WIRED

One Huge Computer

By Kevin Kelly and Spencer Reiss

The Net made it possible. Java made it doable. Jini might just make it happen. An on-the-fly, plug-and-work, global nervous system that connects his cam to her RAM to your PDA.

Also: A conversation with Sun's founding spirit Bill Joy.

The Irresistible Dream: Ever since Marshall McLuhan, a central dream of the digital culture has been to create one huge computer. Not a towering superbrain tended by white-coated priests, but a vast constellation of interacting machines - processors, memory modules, disk drives, and a million other devices, all networked into a vast planetary system. A means of thinking, creating, and communicating that is everywhere at once, but nowhere in particular. A computer that is always on. Such a system would continuously spread itself and thicken, expanding by its own internal logic. It would be supremely adaptable, and hard to break. It would have myriad access points, but no CPU, no single point of failure. The global village, to coin a phrase, made real.

Engineers have long had a word for systems whose powers are widely dispersed: distributed. Banking, telephones, the electric power grid - the bigger something is, the more likely that it will be distributed. The Internet is arguably the biggest distributed system ever built, and the most complex. But all these are specialized, essentially one-dimensional undertakings - processing money, electricity, or communications bits. They pale against the ambitions of a system that aspires to be everything - to everyone.

For the biggest of thinkers, that sets up an irresistible dream: to build the network that makes all networks one, a global nervous system. The napkin sketch is simple: Take all the intelligent machines in the world - from giant mainframes to the tiniest embedded chip - and hook them together in a single intelligent network. A system open to novelty, new members, and features. A system that can tolerate what engineers ruefully call faults. A system with no limits on how large it can get, nor how small its smallest part can be.

Add a few more stipulations. To have any chance of working, the global network's structure will need to unfold from simple principles, rather than from ever more complex planning and central control. And, like another well-known distributed-computing device - the human brain - it will need to be able endlessly to reconfigure itself, to solve unanticipated problems and address unforeseeable new needs.

The key pieces for such a system - millions and billions of microprocessors - are already here, or coming. So, too, are the riotously expanding networks. Indeed, to start building that one great computer, only a single essential ingredient is missing: an architecture, a universal language, a set of superprotocols, something - and very possibly today's lexicon can't name it - to hold it all together and let the magic work. A constitution, if you like, a digital equivalent of the genetic code that all living things share.

Or, just maybe, this: a crash effort cooked up by some of the most ambitious minds ever to flee the corporate confines of Silicon Valley - a secret project spearheaded by Bill Joy, the software luminary

who put the Internet on Unix and Java on close to 100 million desktops and whose fondest wish now is to give the world, to use a favorite Joy phrase, one more good "technological dislocation." He's sure he's found one. And appropriately, it's called Jini, loosely from the Arabic for magician.

General magic

In a windowless second-floor room in a deliberately obscure Sun Microsystems outpost in Sunnyvale, California, half a dozen anonymous chunks of expensive-looking hardware sit on long folding tables. Some barely rate a first look: a not particularly recent printer, what look like a pair of flat-screen monitors, a video camera, a couple of keyboards. Others are clearly prototypes: overdesigned purple computer-somethings with curved sides and stylized vents. Any Demo, Silicon Valley, USA.

Turn any of the devices around, however, and only two wires are visible: electric power and an RJ-45 Ethernet connection. Each box - even the display screens and the little handheld camera - is a fully independent network citizen, able to hold its own on the system, unencumbered by specialized cables, software drivers, or the rest of the usual array of digital life support.

Say you want to use the camera. Plug it in, and poof - a second later, an icon appears on your display screen. All the configuration chores are done automatically by one of those purple boxes - a low-end server called a lookup device - and by a 25K communication program in the camera. What's in the viewfinder? Bring the camera image up on a monitor - any one you like. Store a clip? The 10-gig storage device - a slightly smartened-up disk drive - is waiting. Edit? There's another of those purple boxes, the computing device, with full workstation power. Pull some video-edit software out of the storage module, and you're off.

That's one possibility. Or maybe you'd rather batch print some letters from your laptop. Done. Or get that old laser printer online. A pocket-sized adapter does it. Or add another 10 gigs of storage - no need to call a sysadmin, just grab a drive off the shelf, and plug it in.

