ReiserFS

Hans Reiser

http://www.namesys.com/content_table.html

Filesystem impl. problems

- Slow file system recovery process
- Inefficient handling of disk space

- ReiserFS hypothesis:
  - Existing FS implementations (based on inodes and blocks) are not efficient enough to fix the above
  - Existing implementations have chosen the *easy route* to file system development

ReiserFS solution to FS impl. problems

- Features
  - Fast journaling
  - Based on balanced trees
  - More space efficient
- Necessary evil
  - Increased code complexity (ext2fs 6K LOC vs reiserFS 30K LOC)

Revisiting a previous question

" *The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks….The real disadvantage of a linear list of directory entries is the linear search to find a file. Directory information is used frequently, and a slow implementation of access to it would be noticed by users. In fact, many operating systems implement a software cache to store the most recently used directory information. …*

…However, the requirement that the list must be kept sorted may complicate creating and deleting files, since we may have to move substantial amounts of directory information to maintain a sorted directory. A more sophisticated tree data structure, such as a ***B-tree****, might help here."*

(Emphasis added)

Silberschatz, Abraham and Galvin, Peter. 1998. Operating System Concepts. Fifth Edition. Addison-Wesley. Reading, MA.

B-Trees

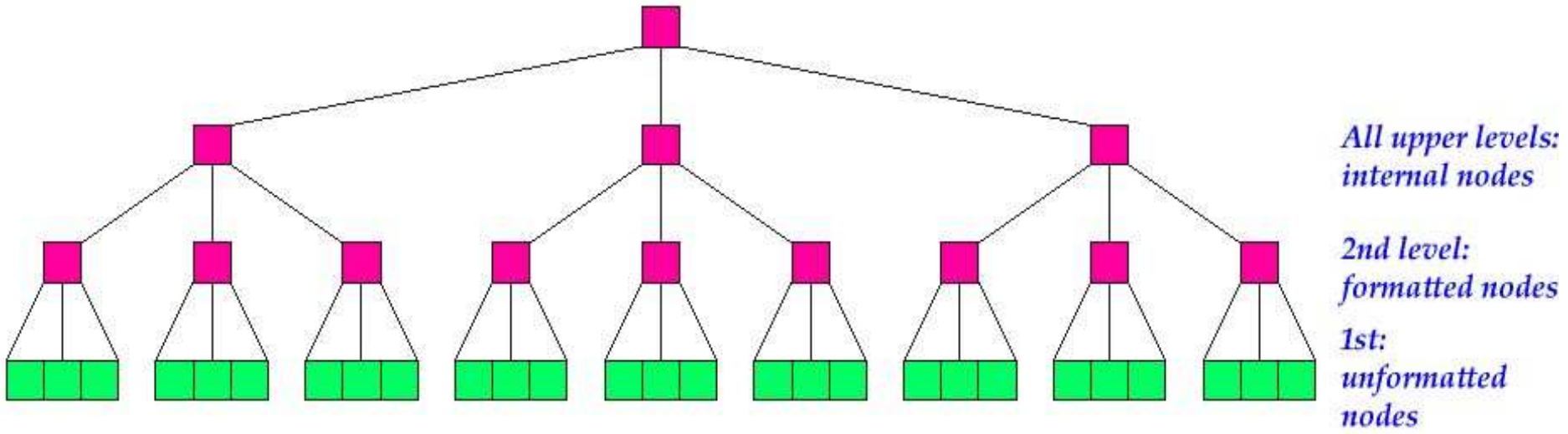
- Used extensively in Database systems.
  - Minimize the avg. number of disk accesses.
  - "to optimize the reference locality and space efficient packing of objects"
- Definition:
  - The root is either a leaf or has between 2 and m children
  - All nonleaf nodes (except the root) have between ceiling(m/2) and m children
  - All leaves are at the same depth

Source:

Weiss, Mark A. 1993. Data Structures and Algorithm Analysis in C. Benjamin/Cummings Publishing Company, Inc. Redwood City, CA.

