How Do Scientists Really Use Computers?

Gregory Wilson

COMPUTERS ARE NOW essential tools in every branch of science, but we know remarkably little about how—or how well—scientists use them. Do most scientists use off-the-shelf software or write their own? Do they really need state-of-the-art supercomputers to solve their problems, or can they do most of what they need to on desktop machines? And how much time do grad students really spend patching their supervisors' crusty old Fortran programs?

To answer these questions, my colleagues and I ran a Web-based survey during the last two months of 2008. We were surprised and gratified that almost 2,000 people took the time to tell us what they were doing. We were equally surprised by what they told us.

A Web-based survey offers clues

Who Responded

First, a few facts about who answered. Thirty-one percent told us they were from the United States, 20 percent from Canada, and 8 percent from the United Kingdom. Germany and Norway came next with 7 percent and 6 percent respectively, while the rest of the world made up the remaining 28 percent. The high representation from Canada and Norway reflects the fact that my colleagues and I are based there, while the low response rate from areas such as Russia and East Asia is undoubtedly due to the fact that we only advertised the survey in English-language channels.

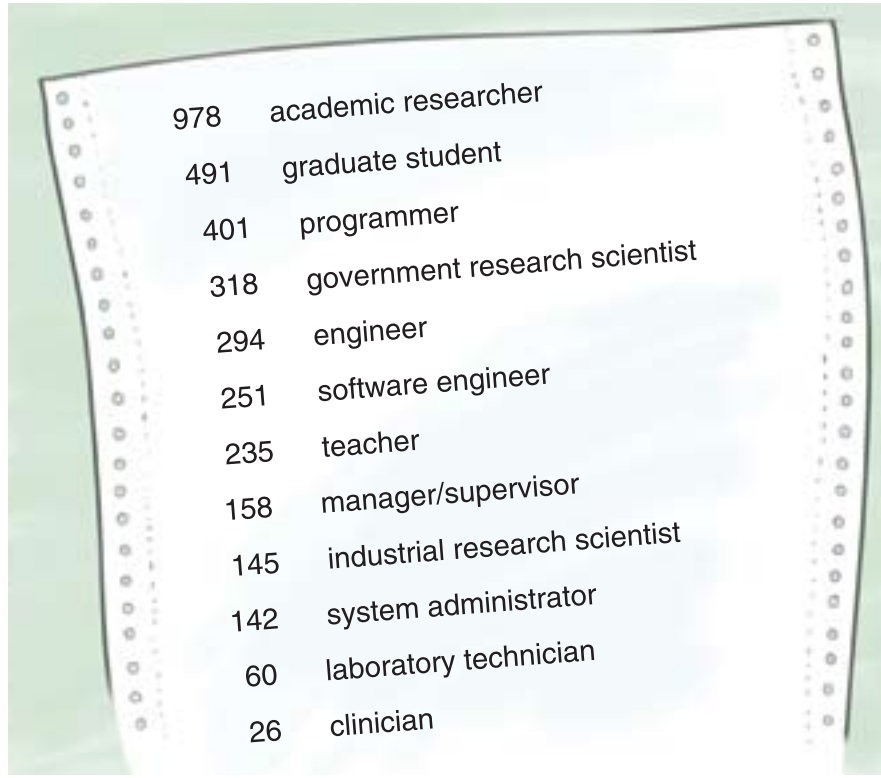
Thirty-three percent of respondents were 18 to 30 years old; 35 percent were 30 to 40, and 17 percent were 40 to 50. The remaining 15 percent were over 50 or, in the case of 15 respondents, didn't answer. These figures are consistent with reports about degrees: Seventy-one percent had a Ph.D. or equivalent, with 18 percent reporting at least an M.Sc.

When asked to identify their roles, over half of our 1,972 respondents chose more than one category *(below)*—which is probably an accurate reflection of how many jobs working scientists actually do.