On one level, the demo is the ultimate in plug-and-play technology - "plug-and-work," its Sun-shirted minders note with a smile. No mean feat. Not surprisingly, some of the Jini demo's most interested visitors have been from hardware companies that would dearly love to find a way for us all to snap a few billion more microprocessors, disk drives, and other smart devices into our personal networks.

But Jini aims much higher. What Joy and the two dozen programmers working with him aspire to do is nothing less than dynamite the whole creaky logjam of computing, as it has evolved from giant mainframes through the first clunky PCs to today's cobbled-together Internet and Windows Everywhere. If they succeed, Jini code will provide connections that will make today's information "superhighways" look as confining as 19th-century railways. And that, Jini thinking goes, will be the foundation for truly networked, global computing - organic and ever changing, and keyed to a hurtling future instead of being shackled to the platforms and conventions of the past. "When the foundations are so far off," reads an internal Sun document written to support the project last year, "it makes sense to do a reset."

Coming from almost anywhere else, that declaration would be laughable. But Sun and Bill Joy have come close once already to pushing computing's reset button, with its still-expanding programming language Java, the most important development in computing since the explosion of the Internet. What Java aims to do for software - be a lingua franca - Jini hopes to do for the machines that run it: provide an overarching, universal platform - a distributed operating system, in effect, on which devices of every description can meet. "Jini is the next chapter in the Java story," reads another project mantra.

And Jini is no clunky hack, strung together in a lab with glue and wire to impress the boss and calm investors. Most of the demo devices are modified versions of existing hardware - one of the project's

driving ideas is to not have to throw existing systems away. Jini software has been in limited-release beta since June, with testing under way by some of the biggest names in computers and consumer electronics - NEC, Toshiba, Quantum, Ericsson, Siemens, Computer Associates, and a dozen others. By the end of the year, Sun hopes to release a full package, from a network infrastructure to the little 25K program that can put your front-door light switch onto the network. The release name is still being debated, but the marketing plan is not: It will reprise the same strategy that fueled the explosive take-offs of both the World Wide Web and Java - essentially, give it away. "There's one thing we've all learned from watching Java and the Net," says Mike Clary, Joy's key colleague in Aspen and Jini's overall project manager. "This can only be a ubiquity play."

Jini's prelaunch team shares a building with what remains of another audacious attempt at networking heroics, General Magic - a reminder of the casualty rate of would-be technological revolutionaries. A Jini victory would mean the creation of a loosely connected federation of computers freed from today's OS tyrannies - one reason not to expect a friendly Microsoft embrace. Neither Bill - Gates or Joy - needs reminding that it was the modest little PC's universal appeal, not the US Justice Department, that ultimately humbled IBM's mighty mainframes. And if lightning strikes again, those anonymous boxes in the windowless demo room could someday end up in a technology museum: cell zero of the global computer. Not to mention giant slayers.

If ...

Up from Java

Bill Joy doesn't like the word "exile," but he's made a second career out of keeping most of the Rocky Mountains between himself and Silicon Valley. A founder of Sun Microsystems and still officially Sun's VP for research, Joy took himself to Aspen a decade ago to build a geek-lord's dream: his own custom research-and-development lab, aka Sun Aspen Smallworks. Small? "Ideas resemble the organism that built them," Joy says, "so a small organization will build simple things that work." Meaning what? "The idea is that we do whatever is most important - not necessarily most urgent. Sun has 20,000 other people doing that. I left the urgent behind to get to the important."

In the early days, Joy and a rotating Smallworks crew focused on what they dubbed the "4MY" program - "Four Miracles a Year," everything from microchip design to networking theory. More recently, Aspen was a refuge for the long-running project that became Java, from its early near-death experiences as "Green" and "Oak" to the first big licensing deals.

It's a pleasant place, Smallworks, behind one of those too-cute Victorians above the year-round commotion of shops and restaurants in downtown Aspen. Joy and a couple of permanent staffers inhabit a cheerful clutter of exotic gear, whiteboard, and piles of books. But blissed-out the view definitely ain't - the view of the high tech landscape, anyway. "We're in the Dark Ages," Joy says, wheeling out his favorite rant. "It's 900 AD - medieval computing. Except for the Web, what's really getting better? I managed to get my notebook computer to talk to the printer - it took a month. Our basic operating systems now have some 20 million lines of code, and more is being piled on every day. It's insane to try to build the future on that."

