

Re: HZ=100: not necessarily better?

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2006-06/msg00038.html>

- *From:* Danial Thom <danial_thom@xxxxxxxxxx>
 - *Date:* Sat, 17 Jun 2006 17:06:42 -0700 (PDT)
-

----- Robert Watson <rwatson@xxxxxxxxxxxxx> wrote:

Scott asked me if I could take a look at the impact of changing HZ for some simple TCP performance tests. I ran the first couple, and got some results that were surprising, so I thought I'd post about them and ask people who are interested if they could do some investigation also. The short of it is that we had speculated that the increased CPU overhead of a higher HZ would be significant when it came to performance measurement, but in fact, I measure improved performance under high HTTP load with a higher HZ. This was, of course, the reason we first looked at increasing HZ: improving timer granularity helps improve the performance of network protocols, such as TCP. Recent popular opinion has swung in the opposite direction, that higher HZ overhead outweighs this benefit, and I think we should be cautious and do a lot more investigating before assuming that is true.

Simple performance results below. Two boxes on a gig-e network with if_em ethernet cards, one running a simple web server hosting 100 byte pages, and the other downloading them in parallel (netrate/http and netrate/httpd). The performance difference is marginal, but at least in the SMP case, likely more

Re: HZ=100: not necessarily better?

than a measurement error or cache alignment fluke. Results are transactions/second sustained over a 30 second test -- bigger is better; box is a dual xeon p4 with HTT; 'vendor.*' are the default 7-CURRENT HZ setting (1000) and 'hz.*' are the HZ=100 versions of the same kernels. Regardless, there wasn't an obvious performance improvement by reducing HZ from 1000 to 100. Results may vary, use only as directed.

What we might want to explore is using a programmable timer to set up high precision timeouts, such as TCP timers, while keeping base statistics profiling and context switching at 100hz. I think phk has previously proposed doing this with the HPET timer.

I'll run some more diverse tests today, such as raw bandwidth tests, pps on UDP, and so on, and see where things sit. The reduced overhead should be measurable in cases where the test is CPU-bound and there's no clear benefit to more accurate timing, such as with TCP, but it would be good to confirm that.

Robert N M Watson
Computer Laboratory
University of Cambridge

```
peppercorn:~/tmp/netperf/hz> ministat *SMP
x hz.SMP
+ vendor.SMP
```

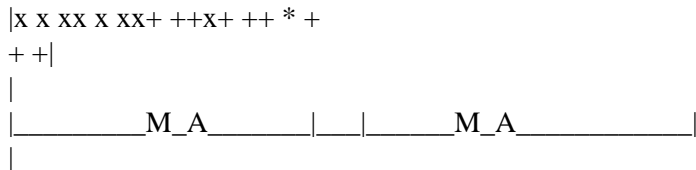
```
+-----+
|xx x xx x xx x ++
+ + + + + + + + |
| _____A_____ |
| _____A_____M_____ | |
+-----+
```

Re: HZ=100: not necessarily better?

N Min Max
Median Avg Stddev
x 10 13715 13793 13750
13751.1 29.319883
+ 10 13813 13970 13921
13906.5 47.551726
Difference at 95.0% confidence
155.4 +/- 37.1159
1.13009% +/- 0.269913%
(Student's t, pooled s = 39.502)

peppercorn:~/tmp/netperf/hz> ministat *UP
x hz.UP
+ vendor.UP

+-----+



+-----+

N Min Max
Median Avg Stddev
x 10 14067 14178 14116
14121.2 31.279386
+ 10 14141 14257 14170
14175.9 33.248058
Difference at 95.0% confidence
54.7 +/- 30.329
0.387361% +/- 0.214776%
(Student's t, pooled s = 32.2787)

frebsd-performance@xxxxxxxxxxx mailing list

--- Robert Watson <rwatson@xxxxxxxxxxx> wrote:

Scott asked me if I could take a look at the impact of changing HZ for some simple TCP performance tests. I ran the first couple, and got some results

Re: HZ=100: not necessarily better?

Re: HZ=100: not necessarily better?

that were surprising, so I thought I'd post about them and ask people who are interested if they could do some investigation also. The short of it is that we had speculated that the increased CPU overhead of a higher HZ would be significant when it came to performance measurement, but in fact, I measure improved performance under high HTTP load with a higher HZ. This was, of course, the reason we first looked at increasing HZ: improving timer granularity helps improve the performance of network protocols, such as TCP. Recent popular opinion has swung in the opposite direction, that higher HZ overhead outweighs this benefit, and I think we should be cautious and do a lot more investigating before assuming that is true.

Simple performance results below. Two boxes on a gig-e network with if_em ethernet cards, one running a simple web server hosting 100 byte pages, and the other downloading them in parallel (netrate/http and netrate/httpd). The performance difference is marginal, but at least in the SMP case, likely more than a measurement error or cache alignment fluke. Results are transactions/second sustained over a 30 second test -- bigger is better; box is a dual xeon p4 with HTT; 'vendor.*' are the default 7-CURRENT HZ setting (1000) and 'hz.*' are the HZ=100 versions of the same kernels. Regardless, there wasn't an obvious performance improvement by reducing HZ from 1000 to 100. Results may vary, use only as directed.