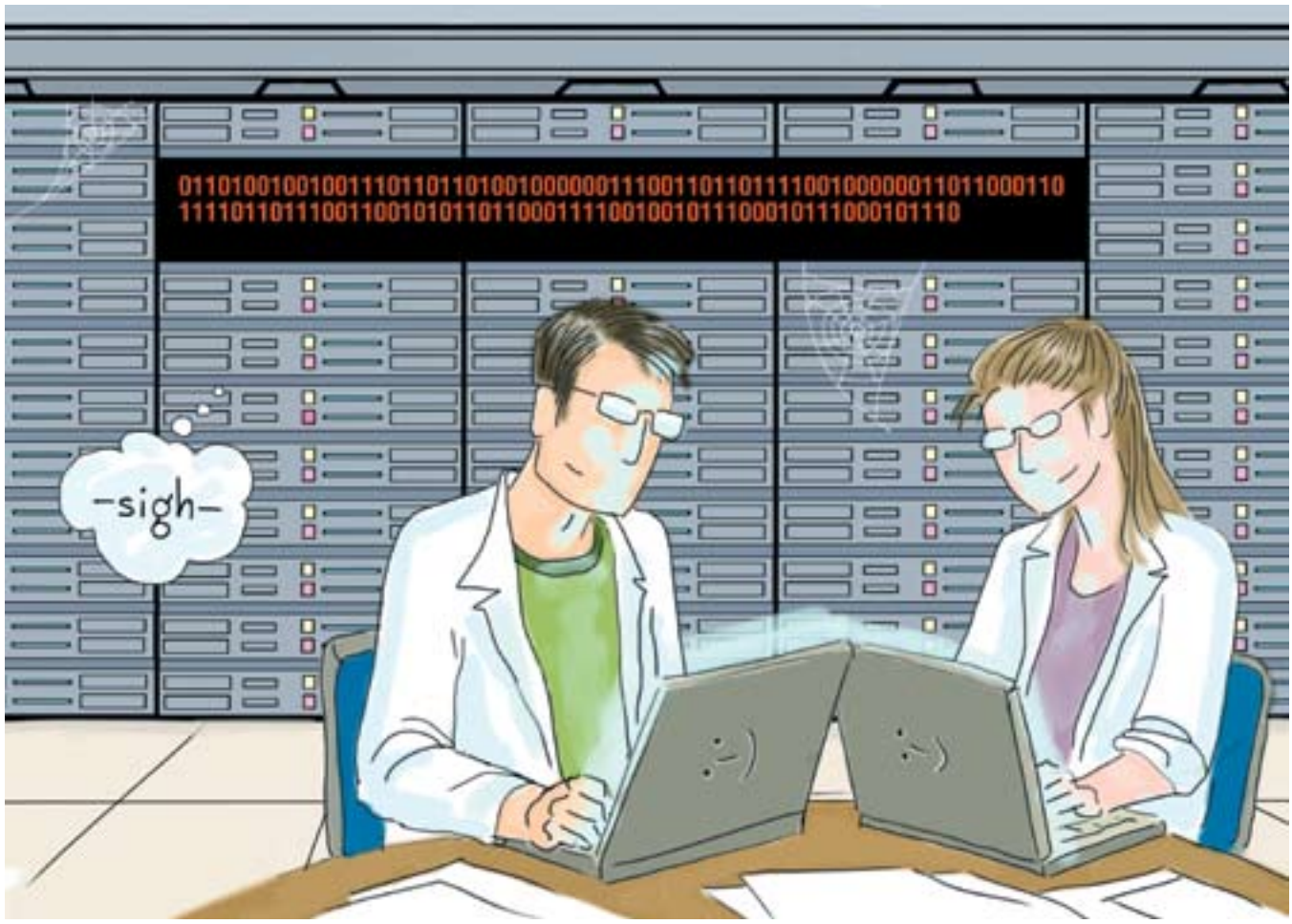
Respondents' descriptions of their disciplines were much more diverse. Roughly 150 identified themselves as physicists, but no other discipline made up more than 5 percent of the sample. These figures are necessarily imprecise, since we had to make a lot of judgment calls when coding them. For example, should astrophysics be classified as a separate discipline from astronomy and physics? If so, what about plasma physics? And how exactly do we count "theological engineering"? (In the end, we discarded that response entirely.)

Getting the Answers

So what did these people tell us? First, respondents work an average of 48 hours a week, of which 30 percent is spent developing software and 40 percent is spent using it. They also re-



Greg Wilson is an adjunct professor of computer science at the University of Toronto. His course material is available at http://www.third-bit.com/swc. Address: Room 3230, Bahen Centre for Information Technology, University of Toronto, Toronto, Ontario, M5S 2E4. Internet: gvwilson@cs.utoronto.ca



port that these proportions are going up—45 percent of respondents say that scientists spend more or much more of their time developing scientific software than they did 5 years ago, and 70 percent say that they spend more or much more time using it. These answers are much higher than we expected, and probably signal that our (self-selected) respondents use computers more than the "average" scientist (if in fact there is such a thing).

Second, most scientists generate and archive a few gigabytes of data each year. This answer was more popular than all the others together, which were "a few megabytes," "a few terabytes" and "more than a few terabytes." One thing we didn't ask (but should have) was how that data is archived: Is it stored in a Web-accessible database with searchable metadata, or on a DVD stuck in the bottom drawer of someone's desk? Personal experience tells us the latter is far more likely….