Indeed, from whichever angle you look - Silicon Valley prince or baffled user - complexity and scale are the mad aunts in the attic of today's computing. Lines of code piling up like crust on hard drives are only part of the problem - the real nightmare starts when you add blossoming networks to the mix. Systems engineers measure complexity with a metric: number of users times number of machines times number of functions being undertaken. Put a couple of those numbers into the millions or billions - which the Net explosion is doing - and you get unmanageably huge, quickly. Unless, of course, you have a system that can pull order from networking chaos.

Visionaries and hard-headed engineers - not to mention Windows-for-all Gates - have been groping for years to find paths through the spreading complexity. Ted Nelson's Xanadu, Xerox PARC's Smalltalk, David Gelernter's Linda; the list is long and not encouraging. One general path has been idealized - start-from-scratch systems, most of them quixotic or mainly research ventures. Another, less sweeping approach has been object-oriented programming - building applications on the fly from small code modules, usually called objects, the better to move them around a network or translate across platforms. Two rival object standards, the industrywide Corba and Microsoft's DCOM, have kept sprawling corporate networks from degenerating into towers of Babel. And then, of course, there's the one unalloyed success story of distributed computing: the Internet, and its prodigal, the Web. Ironically, though, TCP/IP's very success in creating a global medium has only made the overall problem of complexity even worse.

As the Net's explosion gained force three years ago, Bill Joy was deep into Sun's own object-oriented programming effort. The motives for releasing Java - an elegantly stripped-down language originally designed to run consumer electronics - were less than pure and more than a little desperate: to blunt "Windows Everywhere!" with a new technology that promised platform independence. The ability to run the same program on any computer - Mac, Windows, Unix, a tiny device on your wrist - is a key distributed-computing tenet, not to mention an obvious boon to a global information network. Skeptics laughed nonetheless. But the timing was perfect - even more so when Netscape, looking for allies and ammunition against Microsoft's gathering counterattack, built Java compatibility into its runaway-hit browser. What might have been another high-minded experiment instead became an instant global standard.

Sun from the start has famously been the company that preached "the network is the computer." But even for Joy, holed up in Aspen writing the Java specs, that explosion was astonishing. Though Java was launched as a new programming language, Joy and the others had always assumed that they would slowly build it into a full software platform - one that really fulfilled the brash early promise of "write once, run anywhere." Their best guess had been that it would take five years to achieve what they reckoned was the critical mass needed to launch a viable distributed platform - about 100 million users. But the Net's amazing growth had them scrambling almost immediately. The good news was that Joy and the rest of Sun's software research team already had a clear sense of where they wanted to go. "We knew that whatever we did had to be technically simple," says Joy, "because it's hard to write programs, and even harder to write distributed programs - you have the whole big complicated system to think about. What we wanted was a very simple communications mechanism that would let the distributed system work."

One of Joy's favorite engineering maxims - "Large successful systems start as small successful systems" - is another way of saying: Use what already works. In 1994, the Aspen skunkworks already had a workstation running Oberon, an ambitious attempt by Zürich-based Niklaus Wirth, the inventor of Pascal, to create a featherweight system written entirely in one simple programming language. Such knowledge-based computing erases the conventional distinction between the OS and applications. Building distributed networks, Joy believed, was a key breakthrough. Another intriguing model was Gelernter's Linda, whose central idea, called "tuple spaces," is a radically simple way to organize communication between software objects; Linda's broad concepts had already been adopted for JavaSpaces, a tool for building distributed applications.

And then there was Java itself, which continued to build momentum among programmers - and with that, more and more of the plug-and-play software components crucial to making object-based programming work.

In the spring of last year, Joy sat down in Aspen with Sun senior staff engineer Jim Waldo, whose research group had just completed Java RMI - Remote Method Invocation, an interface tool that lets distributed software objects find and communicate with each other over a network. Sketching on - yes - a napkin, they realized that the practical outlines for a full-out distributed-computing system were already visible. They also had the people, based mainly in Sun's East Coast software research lab in Chelmsford, Massachusetts. Waldo himself had already started the basic code for what programmers call transactions, which ensure that groups of commands sent out over the network actually occur as a

unit. A variety of programmers had worked out leasing, a framework for short-term relations between objects. Bob Scheifler, a leader of the X Consortium - an industrywide initiative to build cross-platform interface technology - had the network-security know-how. "Two coffees into breakfast," recalls Joy, Jini was in high gear.

