

## Re: 20TB Storage System (fsck????)

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-09/0046.html>

---

**From:** Terry Lambert ([tlambert2\\_at\\_mindspring.com](mailto:tlambert2_at_mindspring.com))

**Date:** 09/05/03

Date: Fri, 05 Sep 2003 00:28:31 -0700

To: Geoff Buckingham <[geoffb@chuggalug.clues.com](mailto:geoffb@chuggalug.clues.com)>

Geoff Buckingham wrote:

- > *On Thu, Sep 04, 2003 at 01:12:45AM -0700, Terry Lambert wrote:*
- > > *Yes. Limit the number of CG bitmaps you examine simultaneously,*
- > > *and make the operation multiple pass over the disk. This is not*
- > > *that hard a modification to fsck, and it can be done fairly*
- > > *quickly by anyone who understands the code. The code in time to*
- > > *fsck the disk will go up inversely proportionally to the amount*
- > > *of RAM it's allowed to use, which is limited to the UVA size*
- > > *minus the fsck program size itself, and the fsck buffers used for*
- > > *things like FS metadata for a given file/directory.*
- >
- > *Pardon my ignorance but does the number of inodes in the filesystem have a*
- > *significant impact on the memory requirement of fsck?*

I can't answer empirically, but extrapolating from the empirical data that I \*do\* have, the time is going to go up proportional to the number of blocks in use, and the number of blocks in use is going to equal the average number of blocks per file times the number of files, and given that there is one inode per file, you will bound the amount of blocks by bounding the number of inodes.

This makes the answer "yes, indirectly". What passes get run really depend on how your FS is configured. By default, a background fsck will only check for blocks that are marked as used in the CG bitmaps that are not actually used; so this is a CG bitmap vs. all inodes direct and indirect block lists consistency check only.

Most of the incremental or multipass techniques I've discussed on the mailing list assume either a full fsck, or that you are able to lock individual CGs, or at least ranges on the disk, if you wish to do a BG check; or that you read-only the entire disk until you are done, and maintain a list of "needs update" items (this can be very compact, since it can be run-length encoded or otherwise highly compressed).

## freebsd-performance: Re: 20TB Storage System (fsck????)

If you read the fsck manual page and understand what it means, you can get some idea of what parameters effect it in what phases:

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free map.

Every inode needs to be scanned to see what blocks in the free map should not be in the free map. The "free map" is the set of bits in the set of all cylinder group bitmaps.

Cross-checking multiple references for directories is a process of combinatorial math. You take N inodes 2 at a time and compare them. A trick that is possible, if you are willing to rebuild the CG bitmaps in core, or are willing to double the space for the set you are examining would be to examine a range, and at the same time keep a shadow. Zero the shadow, and pass the list of inodes once, setting bits in the shadow. If you go to set a bit and it's already set, \*then\* you go back and find out who it was who had the bit set. This is probably an OK trade-off, particularly if you maintain a list of "this-file-this-suspect-bit, and then pass the FS again (large numbers of cross-linked blocks are rare).

The second of these operations is as expensive as:

$$\#inodes\_used * (\#inodes\_used - 1) * (\#indirect\_blocks ** 2 - 1)$$

2. Blocks claimed by an inode outside the range of the filesystem.

What this really should say is "which are outside the range". In other words, bogus block numbers. This is a compare that can be made during a direct linear search.

3. Incorrect link counts.

This is a directory entry vs. inode count. The expense of this operation depends on whether you are directory-entry-major or on your pass, and the relative number of directory entries vs. inodes. For most FS's, the number of entries is going to be ~15% higher than the number of inodes; this is because of the hard links to directories from their parents, and to parent directories from their child directories. This number could be much, much higher on an FS with a large number of hard links per file.

The thing you have to worry about is tracking the number of hard links per inode, and whether you can do this all in memory (e.g. with a linear array of integers of the same type size as the link count, whose length is equal to the number of inodes available in the system), or whether you have to break the job

up and pass over the directory structure multiple times. If you can't keep all the items in memory, and must make multiple passes, then it's better to be inode-major; otherwise, it's better to be directory-major.

4. Size checks:

Directory size not a multiple of DIRBLKSIZ.

Simple check; can be done during one of the single linear passes.

Partially truncated file.

Also a linear check, but somewhat harder to handle.

5. Bad inode format.

Self-inconsistent contents on inodes.

6. Blocks not accounted for anywhere.

Every blocks in the free map that's not there and should be, because it's not claimed by a directory or inode. The "free map" is the set of bits in the set of all cylinder group bitmaps. This is the background fsck on an FS with soft updates case.

7. Directory checks:

File pointing to unallocated inode.

Directory says it's there, inode say's it's not. Linear pass over the directory space, looking up each inode.

Inode number out of range.

Linear pass over the directory space, looking at each directory entry. Inode is out of range if it's not in the set of inodes per cylinder group times number of cylinder groups.

Directories with unallocated blocks (holes).

Directories are not allowed to be sparse, since they are accessed via block I/O, linearly, from first byte to last, in order to scan for matches on lookup/create/rename/iteration operations.

Dot or dot-dot not the first two entries of a directory or

Internal consistency check.

having the wrong inode number.

Inode number of child and parent do not match expected corresponding values; this is a one element lookahead hierarchical

## freebsd-performance: Re: 20TB Storage System (fsck????)

traversal, so it's not quite linear; best handled by depth-first recursive descent.

### 8. Super Block checks:

- More blocks for inodes than there are in the filesystem.
- Bad free block map format.
- Total free block and/or free inode count incorrect.

Trivial checks. Free block/free inode are maintained as part of the other checks, so the superblock is kept in core for the duration.

- > *I ask as it was previously stated the smallest file on the 10TB filesystem*
- > *would be 500MB which would enable a vastley reduced number of inodes and*
- > *possibly very large block fragment and cluster sizes?*

The thing that matters is the number of allocated inodes, not the number of total inodes. If they aren't allocated, then they don't have blocks allocated to them, and if they don't have blocks allocated to them, then those blocks don't need to be checked, and they don't need to be checked.

Obviously, the above information is just a brief oversimplification, but it gives you a 50,000 foot view of the issues and trade-offs you could make to fit in less RAM. For more detailed information:

The UNIX+ File System Check Program

<http://citeseer.nj.nec.com/31351.html>

A Fast File System for UNIX (1984)

<http://citeseer.nj.nec.com/mckusick84fast.html>

Basically, if you want to learn, you're going to have to read. 8-).

-- Terry

---

freebsd-performance@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-performance>

To unsubscribe, send any mail to "freebsd-performance-unsubscribe@freebsd.org"