Re: mmap/mlock performance versus read

From: Linus Torvalds (*torvalds@transmeta.com*)
Date: Wed Apr 05 2000 - 15:18:19 EST

- **Next message:** JM Geremia: "Audio FS Bad Idea"
- **Previous message:** Elmer Joandi: "Doubled bogomips on SMP"
- **In reply to:** Paul Barton-Davis: "mmap/mlock performance versus read"
- **Next in thread:** Albert D. Cahalan: "Re: mmap/mlock performance versus read"
- **Messages sorted by:** [date] [thread] [subject] [author]

In article <200004042249.SAA06325@op.net>,
Paul Barton-Davis <pbd@Op.Net> wrote:
>
>I was very disheartened to find that on my system the mmap/mlock
*>approach took *3 TIMES* as long as the read solution. It seemed to me*
>that mmap/mlock should be at least as fast as read. Comments are
>invited.

People love mmap() and other ways to play with the page tables to
optimize away a copy operation, and sometimes it is worth it.

HOWEVER, playing games with the virtual memory mapping is very expensive
in itself. It has a number of quite real disadvantages that people tend
to ignore because memory copying is seen as something very slow, and
sometimes optimizing that copy away is seen as an obvious improvment.

Downsides to mmap:
- quite noticeable setup and teardown costs. And I mean _noticeable_.
  It's things like following the page tables to unmap everything
  cleanly. It's the book-keeping for maintaining a list of all the
  mappings. It's The TLB flush needed after unmapping stuff.
- page faulting is expensive. That's how the mapping gets populated,
  and it's quite slow.

Upsides of mmap:
- if the data gets re-used over and over again (within a single map
  operation), or if you can avoid a lot of other logic by just mapping
  something in, mmap() is just the greatest thing since sliced bread.

  This may be a file that you go over many times (the binary image of
  an executable is the obvious case here - the code jumps all around
  the place), or a setup where it's just so convenient to map the whole
  thing in without regard of the actual usage patterns that mmap() just
  wins. You may have random access patterns, and use mmap() as a way
  of keeping track of what data you actually needed.

- if the data is large, mmap() is a great way to let the system know
  what it can do with the data-set. The kernel can forget pages as
  memory pressure forces the system to page stuff out, and then just
  automatically re-fetch them again.

And the automatic sharing is obviously a case of this..

But your test-suite (just copying the data once) is probably pessimal
for mmap().

Linus

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

- **Next message:** JM Geremia: "Audio FS Bad Idea"
- **Previous message:** Elmer Joandi: "Doubled bogomips on SMP"
- **In reply to:** Paul Barton-Davis: "mmap/mlock performance versus read"
- **Next in thread:** Albert D. Cahalan: "Re: mmap/mlock performance versus read"
- **Messages sorted by:** [date] [thread] [subject] [author]