Reality check

You have to drill down energetically into Microsoft's sprawling Web site, but there it is, in the list of projects under way at Microsoft Research. "We believe it is time to reexamine the operating system's role in computing," reads the opening line of a proposal for an initiative dubbed Millennium, described as "a new self-organizing, self-tuning distributed system." NT's 20 million-odd lines of code notwithstanding, Millennium envisions "a distributed operating system, based on a few principles pervasively applied." As part of that system, "any code fragment might run anywhere, any data object might live anywhere." Sound familiar? It would also be "self-configuring, self-monitoring, and self-tuning. And of course, it would be scalable and secure." Of course.

In the long tradition of Microsoft vaporware, there may be less to Millennium than meets the eye - the team consists of a half-dozen full-time researchers, according to a spokesperson, and a couple of active prototypes. One working system, dubbed Coign, distributes conventionally written applications on the fly; the other, Borg, creates a distributed version of the Java Virtual Machine. Microsoft famously got jumped once before by a technology, the Internet, that didn't quite fit Redmond's worldview; despite all of his current distractions, Bill Gates doesn't want to get paradigm-shifted once again. Whatever Millennium turns out to be - vaporware stalking-horse or shrewdly hedged bet - the Kremlin of centrally planned computing has more reasons than most to be paying attention to new rumblings on the network.

And there's not just Bill Joy and Sun to worry about. Add to the list Lucent's ever about-to-take-off Inferno; an ambitious Caltech project called Infospheres; even Larry Ellison's half-baked network computer scheme - all are pursuing the distributed-computing dream. Even sleepy AT&T this spring unveiled a Java-based "enhanced network infrastructure" called GeoPlex, designed to let telecom companies offer services across the whole array of digital devices and networks. Apparently you don't need to be a software hero with a private Aspen research lab and 20/20 programming vision to detect a potential revolution.

So ... why Jini?

The short answer, of course, is Java, whose slipstream - a million active programmers, by Sun's latest reckoning - can give Jini the kind of instant presence and easy learning curve Java got from Netscape and C++. A quiet argument is under way in Jini's marketing team over how closely to stick to Java branding; the leading contender has been JavaTone, as in the universal telephone signal.

But Jini is primed to ride a potentially even more powerful new wave: hardware geared for the network. Jini's main beta testers are not the usual Silicon Valley coders - licensees already signed up include a dream team of big-time hardware players who seem to be falling over each other with raves. "Anyone who's ever tried adding storage on a LAN can tell you why we need this," says Paul Borrill, vice president and chief architect at Quantum, the disk drive maker. "To use an overused phrase, this is a paradigm shift." Quantum expects to ship its first Jini-ized devices late next year. Billy Moon, Ericsson's New Concepts program director, goes one shift better: "It's a double-barreled paradigm shift that reaches beyond the computer industry. The combination of componentized software running on distributed virtual machines and the bold system architecture transform and blur the very idea of what computers, networks, and applications are."

Things get vaguer when the question turns to the new services that Jini could spawn. Plug-and-play is a nice feature. Exploding the computer back to its components - storage and processing especially - is

a potential revolution, opening the door to everything from supercomputing on demand to massively encrypted remote data storage and your own personal desktop available on any machine in the world. Clever corporate marketers and ecommerce entrepreneurs presumably will sort these offerings out.

Jini avoids one common stumbling block of many clean-slate solutions: incompatibility. Specialized programming languages, legacy applications, and hardware all do fine under a Jini régime; the only requirement, beyond being Java-enabled, is that they observe the basic networking rules. "The whole idea is to be very forgiving," Joy says. "If you have slightly different code than I do, that's fine - when I get one of your objects, I also get the code that goes along with it. We don't have to agree beyond the basic rules, and we can let the best - the most functional, fastest, easiest - code win. So you can keep your Windows if you want it. But now the network will be evolutionary - the survival of the fittest."