Third, most of the software that scientists work with is widely used: Only 10 percent reported that the programs they rely on are used by three or fewer people. When we asked where that software comes from, though, they reported "commercial off-the-shelf software," "open source" and "we build it ourselves" in almost equal numbers.

It's interesting to compare the latter answers with those given for another question. Fifty-eight percent of scientists reported that they do development on their own; 17 percent work with one other person, and 18 percent in teams of 3 to 5 people, while only 9 percent work in larger groups. These numbers are the reverse of what would be expected for professional software developers, who usually work in teams. They also explain the relatively low uptake among scientists of collaborative tools like version control, which most professional software developers consider essential: If you expect to work alone, why invest in tools for working with others?

The prevalence of solo and small-team work is consistent with another finding. Roughly 38 percent of the programs scientists write are between 500 and 5,000 lines long; smaller programs, and programs between 5,000 and 50,000 lines long, each make up about a quarter of the total, while larger programs account for the remaining 12 to 15 percent. To look at it another way, two thirds of the programs used by these scientists are less than 5,000 lines long.

The hardware scientists use is just as interesting. Eighty-one percent primarily use desktop machines; only 13 percent use intermediate-sized machines such as departmental Linux clusters, and a mere 6 percent use supercomputers. This is consistent with their reports about how they use computers: Most said that interactive use was most common, followed by preparing and reformatting data, preparing things for batch processing, and finally systems administration.

As for what occupied the most of our respondents' time, coding and debugging took first place. Planning and quality assurance tied for second place, reading/reviewing code came third, documenting fourth, and packaging software came last. It is ironic to compare this complaint with answers to another question: What "pain points" hurt you most? Lack of documentation was the number-one answer for more

than 40 percent of respondents, and in the top three for 80 percent.

Where do scientists learn how to develop software and use computers in their research? Almost all said that informal self-study had been most important. Peer mentoring came second, with formal instruction at school or on the job trailing well behind.

To close off, we wanted to find out how good scientists are at developing and using software. However, self-assessment is notoriously unreliable, and administering a proficiency test over the web would have been impractical. We therefore asked our respondents to rate how well they felt they understood various aspects of software development, and how important those aspects are.

The results were consistent with answers given to other questions. In most areas—requirements, design, maintenance, product management and project management—scientists reported that they knew as much as they felt they needed to know. This isn't surprising: Scientists are usually their own customers, and as our findings about team and program size suggest, those who develop software are creating small programs for their own use. Skills relevant to large projects done for other people are therefore unlikely to loom large in their minds.

The three areas in which respondents felt they didn't know as much as they should were, in order of increasing gap, software construction, verification and testing. Again, this isn't surprising, since the whole point of science is to be able to prove that your answers are valid--and that requires confidence in the methods and tools used to get them. The necessity of keeping test tubes clean and calibrating equipment is drilled into students from high school onward, but most are uncomfortably aware that we know a lot less about how to ensure that software is correct. The fact that there always seems to be one more bug to fix only reinforces the feeling.

Helping Those Who Need It

Our results can be interpreted in many ways, but I think two things are clear. The first is that if funding agencies, vendors and computer science researchers really want to help working scientists do more science, they should invest more in conventional small-scale computing. Big-budget supercomputing projects and e-science grids are more likely to capture magazine covers, but improvements to mundane desktop applications, and to the ways scientists use them, will have more real impact.

My second conclusion is that we're not doing nearly enough to teach scientists how to use computers effectively as research tools. One reason for this failure is that commercial software development tools and practices often don't fit the needs of people doing exploratory research in domains where years of training are required to understand the problems being solved. At the same time, university science and engineering departments feel their curricula are already overfull. As a physicist said to me some years ago, "What should we take out to make room for more programming—thermodynamics or quantum mechanics?" Figuring out how to square these circles is, in my opinion, the only grand challenge in scientific computing that really matters.

Acknowledgments

This work was made possible by a grant from The MathWorks, Inc. I'd like to thank my co-investigators, as well as Jon Pipitone and Dr. Laurel Duquette, who helped with data coding and analysis.

Bibliography

Hannay, Jo Erskine, Hans Petter Langtangen, Carolyn MacLeod, Dietmar Pfahl, Janice Singer and Greg Wilson. 2009. "How Do Scientists Develop and Use Scientific Software?" Proceedings of the Second International Workshop on Software Engineering for Computational Science and Engineering. New York: IEEE Press.