COLLECTIVE WISDOM

Brian Hayes

A reprint from

American Scientist

the magazine of Sigma Xi, the Scientific Research Society

Volume 86, Number 2
March–April, 1998
pages 118–122

COLLECTIVE WISDOM

Brian Hayes

The most powerful computer in the world, according to a recent ranking, is a machine called Janus, which has 9,216 Pentium Pro processors. That's a lot of Pentia, but it's a pretty puny number in comparison with the 20 million or more processors attached to the global Internet. If you have a big problem to solve, recruiting a few percent of the CPUs on the Net would gain you more raw power than any supercomputer on earth.

Of course the trick is to get all those millions of scattered machines working on *your* problem. The 9,216 Pentiums are all conveniently housed in a single room at the Sandia National Laboratory in Albuquerque. Setting them to work on the task of your choice is a simple matter; all you need is an account on the machine, a password, an allocation of CPU time, possibly a security clearance, and a little knowledge of programming in a specialized dialect of FORTRAN or C. Persuading the Internet to do your bidding is not so easy.

And yet it *can* be done. Consider the hunt for trophy-quality prime numbers. For two decades, the weapon of choice in this elite sport was a supercomputer—preferably the latest model from Cray Research. Beginning in 1979, the prime-number pursuit was dominated by David Slowinski and his colleagues at Cray (which is now a division of Silicon Graphics). The Cray team had to keep topping their own records, because they had so little competition elsewhere. In 1996, however, a new largest prime was found with an unpretentious desktop PC. The discoverer was a member of an Internet consortium who attacked the problem collectively with a few thousand computers. In August of 1997 another member of the same group found a still larger prime, which stands as the current record. Slowinski, being a good sport, offered one of his supercomputers to verify the discoveries.

The rise of cooperative-computing projects on the Internet is both a technical and a social phenomenon. On the technical side, the key requirement is to slice a problem into thousands of tiny pieces that can be solved independently, and then to reassemble the answers. The social or logistical challenge is to find all those widely dispersed computers and persuade their owners to make them available.

Brian Hayes is a former editor of American Scientist. *Address: 211 Dacian Avenue, Durham, NC 27701. Internet: bhayes@amsci.org.*

Recycled Cycles

What's most charming about collective computing is that it relies entirely on resources that would otherwise go to waste. The computing is done with spare CPU cycles on idling machines.

It is one of the wonders of our age that we squander vast quantities of computational labor. Forty years ago, when electronic computers were rare and expensive, CPU time was scheduled and billed by the millisecond. Now, computers spend most of their time displaying zooming multicolored windowpanes or simulating an aquarium. These "screen saver" programs, which compute nothing, whose only purpose is to stir up pixels on the display screen, probably consume more of the world's computational capacity than any other kind of software. Go into almost any office and you'll find machines busily saving their screens all night and all weekend.

Even when a computer is ostensibly in use, it is mostly idle. Typing furiously, you might produce 10 keystrokes per second; that's not much of a distraction for a processor that can execute 100 million instructions in a second. Under these conditions the processor spends most of its time going around in a tight little loop, asking over and over, like a fidgety toddler, "What can I do *now*?"

This waste of computational machinery is not something we need be ashamed of. The CPU cycles we fritter away today will not be deducted from the legacy bequeathed to our grandchildren. Still, every waste is also an opportunity, and the cycles you have no use for may prove valuable to someone else.

The idea of scavenging unused cycles arose almost as soon as computers were linked by networks. A few early experiments with distributed computing, including a pair of programs called Creeper and Reaper, ran on the ARPAnet, the

1970s predecessor of today's Internet. Later, when the Xerox Palo Alto Research Center (PARC) installed the first Ethernet, a program cruised the network at night, commandeering idle computers for CPU-intensive tasks. This early cycle recycler was the creation of John F. Shoch and Jon A. Hupp, who called it a "worm," citing the inspiration of John Brunner's novel *Shockwave Rider*. (A colleague, noting the program's nocturnal habits, suggested the alternative name "vampire.") A later scavenger system called Condor, developed by Miron Livny and his colleagues at the University of Wisconsin at Madison, is now running at more than a dozen universities and other sites. Condor roams within clusters of Unix workstations, usually confined to a single laboratory or department.

