

Re: SBCL performance on OS X

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-12/msg00455.html>

- *From:* Christophe Rhodes <csr21@xxxxxxxxxx>
 - *Date:* Tue, 06 Dec 2005 10:02:15 +0000
-

Christophe Rhodes <csr21@xxxxxxxxxx> writes:

> William Bland <doctorbill.news@xxxxxxxxxx> writes:

>

>> On Linux, the program dies after less than a second, having
>> allocated around 520MB. On my Powerbook the same program takes 1
>> minute 13 seconds to allocate the same amount of memory!

>

> The garbage collection implementations on x86-family machines and
> non-x86es for SBCL are completely different. The non-x86 machines use
> a simple Cheney semi-space collector [...]

One thing I failed to mention, but is relevant for your experiment, is that the "semi-space"ness is important: if there is 512Mb of live data, the working set of the GC is 1024Mb: because that live data has to be copied to the new semi-space. Thus, you will hit swap earlier than you might expect.

Other things from your blog that I'll respond to:

> I had been led to believe that modern GCs could actually be faster
> than C at allocating memory.

I don't think anyone's ever claimed that. However, modern GCs can actually be faster than C at memory management: that is, you need not just malloc() but also free(), or its equivalent. There are also usually weasel-words such as "for real-world tasks" associated, because for your experimental application the obvious winning strategy is a malloc() which bumps a heap free-pointer and a free() which does nothing: I guarantee that this will beat any implementation you can come up with, and also that this implementation strategy would be wildly unpopular with every other user :-).

> OpenMCL on the powerbook is an order of magnitude slower than SBCL
> on the server

OpenMCL, similar to SBCL on x86s, has a generational (I think they call it "ephemeral", but it means the same) garbage collector, so I would expect it to perform better than SBCL does on powerpcs; the

Re: SBCL performance on OS X

difference you see between OpenMCL on the powerpc and SBCL on the x86 is more likely to be representative of the true difference in memory handling between the hardware/kernel platforms than straight comparisons between SBCLs.

```
> if(malloc(10*1024*1024) == NULL)
> {
> ok = 0;
> }
```

This body of your "equivalent" C program doesn't lend itself to comparison with the lisp implementation of your experiment terribly well. In likely ascending order of seriousness:

- * the preprocessor will constant-fold away the multiplication here, whereas you have a variable allocation in the lisp;
- * your timing code does exact integer division in lisp and approximate floating point division in C; the exact integer division is likely to create garbage;
- * you don't do anything with the returned pointer, whereas in lisp you allocate an extra two-word structure to store it in, and mutate the heap;
- * you are returning uninitialized memory here, whereas I would guess that the lisp make-array returns initialized memory; to measure the same amount of work, I would suggest using calloc() instead.

Which of these actually matters I don't know; I'd be surprised if the first had any measurable effect; the fourth is likely to be quite serious, as it involves touching 10Mb of memory per iteration; the third is more interesting, because it potentially leads to heap fragmentation.

```
> if I can't rapidly make Lisp processes die with out-of-memory
> errors
```

Possibly the most straightforward way of achieving this is to shrink the available heap size. This is easy to do for SBCL: simply change the parameters in src/compiler/ppc/parms.lisp and recompile.

```
> If anything, the powerbook has the edge over the server (in terms of
> raw MHz/GB numbers – yes, I know these are completely different chip
> architectures though).
```

The difference in the CPU chip architecture is probably not relevant; much more important for this experiment, since almost no computation is being done, are memory bandwidth and latency.

Christophe

Re: SBCL performance on OS X

- *Follow-Ups:*
 - ◆ *Re: SBCL performance on OS X*
 - ◇ *From:* William Bland

- *References:*
 - ◆ *SBCL performance on OS X*
 - ◇ *From:* William Bland
 - ◆ *Re: SBCL performance on OS X*
 - ◇ *From:* Christophe Rhodes

- Prev by Date: *Re: SBCL performance on OS X*
- Next by Date: *Re: Koch Figures – LTK package*
- Previous by thread: *Re: SBCL performance on OS X*
- Next by thread: *Re: SBCL performance on OS X*
- Index(es):
 - ◆ *Date*
 - ◆ *Thread*