


By Poul-Henning Kamp

STDDEV or: "It Didn't Happen!"



Many, many years ago, back when Dilbert cartoons were still funny, I tried to do what can best be described as "nano-benchmarking." The basic idea was to boot two hardware-identical FreeBSD computers in disk-less configuration, run them for several days, and measure how proposed patches to the FreeBSD kernel code improved or worsened performance.

It soon became obvious that the quartz crystals on the motherboards were better at measuring temperature than time, and that limited my precision to, at best, three significant digits.

Being a "time-nut," I replaced the crystals with an external signal from a high-quality ovenized, quartz crystal, which, again, was steered using GPS signals, from a state-of-the-art, "Motorola Oncore," timing GPS receiver. That got me to eight significant digits.

.....Or did it?

That's when I voluntarily broke out the statistics textbook and started to appreciate statistics as a tool. Statistics is not any normal person's favorite branch of mathematics; it involves a lot of number-crunching and it gives fuzzy answers:

65% of all Americans believe that frozen pizza will never be any good, and there's nothing science can do about it. Source: Widgey & Associates. Margin of error within 9%. From a telephone survey of 204 Americans, Spring 1993 [1].

But even the most cursory study of the developments from the previous century will reveal that it is statistics, all the way from Erlang's formulas for telephone system capacity, over Shewhart's quality control of the manufacturing of telephone handsets, to Googles PageRank, and the current "Big Data" bubble, which I trust some future historian will chronicle in a book called "Covariances Gone Wild."

Statistics is powerful, but hard. My favorite piece of overlooked statistics—an article which fixed the odds of global warming to at least 100,000 against one (in 1992, back when Windows 3.1 came out)—

took me a full week to comprehend [2].

Because statistics is hard work, most people just skip it entirely, and, therefore, almost all claims about performance improvements that came across the FreeBSD mailing lists, including some of my own, were just unsubstantiated and anecdotal claims.

Around the time of FreeBSD 5, I decided to do something about that, but what?

Pointing people at a heavy-duty statistics package such as 'R' would be a nonstarter: I had tried it myself and given up on it.

What the project needed was a "Software Tool": A program that did one thing well, and without a lot of work, mental or otherwise.

I wrote a ~500-line C-program, which read one number per line from one or two input files, plotted the data in a cheesy little ASCII-art plot, printed the basic statistical measures of each input file underneath the plot: minimum, maximum, average, median, and standard deviation.

If given two input files, it would calculate the Student's T test, and report if there were more than a 95% chance that the two data sets differed in a meaningful way:

```

$ ministat iguana.txt chameleon.txt
x iguana.txt
+ chameleon.txt
+-----+
|x      *  x              *  +          +  x      +|
|  _____ M  |  A  _____ M  A  _____ |  |
+-----+
      N      Min      Max      Median      Avg      Stddev
x      7      50      750      200      300      238.04761
+      5      150     930      500      540      299.08193
No difference proven at 95% confidence

```

Ministat is an incredibly primitive tool. The ASCII-plot looks horrible; it has no scales and is barely good enough to spot outliers, bifurcations, and other major data trouble, but it has the big advantage that you can email it as plain text.

Student's T is a relatively new breakthrough in statistics, a mere 100 years old [3], and while it is

ZFS experts make their servers **ZING**

Now you can too. Get a copy of.....

Choose ebook, print, or combo. You'll learn to:

- Use boot environment, make the riskiest sysadmin tasks boring.
- Delegate filesystem privileges to users.
- Containerize ZFS datasets with jails.
- Quickly and efficiently replicate data between machines.
- Split layers off of mirrors.
- Optimize ZFS block storage.
- Handle large storage arrays.
- Select caching strategies to improve performance.
- Manage next-generation storage hardware.
- Identify and remove bottlenecks.
- Build screaming fast database storage.
- Dive deep into pools, metaslabs, and more!



Link to: <http://zfsbook.com>

WHETHER YOU MANAGE A SINGLE SMALL SERVER OR INTERNATIONAL DATACENTERS, SIMPLIFY YOUR STORAGE WITH **FREEBSD MASTERY: ADVANCED ZFS**. GET IT TODAY!

not the statistically optimal tool for all circumstances, it is very robust with respect to the input data, and one does not need a lot of skill to interpret its yes/no result. Which is why I committed ministat to FreeBSD with the message:

If your benchmarks are not significant at the 95% confidence level, we don't want to hear about it.

The example data files that go with the source code are from "The Cartoon Guide to Statistics" [4], a competent introductory statistics textbook, written as a comic full of "dad-jokes," by the inimitable Larry Gonick [5].