ReiserFS and B-trees

- Visual

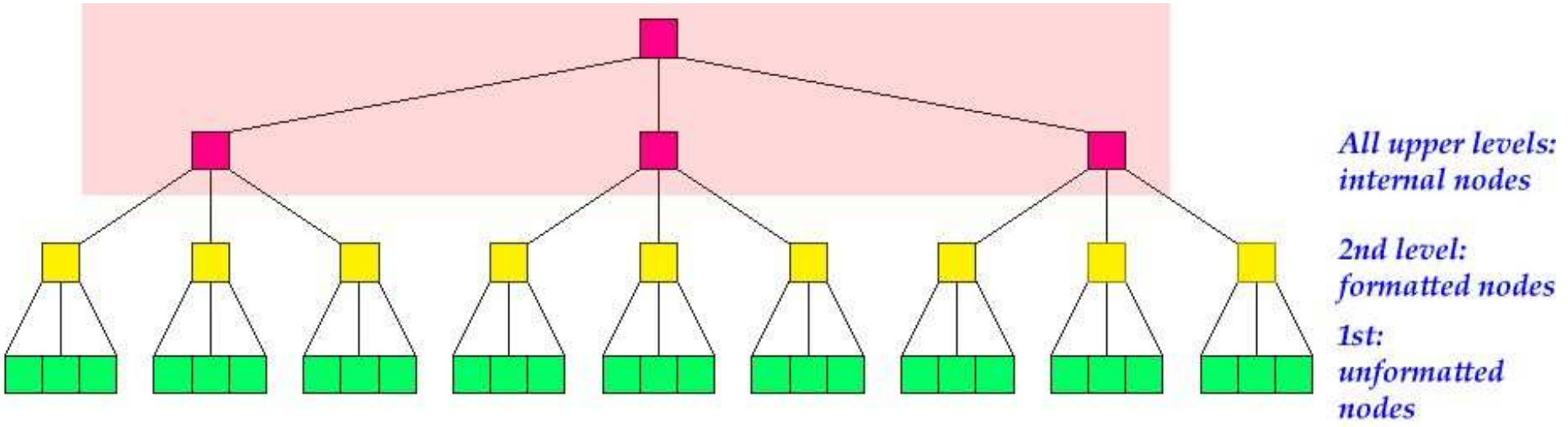


Adapted from: Reiser, Hans. Reiser4 Draft Document. http://www.namesys.com/v4/v4.html

ReiserFS and B-trees

- The tree is *in the disk blocks*
- In Reiserfs, keys are used in place of inode #'s
  - Lookup(Key) = internal node location

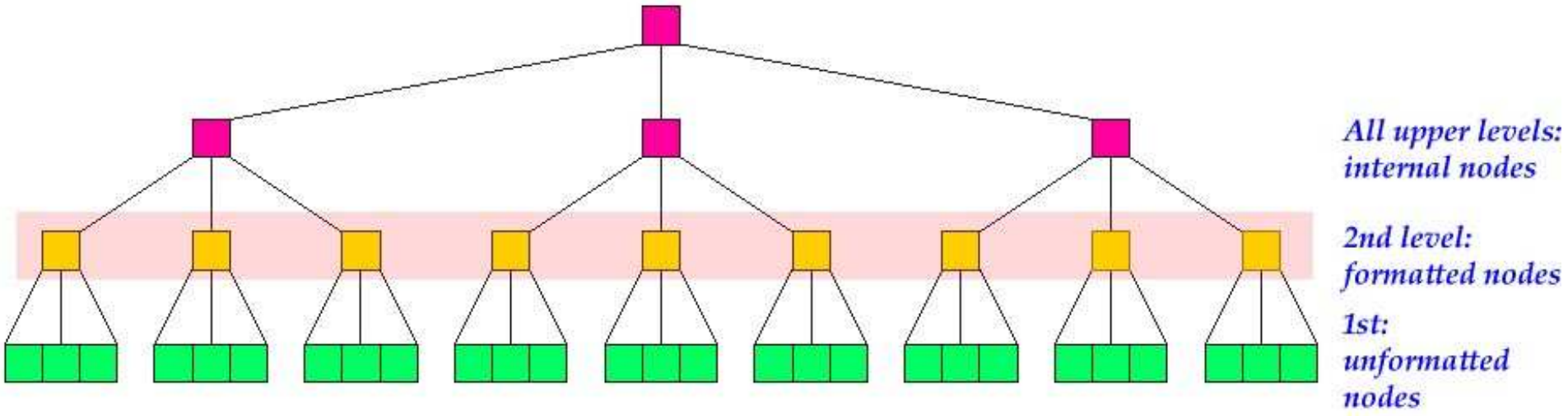
Internal Nodes - Visual



What are internal nodes?

- Exist to determine which formatted node contains the item corresponding to a key
- Used to drive the search through the tree