Going out over the public Internet to scrounge cycles is more difficult because the machines are more diverse and the network connecting them is more tenuous. Furthermore, communicating with the machines is only part of the problem; making connections with the machines' owners is also harder. Nevertheless, Internet "metacomputing" has been going on for at least a decade. Here are some of the notable projects.

Success Stories

Factoring. Finding the prime factors of a composite integer is a classic among computationally hard problems. The inverse process—multiplication—has an efficient algorithm we expect children to master, but factoring seems to be intractable when numbers get big. Adding three decimal digits to the length of a number doubles the effort needed to factor it.

But if factoring is hard, it is also ideally suited to parallel computation. Splitting the work among k computers produces an answer very nearly k times faster.

The first Internet factoring project was organized in 1988 by Arjen K. Lenstra (now of Citibank) and Mark S. Manasse of the DEC System Research Center in Palo Alto. They and their colleagues had written software to distribute factoring tasks among workstations within the DEC laboratory, and they extended this system so that computers elsewhere could contribute to the effort. The infrastructure was simple: Factoring tasks were parceled out by electronic mail, and results came back the same way.

As early as 1989 Lenstra and Manasse had already given an astute analysis of the economics of collective computing. They could get equivalent performance, they estimated, from 300 workstations or 1,200 PCs or a single high-speed machine designed especially for factoring. If they had to buy all the hardware, the last option was clearly the best choice. But if the owners of workstations or PCs could be induced to donate CPU cycles free of charge, that price would be hard to beat.

By 1990 Lenstra and Manasse and about a hundred e-mail collaborators from around the world were routinely factoring numbers of 100 decimal digits. In 1993 a larger group was assembled to take on a number known as RSA-129. Some 600 volunteers labored for eight months to factor this 129-digit number, winning a prize of $100 for their efforts. Two years later RSA-130 (a digit longer) succumbed to another army of factorers. This time some of the work was coordinated not by e-mail but through a World Wide Web interface designed by James Cowie of Cooperating Systems Corporation.

Primes. A counterpoint to factoring is the search for primes—numbers that cannot be factored. Primes of 100 or 200 decimal digits no longer attract much notice; with the right software any modern computer can find examples of such numbers in a few seconds. The frontier of knowledge for primes is now out near a million digits. Even the mathematically jaded may marvel at a prime of such magnitude: Think of all the smaller numbers that *might* divide it, but don't.

Most of the largest known primes are Mersenne primes, named for the 17th-century friar Marin Mersenne. A Mersenne number has the form 2^n-1, but not all such numbers are prime. In the first place, for 2^n-1 to be prime, n itself must be prime, but even that is only a necessary condition, not a sufficient one. (Try $n = 11$.) To find Mersenne primes, you must first calculate 2^n-1 for each prime value of n, then test the result for primality. Algorithms based on the work of Edouard Lucas and D. H. Lehmer greatly speed up the primality tests.

The two largest known primes were found by participants in the Great Internet Mersenne Prime Search, or GIMPS. The founder of this project is George Woltman, a computer scientist who wrote efficient software for Lucas-Lehmer primality tests and made it available on a Web site. (You need not understand the mathematics of the Lucas-Lehmer test to run the software.) Some 4,000 volunteers have contributed to the search so far. In November 1996 Joel Armengaud discovered that $2^{1,398,269}-1$ is prime. Then in August 1997 Gordon Spence proved the primality of $2^{2,976,221}-1$. This number, which has 895,932 decimal digits, is the 36th Mersenne prime to be discovered. (It may not be 36th in the numerical sequence of Mersenne primes, however, because there are exponents less than 2,976,221 that have not yet been checked.)