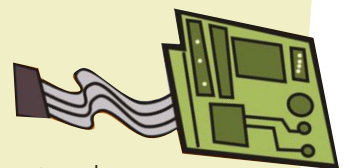
Ministat was not an overnight success; it came with the full stigma of being "statistics," but after a couple of years of reminding people of its existence, it had become the de facto way to argue performance improvements in FreeBSD, and I saw it regularly in BSDcan presentations.

I think OpenBSD was the first place it appeared outside FreeBSD, but today there are also packages for Linux and ports of ministat written in at least Python, Ruby, and Erlang.

And every so often, strangers offer me a beer, because, of course, ministat had to be beerware licensed: Student's T was discovered by William Gosset, an employee of the Guinness Breweries [6].

PHK's Tips for Good Benchmarks

- Run in single user mode. Cron(8) and all the other daemons only add noise.
- Make sure the CPU always runs at the same clock-speed. (BIOS settings etc.; AC-power, etc.)
- If syslog events are generated, run syslogd with an empty syslog.conf; otherwise, do not run it.
- Minimize disk-I/O; avoid it entirely if you can.
- Don't mount filesystems you do not need.
- Mount filesystems read-only if possible.
- Newfs your R/W test filesystem and populate it from a tar or dump file; then unmount and mount it again before every run.
- Use malloc backed or preloaded MD(4) partitions if you can. When writing, SSDs are more unpredictable than rotating platter disks.
- Remove all unnecessary device drivers from the kernel. For instance, if you don't need USB for the test, don't put USB in the kernel. Drivers that attach often have timeouts ticking away.
- Unconfigure hardware you don't use. Detach disk with atacontrol and camcontrol if you do not use them for the test.
- Do not connect or configure the network unless you are testing it.
- Do not run NTPD unless your test takes days to complete.
- Minimize output to serial or VGA consoles. Running output into files on a ramdisk causes less jitter.
- Make sure your test is long enough, but not too long. If your test is too short, timestamping is a problem. If it is too long, temperature changes can ruin it. Rule of thumb: more than a minute, less than an hour.
- Try to keep the temperature as stable as possible around the machine. Temperature changes affect both quartz crystals, CPU thermal management, and seek algorithms in disk drives.
- Run at least 3, but preferably many more, tests of the "before" and "after" code, and analyze the results with ministat.



(continues next page)

(continued)

• **PHK's usual test-plan:**

a[1] b[1]

Do a sanity-check.

a[2] b[2] a[3] b[3]

Run minostat on a[1:3] vs b[1:3]

Do not despair if there is no difference.

a[4] a[5] b[4] b[5] a[6] a[7]

b[6] b[7] a[8] a[9] b[8] b[9]

Run minostat on a[4,6,8] vs a[5,7,9] and on b[4,6,8] vs b[5,7,9]

If there is a difference, stop and think.

Run minostat on a[1:9] vs b[1:9]

If the result is solid, you can stop here.

a[10] a[11] a[12] b[10] b[11] b[12]

a[13] a[14] a[15] b[13] b[14] b[15]

a[16] a[17] a[18] b[16] b[17] b[18]

Run minostat on a[1:3] vs a[16:18] and

on b[1:3] vs b[16:18]

If there is a difference, stop and think.

Run minostat on a[1:18] vs b[1:18]

This is your final result; it is unlikely

to improve much.

- [1] Michael Moore: Adventures in a TV Nation, p. 9. ISBN 0-330-41914-5
- [2] Statistics of Extreme Events with Application to Climate Change. <https://fas.org/irp/agency/dod/jason/statistics.pdf>
- [3] Walter A. Shewhart, the founder of statistical quality-control, positively gushed about Student's T when he heard about it in 1926. <https://archive.org/details/bstj5-2-308>
- [4] <http://www.larrygonick.com/titles/science/the-cartoon-guide-to-statistics/>
- [5] <http://www.larrygonick.com/> Highly recommend all of his books, but in particular the history series.
- [6] https://en.wikipedia.org/wiki/William_Sealy_Gosset



Poul-Henning Kamp is phk@FreeBSD.org, and he used to be one of the most active kernel programmers in the project. These days his main project is the Varnish HTTP-cache, but his laptop still runs FreeBSD-current.

Thank you!

The FreeBSD Foundation would like to acknowledge the following companies for their continued support of the Project. Because of generous donations such as these we are able to continue moving the Project forward.



Are you a fan of FreeBSD? Help us give back to the Project and donate today! freebsdfoundation.org/donate/

Please check out the full list of generous community investors at freebsdfoundation.org/donate/sponsors

Uranium



Iridium

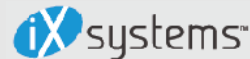


Platinum



VERISIGN™

Gold



Silver



HUAWEI



STORMSHIELD