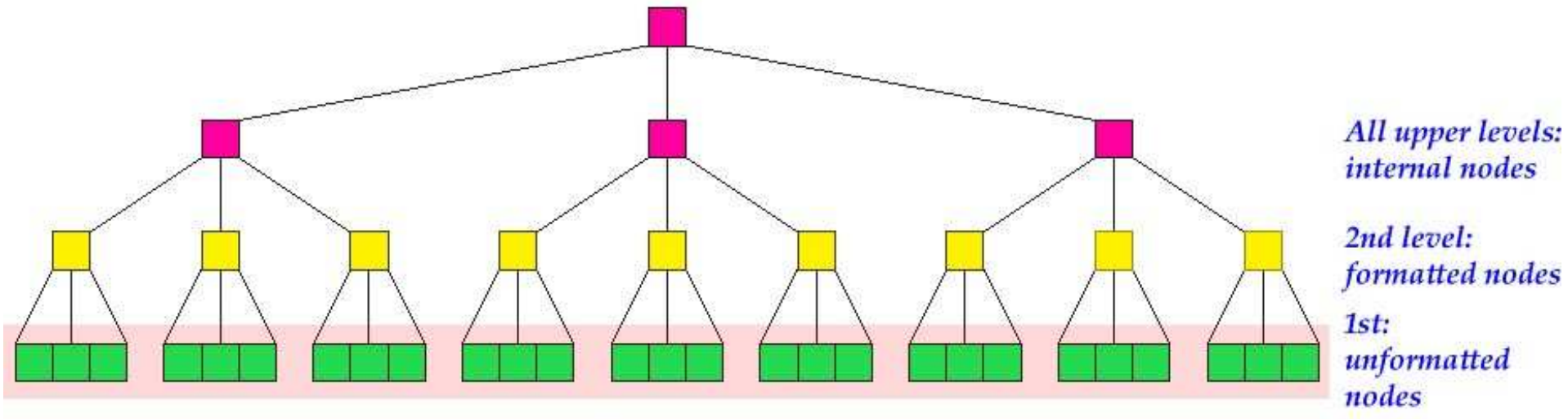
Formatted Nodes - Visual



What are formatted nodes?

- Contains *items*, with every item having a unique key
- Types of items:
  - Direct – contain the tails of files
  - Indirect – pointers to unformatted nodes
  - Directory – contain the key of the first directory entry in the item, followed by a number of directory entries
  - Stat data – optional analysis metric data

Unformatted Nodes



What are unformatted nodes?

- Size = FS_block_size
- *Do not* contain keys
- Do contain the body of big files
  - How big is "big"?
  - Anything over X bytes, where X=block size – 112 bytes. For example, using blocks of size 4KB (4096 bytes), this is 3984 bytes

How are files handled?

- What is a file? From the paper:

"A file consists of a set of indirect items followed by a set of up to two direct items, with the existence of two direct items representing the case when a tail is split across two nodes. If a tail is larger than the maximum size of a file that can fit into a formatted node but is smaller than the unformatted node size (4K), then it is stored in an unformatted node, and a pointer to it plus a count of the space used is stored in an indirect item."

How are files handled? (cont.)

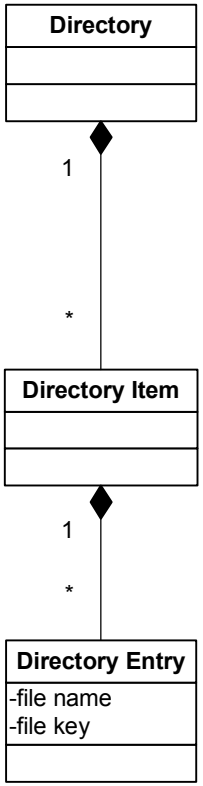
- Great, but what does that mean?
  - We know that files do not exist in internal nodes, and unformatted nodes do not contain keys…
  - So files must "exist" in the formatted nodes.
    - "exist" indicates logical organization, as opposed to physical representation of bits
    - Recall that formatted nodes contain items
    - Tails are the last part of the file (last file_size % FS_block_size)

How are files handled? (cont.)

- Example:
  - Let's say we have a 27K file, FS_block_size=4K
    - Tail = 27648 % 4096 = 3072 bytes
    - Non-tail = 24576 bytes = 24 KB

- Recall that Formatted Nodes/indirect items contain pointers to unformatted nodes.
  - 6 indirect items (24 KB / 4 KB)
  - 1 direct item (Direct length = 4096 -112 = 3984). This is stored directly in the formatted node.

How are directories handled?



There is never more than one item of the same item type from the same object stored in a single node.

Space Efficiency

- Let's write a 3KB block of data, assuming FS_block_size=4KB
- ReiserFS:
  - As 4096 – 112 = 3984, and 3KB = 3072, then the entire block of data will be written into the direct item of a formatted node. 912 bytes of space in the direct node would remain.

ReiserFS Journaling

- ?? While it's mentioned in the "features" section at the beginning of the paper, it is not mentioned in the body of the paper.

- "Unformatted nodes make filesystem recovery faster and less robust, in that one reads their indirect item rather than to insert them into the recovered tree, and one cannot read them to confirm that their contents are from the file that an indirect items says they are from. In this they make ReiserFS similar to an inode-based system without logging."

Filesystem comparisons

All values are in seconds lower is better

fs	bigdir	cp	Rm	total
Reiserfs	40.03	77.75	10.86	81.06
Reiser4	33.51	32.9	17.45	82.67
Ext3	38.79	91.57	26.21	54.25
ext2	32.78	37.28	16.28	38.98

Location: NameSys http://www.namesys.com/benchmarks/v4marks.html

Source: Gram Miner, bench.scm script, http://http://epoxy.mrs.umn.edu/~minerg/fstests/results.html)

Filesystem comparisons (cont)

	ReiserFS (3.6)	Ext2fs
Max file size	2^60	2 GB
Max file name	Block_size – 64 4KB – 64 = 4032 bytes	255 characters (1012 c)*
Max filesystem size	2^32 (4K) blocks	4 TB

Regarding the Paper: Namespaces

- The author spends a significant amount of time advocating the use and "expressive power" of namespaces.
- Not exactly a new idea.
  - Java packages => C++ namespaces (Stroustrup 1997)
  - XML namespaces
  - Unix tools idea?