Code-breaking. The collective-computing projects that have attracted the most participants have been attempts to decipher encrypted messages—but the volunteers are not snooping into anyone's confidential e-mail. RSA Data Security, a company in the secrecy business, has posted a number of cryptographic puzzles, with cash prizes for those who solve them. The company's aim is to test the security of their own products and to demonstrate the vulnerability of encryption schemes they consider inadequate. The factoring of RSA-129 and RSA-130 was part of this program. Other RSA

challenges don't involve factoring but call for a direct attack on an encrypted text.

In one challenge the message was encoded with DES, the Data Encryption Standard, a cipher developed in the 1970s under U.S. government sponsorship. The key that unlocks a DES message is a binary number of 56 bits. In general the only way to crack the code is to try all possible keys, of which there are 2^{56}, or about 7×10^{16}. This task was undertaken by an Internet collaboration called DESCHALL, organized by Rocke Verser, Matt Curtin and Justin Dolske. In June of 1997 they got lucky, discovering the correct key after checking just 18 percent of the possibilities, and won the $10,000 prize.

Another RSA challenge also employed a 56-bit key, but with an encryption algorithm called RC5. Three groups, known as Bovine, Infinite Monkeys and Cyberian, all began recruiting volunteers for the RC5 attack. Bovine eventually attracted the most helpers, and it was they who found the key and deciphered the message in October 1997, after exhausting 47 percent of the key space, or 34 quadrillion keys. Bovine was organized by Adam L. Beberg, Jeff Lawson and David McNett.

Compared with earlier distributed-computing projects, the RC5 efforts were not only technically sophisticated but also reached a new level of promotional and motivational slickness. For example, the Bovine software kept statistics on the contributions of individuals and teams, adding an element of competition within the Bovine ranks as well as between Bovine and the other groups. In the team standings, Macintosh militants finally prevailed over partisans of the Linux operating system. By the end of the contest some 4,000 active teams were processing 7 billion keys per second, a rate equivalent to the work of 26,000 Pentium computers.

On completing RC5-56, the Bovine collaboration turned to RC5-64, a cipher with a 64-bit key. The effort needed to break this code will be 256 times greater, which suggests it could be a labor of decades. It's worth pausing to ask whether the brute-force testing of 18,446,744,073,709,551,616 cryptographic keys is really a better use of resources than displaying pretty fish in an aquarium. Beberg and his colleagues are considering other possible projects. Meanwhile, RSA has announced a new challenge. This time the message is encoded with the same DES algorithm broken last spring, but the contest rules are altered to reward speed of decryption. The initial prize of $10,000 drops to zero after 67.5 days. Bovine has taken up the challenge.

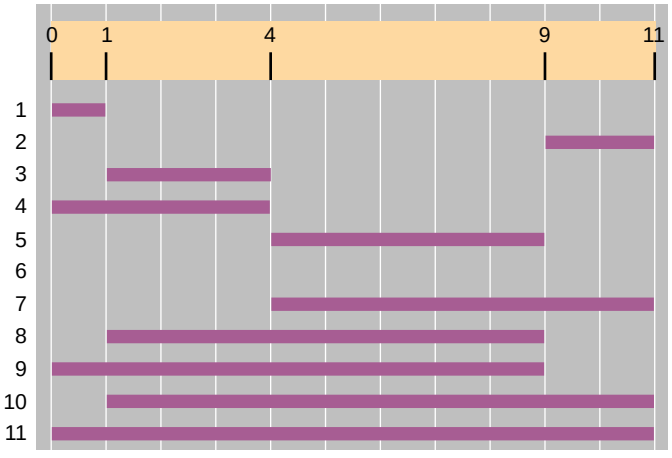

Figure 1. Optimal five-mark Golomb ruler is 11 units long and measures 10 of 11 possible distances; the ruler has no pair of marks six units apart.

