

Re: ten thousand small processes

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-06/0088.html>

From: D. J. Bernstein (*djb_at_cr.yo.to*)

Date: 06/25/03

Date: 25 Jun 2003 06:06:29 -0000

To: freebsd-performance@freebsd.org

As I said, I don't particularly care about the text segment. I'm not talking about ten thousand separate programs.

Why does the memory manager keep the stack separate from data? Suppose a program has 1000 bytes of data+bss. You could organize VM as follows:

```
0x7fffac18 0x7fffb000 0x80000000
<----- stack data+bss text, say 5 pages heap ----->
```

As long as the stack doesn't chew up more than 3096 bytes and the heap isn't used, there's just one page per process.

As for page tables: Instead of allocating space for a bunch of nearly identical page tables, why not overlap page tables, with the changes copied on a process switch?

As for 39 pages of VM, mostly stack: Can the system actually allocate 390000 pages of VM? I'm only mildly concerned with the memory-management time; what bothers me is the loss of valuable address space. I hope that this 128-kilobyte stack carelessness doesn't reflect a general policy of dishonest VM allocation ("overcommitment"); I need to be able to preallocate memory with proper error detection, so that I can guarantee the success of subsequent operations.

As for malloc()'s careless use of memory: Is it really asking so much that a single malloc(1) not be expanded by a factor of 16384?

Here's a really easy way to improve malloc(). Apparently, right now, there's no use of the space between the initial brk and the next page boundary. Okay: allocate that space in the simplest possible way---

```
static wherewenormallystart = 0;
static freebie;
```

```
malloc(n)
{
    if (!wherewenormallystart) {
```

freebsd-performance: Re: ten thousand small processes

```
wherewenormallystart = rounduptopage(sbrk(0));
freebie = wherewenormallystart - rounduptoalign(sbrk(0));
}
n = rounduptoalign(n);
if (n < freebie) {
    freebie -= n;
    if (sbrk(0) <= wherewenormallystart) brk(wherewenormallystart - freebie);
    return wherewenormallystart - freebie - n;
}
do what we normally do;
}

free(x)
{
    if (x < wherewenormallystart) return;
    do what we normally do;
}
```

---with no waste of space and practically no waste of time. Maybe add 8192 to wherewenormallystart; this is lots of room for people who know how to write small programs, and the cost is unnoticeable for people who don't.

(Quite a few of my programs simulate this effect by checking for space in a bss array, typically 2K. But setting aside the right amount of space would mean compiling, inspecting the brk alignment, and recompiling. I also feel bad chewing up space on systems where malloc() actually knows what it's doing.)

As for the safety of writing code that makes malloc() fail horribly: After the Solaris treatment of BSD sockets, and the ``look, Ma, I can make an only-slightly-broken imitation of poll() using select()!" epidemic, I don't trust OS distributors to reserve syscall names for actual syscalls. I encounter more than enough portability problems without going out of my way to look for them.

---D. J. Bernstein, Associate Professor, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago

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"