Take a Freaking Measurement!

Kode Vicious has been going strong for three years now, and thus far there has been no bottom to the well of coding-related questions, conflicts, and conundrums from which he draws. But to keep things fresh and interesting, we need your most pressing, current cries for help from out there in the coding trenches. Are you mystified by a never-before-addressed class of coding problem? Or have you seen an unwelcome shift in development methodologies? Chances are Kode Vicious is familiar with your plight. Drop him a line at kv@acmqueue.com.

Dear KV,

Have you ever worked with someone who is a complete jerk about measuring everything? I work with one such jerk at the moment, and he is driving me up a wall. I cannot make the smallest change in the system without rerunning all sorts of tests, which takes hours, and any suggestion of a change in the design seems to give this jerk the idea that he has to start lecturing me about how there is no data to support what he calls my “suppositions.”

How do you deal with such jerks at work?

Dataless and Damned Annoyed

Dear Dataless,

Either you are a masochist or you do not read my columns, because if you had been reading them, you would know that I would not be taking your side against this so-called “jerk” at work. The “jerk” is actually right, and I’ve always felt that dataless suppositions are like suppositions: they provide only temporary relief and they should both be shoved in the same place.

Since I have recently run into several people who seem

Got a question for Kode Vicious? E-mail him at kv@acmqueue.com—if you dare! And if your letter appears in print, he may even send you a Queue coffee mug, if he’s in the mood. And oh yeah, we edit letters for content, style, and for your own good!

A koder with attitude, KV ANSWERS

YOUR QUESTIONS.

MISS MANNERS HE AIN’T.

to share your disregard for measurement, I figure it’s time to explain to people *how to take a freaking measurement!*

Your letter showed up at an opportune time because I recently decided to make a modification to my trusty MacBook, and it will be relatively easy to use this as an example of how to take a freaking measurement. Before we get to the modification, measurement, and results, I’ll lay out the basics of measurement.

Remember that “computer science is a science,” so we will be using the scientific method, which I hope you learned in elementary school. The scientific method is simple: form a hypothesis, run an experiment, and fake the results to win fame and fortune! Actually we evaluate the experiment, repeating it as many times as necessary to have confidence in the results. The fame and fortune comes after that, or so I am told.

Now let’s get back to the modification I made on my MacBook and how we can use that as an example. I decided to upgrade my internal hard drive from 160 GB to 200 GB since 200-GB drives are now cheaper than they were when I bought the computer and you can now get them with higher rotational speeds.

Of particular interest was a hard disk that spun at 7200 RPM and that the manufacturer claimed was just as cool, in terms of temperature, and required no more power than my original 160-GB, 5400-RPM drive.

My hypothesis, and my hope, was that the new drive would be faster, in terms of throughput, than my old drive; 7200 RPM is greater than 5400 RPM, and since seek time—the time it takes the head of the disk to be over the data you want—is governed by the speed at which the disk spins, it seemed logical to believe that the new drive would improve the responsiveness of my system. How would I test such responsiveness?

Well, I could use the “it feels better” method: install the new drive and see if my computer feels snappier. That is actually a test for idiots—any idiot who says a computer is snappier needs to be, well, dealt with. Instead of the “it feels better” method, I devised a couple of quick tests.

The first was to run a standard benchmark written for Mac OS X, the operating system I'm using. The second was to time how long it took to complete a typical workload. A workload is some job that needs to get done and that is easy to reproduce so that it can be run repeatedly. Not being able to repeat results is called a faith-based approach, and it does not hold water with KV, or anyone using the scientific method.

For my typical workload I chose something I do fairly often on my machine: compiling an operating-system kernel within a virtual machine. In my copious free time I work on FreeBSD, and it's very convenient to carry your test lab in your laptop when you travel as much as I do. This particular workload had several excellent characteristics:

- It is something I do all the time, and it matters to me if my machine is faster or slower when performing it.
- The workload is easy to generate and easy to repeat.
- Compiling a kernel, which consists of a few hundred files, requires a good deal of disk I/O.

Just to have some fun, I also compared two different virtual machine systems—Parallels and VMware—which, it turns out, had some interesting effects.

I decided also to measure the temperature of the drives, as well as the time it took them to do the job. Since the manufacturer was saying that its faster product produced the same amount of heat as a slower product, I thought it made sense to test that claim as well. I tested the temperature using two different methods. The first was to use the internal sensors in the computer, which tell the system the temperature in several places, including on the disk itself. I also used an infrared thermometer pointed at the bottom of the computer to check periodically against what the computer's sensors were telling me.

At this point I had everything I needed to take a freaking measurement: a hypothesis that the new disk would be faster than the old disk; two different ways to measure the performance of the new systems; and, of course, my shiny new disk. First I took a baseline—a set of measurements before the change—and then ran the exact same set of tests after the new drive had been installed. How did it go?

The results were interesting—and not for the reasons I expected. To form a baseline, I used the Xbench program, which runs benchmarks that tell you the speed of the CPU, the graphics and memory subsystems, and, of course, the disk. Here I found mostly what I had expected to find: the new disk was indeed faster on several measurements, in one instance by a factor of five. I was a bit skeptical of such good results because 7200 is not five

times 5400, and the new drive had the same amount of cache as the old drive. I was willing to believe the virtual machine tests more than Xbench for one reason, and that was because they showed that the performance difference between the two disks was much smaller, and more reasonable. The runtimes for the tests went from four minutes to 3:57 for VMware and stayed roughly the same for Parallels.

Was my new disk actually just as slow as my old disk? Was Xbench lying? No. The answer came after another test. VMware has the ability to use both cores of the dual-core processor on my system, so I enabled that feature and reran the test on the new disk. The kernel compile time went down to 2:46. What does that mean? It means that kernel compiles are bound by the CPU and not the disk. So, no, Xbench wasn't lying, but it was not measuring a real workload, or at least not one that matters to me day to day.

What about heat? It turns out that, as far as I can tell, the manufacturer didn't lie. The temperature of the system on all sensors remained nearly the same, to within one degree centigrade on all tests for both disks.

Now, there are questions you might want to ask me about this experiment, such as: Is it statistically significant? The answer is no, and that's because I ran each test only once since in reality I wanted to actually use the disk, not spend a week testing it. A better test would have been to run a number of trials, compare the results, and increase the confidence level of the measurement, but for me it was enough to confirm that I hadn't made the system slower or hotter. Were there other tests I could do? Certainly, but, again, I wanted to use the system, not just test it. The fact is, there is a middle ground between testing every single thing to death, as you believe the jerk wants you to do, and working only on hunches and dataless suppositions. It just depends on what level of confidence you want to have.

KV

KODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who has made San Francisco his home since 1990.

© 2007 ACM 1542-7730/07/1100 \$5.00