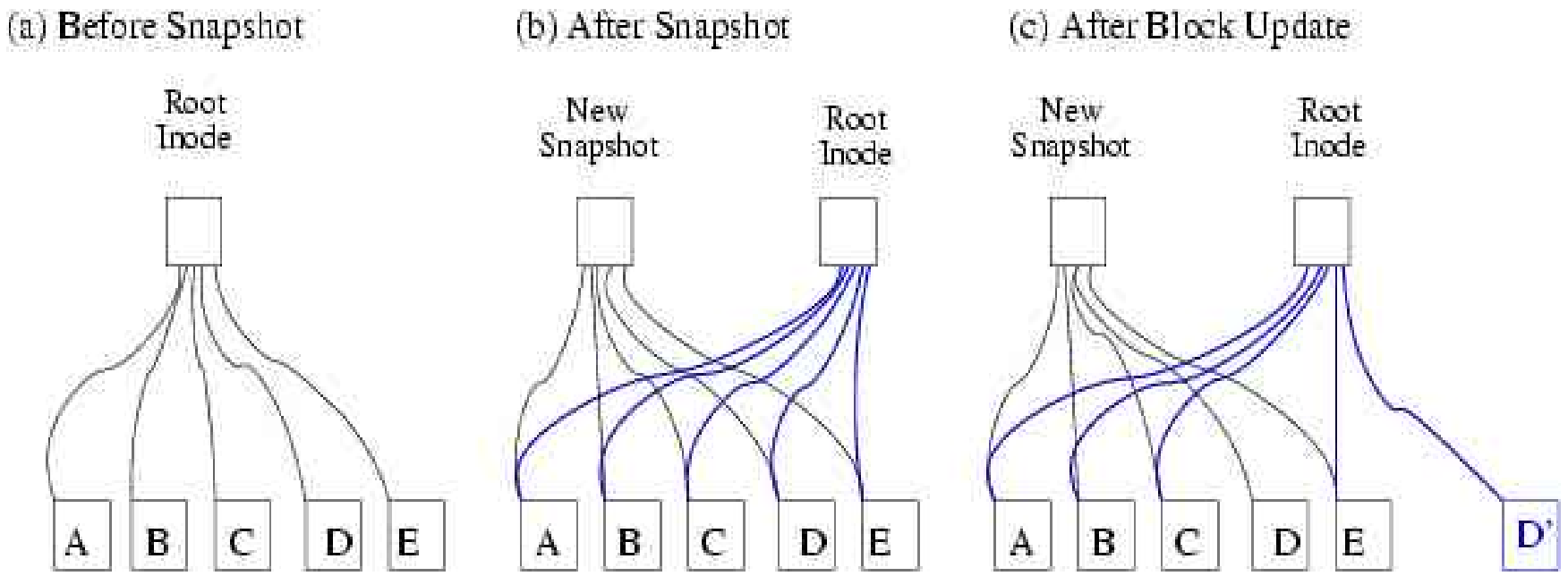
Directories

- XFS
  - For directories with many items, index them!
- ext3 htrees help
  - but not perfect

Extended Attributes

- reiserfs doesn't do them
  - Hans Reiser thinks the API sucks
    - he's right, but....
- ext3 does them
  - in a different block than the inode
- XFS does them
  - in the i-node (if they fit)

WAFL Snapshots/atomicity



So what should you use?

- Flame retardant underwear