Golomb rulers. Imagine a six-inch ruler with marks inscribed not at the usual equal intervals but at 0, 1, 4 and 6 inches. Taking all possible pairs of marks, you can measure six distinct distances: 1, 2, 3, 4, 5 and 6 inches. A ruler on which no two pairs of marks measure the same distance is called a Golomb ruler, after Solomon W. Golomb of the University of Southern California, who described the concept 25 years ago. The 0-1-4-6 example is a *perfect* Golomb ruler, in that all integer intervals from 1 to the length of the ruler are represented. On rulers with more than four marks, perfection is not possible; the best you can do is an *optimal* Golomb ruler, which for a given number of marks is the shortest ruler on which no intervals are duplicated.

The world is not panting in desperate need of better Golomb rulers, and yet these combinatorial curiosities do have practical uses. For example, in setting up an interferometer for radio astronomy, placing the antennas on the marks of a Golomb ruler maximizes the recovery of information about the phases of the signals received.

Many good Golomb rulers have been found by trial-and-error, but proving them optimal (or finding a better ruler if one exists) is computationally expensive. In 1995 a dozen workstations took four CPU-years to prove there is no 19-mark ruler shorter than 246 units. The computation was done by Apostolos Dollas, William T. Rankin and David McCracken of Duke University.

In 1996 David Vanderschel and Mark Garry, who had both worked on Golomb rulers independently, merged their ideas in a program called GVANT, which turned out to be significantly more efficient than earlier programs. They quickly confirmed the known results for rulers of up to 19 marks, but even with their highly optimized algorithm a search for the best 20-mark ruler was a formidable undertaking. They therefore sought Internet collaborators. With seven helpers it took about half a year to prove that a 20-mark ruler of length 283 is optimal.

In the spring of 1997 Vanderschel and Garry turned to the 21-mark ruler, for which the shortest known arrangement is 333 units long. For this ruler a naive algorithm would have to check more than 10^{30} arrangements of the marks; GVANT prunes the number of cases to about 10^{15}. Roughly 100 volunteers have pitched in to help, but after a year's searching there is still plenty of work left for latecomers. Out of 1,200

trillion arrangements to be checked, fewer than 200 trillion have been examined so far.

Aliens. If the prospect of finding a bigger Mersenne prime or a smaller Golomb ruler won't induce you to pledge your CPU, perhaps you would like to discover an extragalactic civilization. A proposal called SETI@home would put thousands of computers to work sifting for signs of life in signals recorded with the radio telescope of the Arecibo Observatory in Puerto Rico.

A search for extraterrestrial intelligence has been going on at Arecibo for almost two decades. The telescope slowly scans the sky, detecting emissions over a broad range of radio wavelengths. A special-purpose signal-processing computer applies a Fourier transform to the data to pick out narrow-bandwidth signals, which could be the alien equivalent of "I Love Lucy." The astronomers would like to put the data through a more thorough analysis, but computing capacity is a bottleneck. That's where SETI@home comes in.

With enough computers on the job, the Arecibo data could be sliced into finer divisions of bandwidth and frequency. Moreover, the analysis software could check for other kinds of regularities, such as signals that pulsate or that "chirp" through a sequence of frequencies. The task is well suited to Internet computing in that only small blocks of data need to be passed back and forth over the network, but a great deal of computing is needed on each block.

SETI@home is the project of Dan Werthimer of the University of California at Berkeley and several colleagues. Their plans are ambitious; they seek a mass audience. The client software they envision would run as a screen saver, starting up automatically whenever a machine is left idle. As the program churned away on the data analysis, it would also display a series of images related to the project, such as a representation of the data currently being examined or a map showing the progress of the sky survey.

Some 70,000 people have signed up for SETI@home. Unfortunately, the project is on hold for lack of funding.

Cycles to Spare?

