

RE: Flash disks and FFS layout heuristics

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2008-03/msg00191.html>

- *From:* "Martin Fouts" <mfouts@xxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 12:51:35 -0700
-

-----Original Message-----

From: Matthew Dillon [<mailto:dillon@xxxxxxxxxxxxxxxxxxxxxxxxxx>]

Hamming codes (ECC codes) are very fragile beasts. While they are in the same family as a CRC it is a really bad idea to try to use the ECC code as your CRC which is why I recommended against it in my previous posting.

True, but when you're working with a part that does ECC in HW, you're stuck with the ECC it does.

I've written numerous filesystems, including a NOR flash filesystem (whos characteristics are somewhat different due to the availability of byte-write). In my opinion, designing a filesystem *specifically* for NAND flash is a mistake because the technology is

rapidly

evolving and such a filesystem would wind up being obsolete in fairly short order.

Well, those of us who are shipping devices with flash parts in them have a somewhat different view on that, which is why I've worked on three NAND specific file systems in the last four years. Two of those are in use in shipping devices, and are expected to be in use for five or more years.

For example, the simple addition of some front-end non-volatile cache,

RE: Flash disks and FFS layout heuristics

such as a dime-cap-backed static ram, would have a very serious effect on any such filesystem design.

Yes. However since the phone market makes such a change very unlikely, because of cost pressures, it's not one we take into consideration.

It is far far better to design the filesystem around generally desired characteristics, such as good write locality of reference (though, again, indices still

have to be updated and those usually do not have good locality of reference).

You've talked yourself into pretty much the same mistake that led to jffs2, which turned out to be a terrible idea.

DragonFly's HAMMER has pretty good write-locality of reference but still does random updates for B-Tree indices and things like

the mtime and atime fields. It also uses numerous blockmaps that could

make direct use of a flash sector-mapping translation layer (1). It might be adaptable.

You are pretty much describing the data structures that have made jffs2 such a poor performer.

(1) A flash sector-mapping translation layer gives a filesystem the ability to use 'named block numbers'. For example, the

NOR filesystem I did used 32 bit named block numbers regardless of the size of the flash (which was typically only 2MB). The filesystem topology was

RE: Flash disks and FFS layout heuristics

actually encoded into the block number it self. In other words, the filesystem is not bound to a linear range of block numbers

it is

simply bound

Works OK for NOR. Has interesting problems, mainly with maintaining the block number map reliably in storage, when attempted on NAND.

What does this mean? This means that what you really want to do is not necessarily write a filesystem that is explicitly designed for NAND operation, but instead write a filesystem that is explicitly designed to run on top of an abstracted topology (such as

one

where you can have named block numbers), and which generally has the

desired

features for locality of reference. Such a filesystem would not become obsolete anywhere near as quickly as a nand-specific filesystem

would and

rebuilding an abstracted topology (whos underlying code would become obsolete as the technology changes) is a whole lot easier

then

redesigning a filesystem.

There's really only one topology that's efficient for a NAND device, and that's to do log-like writing coupled with garbage collection.

I am quite partial to the named-block concept, I really think it's the best way to go for flash filesystem design. The flash already has to have a sector-translation mechanism, making the jump to

a

full blown named-block model is only a small additional step.

The devil in the details of your naming scheme turns out to be managing

RE: Flash disks and FFS layout heuristics

the name translation information within the NAND storage itself. This is the source of significant performance problems in jffs2, for example, and have a huge amount of code complexity in the commercial system I work with.

freebsd-arch@xxxxxxxxxxx mailing list

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

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@xxxxxxxxxxx"