What we might want to explore is using a programmable timer to set up high precision timeouts, such as TCP timers, while keeping base statistics profiling and context switching at 100hz. I think phk has previously proposed doing this with the HPET timer.

I'll run some more diverse tests today, such as raw bandwidth tests, pps on

Re: HZ=100: not necessarily better?

Re: HZ=100: not necessarily better?

UDP, and so on, and see where things sit. The reduced overhead should be measurable in cases where the test is CPU-bound and there's no clear benefit to more accurate timing, such as with TCP, but it would be good to confirm that.

Robert N M Watson
Computer Laboratory
University of Cambridge

```
peppercorn:~/tmp/netperf/hz> ministat *SMP
x hz.SMP
+ vendor.SMP
```

```
+-----+
|xx x xx x xx x ++
+ + + + + + + |
| |_____A_____|
|_____A_____M_____||
```

```
+-----+
N Min Max
Median Avg Stddev
x 10 13715 13793 13750
13751.1 29.319883
+ 10 13813 13970 13921
13906.5 47.551726
Difference at 95.0% confidence
155.4 +/- 37.1159
1.13009% +/- 0.269913%
(Student's t, pooled s = 39.502)
```

```
peppercorn:~/tmp/netperf/hz> ministat *UP
x hz.UP
+ vendor.UP
```

```
+-----+
|x x xx x xx+ ++x+ ++ * +
+ + |
|
|_____M_A_____||_____M_A_____|
|
```

N Min Max
Median Avg Stddev
x 10 14067 14178 14116
14121.2 31.279386
+ 10 14141 14257 14170
14175.9 33.248058
Difference at 95.0% confidence
54.7 +/- 30.329
0.387361% +/- 0.214776%
(Student's t, pooled s = 32.2787)

freebsd-performance@xxxxxxxxxxxxx mailing list

And what was the cost in cpu load to get the extra couple of bytes of throughput?

Machines have to do other things too. That is the entire point of SMP processing. Of course increasing the granularity of your clocks will cause to you process events that are clock-reliant more quickly, so you might see more "throughput", but there is a cost. Weighing (and measuring) those costs are more important than what a single benchmark does.

At some point you're going to have to figure out that there's a reason that every time anyone other than you tests FreeBSD it completely pigns out. Squeezing out some extra bytes in netperf isn't "performance". Performance is everything that a system can do. If you're eating 10% more cpu to get a few more bytes in netperf, you haven't increased the performance of the system.

You need to do things like run 2 benchmarks at once. What happens to the "performance" of one benchmark when you increase the "performance" of the other? Run a database benchmark while you're running a network benchmark, or while you're passing a controlled stream of traffic through the box.

I just finished a couple of simple tests and find that 6.1 has not improved at all since 5.3 in

Re: HZ=100: not necessarily better?

basic interrupt processing and context switching performance (which is the basic building block for all system performance). Bridging 140K pps (a full 100Mb/s load) uses 33% of the cpu(s) in FreeBSD 6.1, and 17% in Dragonfly 1.5.3, on a dual-core 1.8Ghz opteron system. (I finally got vmstat to work properly after getting rid of your stupid 2 second timeout in the MAC learning table). I'll be doing some mySQL benchmarks next week while passing a controlled stream through the system. But since I know that the controlled stream eats up twice as much CPU on FreeBSD, I already know much of the answer, since FreeBSD will have much less CPU left over to work with.

Its unfortunate that you seem to be tuning for one thing while completely unaware of all of the other things you're breaking in the process. The Linux camp understands that in order to scale well they have to sacrifice some network performance. Sadly they've gone too far and now the OS is no longer suitable as a high-end network appliance. I'm not sure what Matt understands because he never answers any questions, but his results are so far quite impressive. One thing for certain is that its not all about how many packets you can hammer out your socket interface (nor has it ever been). Its about improving the efficiency of the system on an overall basis. Thats what SMP processing is all about, and you're never going to get where you want to be using netperf as your guide.

I'd also love to see the results of the exact same test with only 1 cpu enabled, to see how well you scale generally. I'm astounded that no-one ever seems to post 1 vs 2 cpu performance, which is the entire point of SMP.

DT

Do You Yahoo!?

Tired of spam? Yahoo! Mail has the best spam protection around
<http://mail.yahoo.com>

freebsd-performance@xxxxxxxxxxxxx mailing list
<http://lists.freebsd.org/mailman/listinfo/freebsd-performance>

Re: HZ=100: not necessarily better?

Re: HZ=100: not necessarily better?

To unsubscribe, send any mail to "freebsd-performance-unsubscribe@xxxxxxxxxxxxx"