Jussi Kallioniemi, the founder of the Cyberian consortium, writes: "Every second billions of innocent assembler instructions are executed all over the world." It seems there is no hope of stopping the slaughter, but if you would like to make their deaths more meaningful, here are some organizations that accept donations of CPU cycles. In most cases software is available for computers running Microsoft Windows, for Macintosh and for some varieties of Unix.

The Great Internet Mersenne Prime Search: George Woltman is coordinating a search for all Mersenne primes with an exponent less then 5,260,000. The Web site provides links to excellent background on the mathematics of primes and Mersenne numbers.

http://www.mersenne.org/

Golomb rulers: Mark Garry, David Vanderschel and about 100 volunteers are searching for a 21-mark Golomb ruler shorter than 333 units; if no such ruler is found, that will constitute a proof that the 333-unit example is optimal. The job should be finished within a year.

http://members.aol.com/golomb20/index.html

distributed.net: This organization, whose slogan is "fastest computer on earth," formed Project Bovine to win the 56-bit RC5 cipher challenge. As of January 1998 the members' efforts are being directed into a short-term contest to read a message encoded with the DES cipher. Following that, work will resume on a 64-bit RC5 message.

http://www.distributed.net/

SETI@home: 100,000 computers could be kept busy trying to extend the reach of the global Internet (as if we didn't have enough trouble already running out of area codes and IP numbers). But the most urgent need is not for CPU cycles; the current search is for a major funding source.

http://www.bigscience.com/

Unparalleled Parallelism

Factors, primes, codes, rulers—some of these projects sound like they might belong in the *Guinness Book of World Records*. They're not frivolous, but they're not quite in the mainstream either.

There are plenty of other areas of science and engineering that could benefit from cheap and abundant computing. The traditional big consumers of CPU cycles include the analysis of seismic data, simulations of many-body systems, studies of protein folding and other kinds of computational chemistry, studies of turbulent fluid flow, and lattice models of quantum field theories. Could such tasks be shared over the Net?

When viewed as a massively parallel computer, the Internet has a peculiar architecture. It is extraordinarily rich in raw computing capacity, with tens of millions of processors. But the bandwidth for communication between the processors is severely constrained. The 9,216 Pentiums of the Janus computer can talk to one another at a rate of 6.4 billion bits per second; for a node connected to the Internet by modem, the channel is slower by a factor of 100,000.

The limits on bandwidth determine what kinds of algorithms run smoothly when spread out over the Net. Consider the case of an *n*-body simulation, which describes the motion of particles in a force field, such as stars in a galaxy or atoms in a fluid. One parallel *n*-body algorithm assigns each particle to its own processor, which then tracks the particle's path in space. The trouble is, each processor needs to consult all the other processors to calculate the forces acting on the particle, and so the volume of communication goes up as n^2. That won't fly on the Net.

Yet *n*-body problems are not necessarily unsuited to network computing. There are other *n*-body

algorithms, and other ways of partitioning the problem. In particular, "tree codes" organize the computation hierarchically. At the bottom of the hierarchy a processor calculates motions inside a small cluster of particles, without reference to the rest of the universe. At the next level several clusters are combined, ignoring their internal dynamics and looking only at the motions of their centers of mass. Then clusters of clusters are formed, and so on. Tree codes are popular for *n*-body computations, but whether they can be adapted to Internet computing remains to be seen.

Memory capacity is another serious constraint. Computer owners who are willing to give away CPU cycles may be less eager to let someone fill up their machine's memory or disk drive. Both the bandwidth and the memory limits will be difficult hurdles for programs that operate on large volumes of data, as in seismic analysis or weather prediction.

And yet there are powerful incentives for clearing those hurdles. In round orders of magnitude, a typical personal computer will soon execute 100 million instructions per second; it will have 100 megabytes of memory and a gigabyte of disk storage; it will consume 100 watts of electricity and cost $1,000; 100 million of these machines will be attached to the Internet. Multiply it out: 10 quadrillion instructions per second, 10 billion megabytes of memory, 100 million gigabytes of disk storage, 10 gigawatts of electric-power demand, a price tag of $100 billion. It's probably worth rewriting your software to gain access to such a machine.

