Tales of zilla: Adventures in distributed computation

by Richard E. Crandall
Director, Scientific Computation Group
NeXT, Inc.

I have in the last year gained a powerful new colleague: an intelligent beast equipped with a supercomputer brain, but also possessed of a certain flair for etiquette. This creature I named "zilla", an appellation not entirely specious. I understand that in modern Japanese folklore, Godzilla was not fundamentally aggressive, but only attacked when the situation called for intervention. The new creature is not of flesh and blood. It is a collection of NeXT Computers participating in distributed computation.

The first implementation of zilla used most of the office computers in NeXT's Software Division. The initial research success of zilla was to resolve special cases of Fermat's "Last Theorem," one of the very oldest, but still unsolved, conjectures about numbers. I shall describe this experiment below, but first let me turn to a description of the beast.

The zilla network is computationally powerful. We denote by one "zilla unit", or z.u., the equivalent of 50 NeXT Computers. Now one z.u. has 250 MIPS capability, which puts it roughly in the supercomputer region (for problems not requiring Cray-type high-speed throughput). There is a total of 400 megabytes of physical memory, and upwards from 1 gigabyte of virtual memory, the latter depending on how the distributed programs allocate memory.

The idea is to have one master server machine with a running slave thread that scans through a list of "permitted machines." The slave thread uses the Unix rcmd(3) library to remotely execute processes on slave machines. In the initial experiments, each machine dealt with a separate case of Fermat's "Last Theorem," feeding the results to destination files of choice. Usually these results would accumulate in a specific NFS (network file system) directory, but distributed storage is also possible.

Now we come to the friendly character of zilla. Every NeXT Computer screen dims automatically if one leaves one's machine dormant for a (settable) period of typically ten minutes. The etiquette strategy is devilishly simple: if the screen has dimmed, for example at the end of a working day, zilla intervenes and performs calculations; whereas if the screen is brightened via keyboard or mouse events, zilla backs off. In this way, calculations were carried out during our initial experiments of 1989, with office personnel noticing very little, if any, zilla intervention. The initial calculations, which I shall next describe, would require about 500 CPU hours on a Cray-YMP. Depending on industrial rates for such time, the conventional cost of such a calculation would be on the order of 10^5 dollars. Given that zilla provided no untoward interference to office workers during the project, our cost was virtually zero.

Now to the numerical research. Pierre Fermat conjectured in 1637 that for n>2 the equation

$$x^n + y^n = z^n$$

has no solutions in positive integers x, y, z. One learns in elementary geometry simple identities such as the Pythagorean relation $3^2 + 4^2 = 5^2$, but Fermat's "Last Theorem" refers to exponents greater than 2. There is romance connected with this problem for two reasons. The first is that Fermat wrote a marginal scrawl, in his Bachet edition of the works of Diophantus, that he had a proof which the margin was too small to contain. We shall probably never know whether he actually had a proof. The second reason is simple: to this day, nobody has a proof, and there is no known counterexample. The "Last Theorem" persists as sublimely difficult. I am sometimes asked what the use is for such a theorem. For one thing, attempts to conquer it have opened whole new fields of inquiry. For another, there is no telling where *any* intellectual idea may lead.

We investigated various Fermat exponents n on zilla. Each machine was given an n and had to compute the reciprocal of a polynomial of degree (n±1)/2. So each machine had to compute an expression

$$1/(ao + a1\ x + a2\ x^2 + ...)$$

where the denominator could have as many as half a million terms.

The allocation of a polynomial p (the denominator in the previous polynomial fraction) invokes about 4 megabytes of virtual memory. All of this for one polynomial, but each case of exponent n involved tha allocation of a handful of such polynomials. Each machine would set up the polynomials, use Newton's iteration and Fast Fourier Transform (FFT) multiplication to find the polynomial reciprocal, and report either "exponent n is conquered" or "a little more proving must be done." After all primes n between 3 and 1,000,000 were checked, we had established that many n were resolved (i.e., $x^n + y^n = z^n$ is impossible), with some fraction of the n values awaiting one final check which was itself "zillafied." The resulting data, which is being prepared for publication by theoreticians on the project, so far supports the "Last Theorem."

According to zilla, Fermat's "Last Theorem" is true for all exponents less than one million. Previous published tables have only reached 150,000. As an exercise of virtual memory capability, some random, very high exponents were analyzed also. For example, zilla claims that

$$x^{2000291} + y^{2000291} = z^{2000291}$$

has no positive x, y, z solutions. The 1 z.u. zilla network can provide about fifty such results per hour.

To my mind, the superb virtual memory performance of zilla is the most important aspect for research, followed closely by arithmetical speed. It is not difficult to simply append via TCP/IP protocol a machine such as a Cray (provided one has an account, and time, etc.) to zilla's permission list. It is a tribute to the beast that a single such appendage typically increases the computational power only by a small factor.

What is more, the zilla concept is suitable for other distributed processing such as audio paging systems and parallel ray-tracing. I am working on turnkey zilla application so that this wonderful adventure may

continue in a more practical vein.