

Re: [PATCH -mm] mm: more likely reclaim MADV_SEQUENTIAL mappings

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-07/msg09441.html>

- *From:* Nick Piggin <nickpiggin@xxxxxxxxxxxxx>
 - *Date:* Tue, 22 Jul 2008 13:43:39 +1000
-

On Tuesday 22 July 2008 13:04, Rik van Riel wrote:

On Tue, 22 Jul 2008 12:54:28 +1000

Nick Piggin <nickpiggin@xxxxxxxxxxxxx> wrote:

On Tuesday 22 July 2008 12:36, Rik van Riel wrote:

On Tue, 22 Jul 2008 12:02:26 +1000

Nick Piggin <nickpiggin@xxxxxxxxxxxxx> wrote:

I don't actually care what the man page or posix says if it is obviously silly behaviour. If you want to dispute the technical points of my post, that would be helpful.

Application writers read the man page and expect MADV_SEQUENTIAL to do roughly what the name and description imply.

If you think that the kernel should not bother implementing what the application writers expect, and the application writers should implement special drop-behind magic for Linux, your expectations may not be entirely realistic.

The simple fact is that if you already have the knowledge and custom code for sequentially accessed mappings, then if you know the pages are not going to be used, there is a *far* better way to do it by unmapping them than the kernel will ever be able to do itself.

Applications are not developed just for Linux.

Re: [PATCH -mm] mm: more likely reclaim MADV_SEQUENTIAL mappings

Application writers expect MADV_SEQUENTIAL to behave a certain way and this 5 line patch implements that.

OK well applications will still work fine without it, and the ones that really want high performance and low cache pollution on any other platform will not be doing this anyway.

I don't care if it is 5 or 500 lines actually, I can see what it does but I am arguing that it is not a good idea anyway. I have certainly not seen anything to justify adding overhead here.

Also, it would be perfectly valid to want a sequentially accessed mapping but not want to drop the pages early.

That is exactly what Johannes' patch does. All it does is change the behaviour of pages that have already reached the end of the LRU lists.

It does not do any kind of early drop-behind or other strange magic.

I don't understand what you are getting at here. It exactly does put a lower priority on the page access, but what I am saying is that you may not want that.

Consider this: if the app already has dedicated knowledge and syscalls to know about this big sequential copy, then it should go about doing it the *right* way and really get performance improvement. Automatic unmap-behind even if it was perfect still needs to scan LRU lists to reclaim.

Doing nothing _also_ ends up with the kernel scanning the LRU lists, once memory fills up.

But we are not doing nothing because we already know and have coded for the fact that the mapping will be accessed once, sequentially.

Re: [PATCH -mm] mm: more likely reclaim MADV_SEQUENTIAL mappings

Re: [PATCH -mm] mm: more likely reclaim MADV_SEQUENTIAL mappings

Now that we have gone this far, we should actually do it properly and
1. unmap after use, 2. POSIX_FADV_DONTNEED after use. This will give
you much better performance and cache behaviour than any automatic
detection scheme, and it doesn't introduce any regressions for existing
code.

If you run just one instance of the application!

Think about something like an ftp server or a media server,
where you want to cache the data that is served up many
times, while evicting the data that got served just once.

The kernel has much better knowledge of what the aggregate
of all processes in the system are doing than any individual
process has.

That's true, but this case isn't really very good anyway. The information
goes away after you drop the mapping anyway. Or did you hope that the
backup program or indexer keeps all those mappings open until all the pages
have filtered through? Or maybe we can add yet more branches into the unmap
path to test for this flag as well?

I don't think it is a good idea to add random things just because they seem
at first glance like a good idea.

Scanning the LRU lists is a given.

It is not.

Regardless of whether or not the application unmaps the pages
by itself, the pages will still be on the LRU lists.

I clearly was talking about FADV_DONTNEED. I know the pages will be on LRU
lists after unmapping.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>

Re: [PATCH -mm] mm: more likely reclaim MADV_SEQUENTIAL mappings