Collectivists and Capitalists

And what about incentives for the owners of that $100 billion distributed computer? The spirit of volunteerism is a wonderful thing, but it doesn't always scale well. If Internet computing ever catches on in a big way, we'll probably hear less about cooperatives and collectives, and more about return on investment.

One way of paying for Internet computing is through a commodity market in CPU cycles. If you have 100 computers with nothing to do nights and weekends, you offer the spare capacity at an asking price expressed in millicents per trillion instructions or dollars per Pentium-year, or some such fabricated unit. Meanwhile someone with a big batch of numbers to crunch enters a bid for a stated quantity of computation, measured in the same units. An automated clearinghouse matches up buyers and sellers.

It could be fun to watch the fluctuations of such a market. When Lucasfilms needs half the processors in the galaxy to render scenes for the next *Star Wars* saga, prices shoot up. Over academic holidays, excess capacity brings on a seasonal slump. The market is likely to be volatile, because CPU cycles are like electricity or fresh asparagus: You can't stockpile them for later use.

Trading in CPU cycles is not a new idea. As early as 1968 Ivan Sutherland wrote of a "futures market in computer time"—although his market consisted of only a whiteboard and colored pens. The market mechanism was explored in greater depth at Xerox PARC, in an experiment described by Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart and W. Scott Stornetta. For the most part their market worked as economists would predict—each job's share of the total machine time was proportional to its funding—but they did observe some interesting undamped oscillations in prices.

Could a real market, backed by real money, evolve on the Internet? Questions of security and confidentiality would need to be addressed. When I send you my program to run, how do I know you won't pry it open and steal my secrets? How do you know my program won't steal *your* secrets? (The answer to both questions is probably Java.) Another essential precondition is that CPU cycles have to become what economists call a fungible asset, meaning that cycles on one computer are readily converted into those on any other. This is a hard problem but not insurmountable.

In the end, though, the economics of the global metacomputer are probably less interesting than the operation of the machine itself. Given a free flow of computations through all those 100 million nodes, what would such a device wind up computing? I hope that we as a civilization can find a better use for this machine than we have for Janus, whose primary function is to simulate the explosion of nuclear weapons.

Bibliography

Dollas, Apostolos, William T. Rankin and David McCracken. 1998. A new algorithm for Golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory* 44:379-382. http://www.ee.duke.edu/~wrankin/golomb/golomb.html

Golomb, Solomon W. 1972. How to number a graph. In *Graph Theory and Computing*, ed. Ronald C. Read. New York: Academic Press.

Greengard, Leslie. 1990. The numerical solution of the *N*-body problem. *Computers in Physics* 3:142–152.

Hayes, Brian. 1994. The magic words are squeamish ossifrage. *American Scientist* 82:312–316.

Lenstra, Arjen K., and Mark S. Manasse. 1990. Factoring by electronic mail. In *Advances in Cryptology, Eurocrypt '89*. Lecture Notes in Computer Science Vol. 434. New York: Springer-Verlag. pp. 355–371.

Litzkow, Michael J., Miron Livny and Matt W. Mutka. Condor—a hunter of idle workstations. Eighth International Conference on Distributed Computing Systems, San Jose, California, June 13–17, 1988, pp. 104–111.

Mattson, Timothy G., and Greg Henry. 1998. The ASCI Option Red supercomputer. *Intel Technical Journal*, 1st quarter 1998. http://developer.intel.com/technology/itj

Shoch, John F., and Jon A. Hupp. 1982. The "Worm" programs—early experience with a distributed computation. *Communications of the ACM* 25:172–180.

Sutherland, Ivan E. 1968. A futures market in computer time. *Communications of the ACM* 11:449–451.

Waldspurger, Carl A., Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart and W. Scott Stornetta. 1992. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering* 18(2):103–117.