But of course, "fittest" in technology does not always mean "best" - hello, Macintosh and Betamax. On the Net and in court, Sun is already battling competing Java "flavors" - variations of the language - launched from Redmond. In May, Sun filed a suit against Microsoft to try to rein in its licensees and enforce "100 Percent Pure Java." But the fight has at least given Jini's creators the benefit of hindsight. And the sidestep they came up with plays directly to the strength of a distributed system: When Jini tries to run on a nonstandard Java Virtual Machine, Jini automatically queries its capabilities, then uploads whatever chunks of code are needed to make it fully compatible. "You could design a system to prohibit that," says Clary, the Jini project manager. "But that would violate the licensing terms a lot more flagrantly than just leaving some features out. There'd be nothing gray-area about it. And it's hard to see the value in deliberately shutting yourself off from the world."

Sun has also been working overtime to address what remains the favorite bugaboo of Java skeptics: speed, as in lack thereof. Java's "sluggishness" is a favorite complaint of Net surfers watching Web applets - Java's most visible face - slowly unfurl. The seriousness of the problem has attracted a correspondingly high amount of programmer energy. As one result, a new generation of just-in-time compilers is emerging for a variety of operating platforms, produced both by Sun itself and by third-party developers. And later this year, Sun will release the 1.2 version of Java, one of whose new features, HotSpot, is dynamic optimization, which Sun officials claim can take JIT compilation to "C-level performance."

Will Jini scale out to the size of, say, the planet? "We've looked at this every way we can think of," says Clary. "And the answer is yes." Object-based programming makes sense for the same reason that packet-switching is now the technology of choice for networks: It reduces huge problems to small pieces. That in turn points inevitably to a move from today's mostly client/server networking to peer-to-peer relations, with code and data flying in all directions across the network. And the resulting complexity, the Jini team concluded, could be dealt with only by stripping its basic operating rules to an absolute minimum. "How do we know whether we made the right choices?" says Waldo. "You never know. We stopped only when we couldn't throw things out anymore."

When Joy and Clary took Jini to Sun CEO Scott McNealy for a green light in March of last year, they used the phrase "opportunity driven" - Valley-speak for a project that will build its market on the fly. As with Java, the benefits to Sun are a subject for debate - possibly, Jini-configured hardware; more certainly, an inside track on what could well be historic technological changes. What everyone agrees is that timing will determine Jini's fate. "It's like that portal opening in *Star Trek,"* Joy says. "If you're lucky, you get through the opening, and then the portal closes."

Comes the comet

In 1979, Steve Jobs - then an unknown 23-year-old geek - made his now-legendary visit to Xerox PARC to see the radical new Alto computer, with its primitive mouse and icon-based screen. "I saw a very rudimentary graphical interface," he said years later (see "Steve Jobs: The Next Insanely Great

Thing," *Wired* 4.02, page 102). "It wasn't complete. It wasn't right. But within 10 minutes it was obvious that every computer in the world would work this way someday." Two years ago, Jobs made the same prediction about object-based distributed computing. "You can argue about how many years it will take," he said, "and who the winners and losers will be during this transition. But you can't argue about the inevitability."

WebObjects, Jobs's project in pursuit of that vision, never took off in part because its success depended on a wholesale switch to a new hardware platform, the ill-fated NeXT. Not an especially good strategy for an undertaking with universal aspirations.

But, as Jobs predicted, one way or another it will happen - indeed, it is happening, before our very eyes. The Web is growing in every dimension - faster, bigger, deeper and more sophisticated by the day. Intelligence is being embedded in everything. Ever larger chunks of human activity are migrating to the network. And that greater genie surely will not be going back into any bottle.

Joy's Jini, if it takes hold, has the potential to overturn the familiar territory of hardware, personal computers, peripherals, phones, TVs, and appliances. The vision of what comes after is just that - a vision. For people like Bill Joy, it hovers like a city on a hill, elegant and platonic, waiting for us humans to make it so. But the closer it gets, the easier it will be for everyone to see. "Imagine a global network so complex it will be a kind of organism, a dynamic, richly interconnected medium wrapped around the earth 24,000 miles deep." That's not Teilhard de Chardin - it's the 1997 annual report from Daimler-Benz North America.

For now, though, some old lines are still drawn: central planning versus competition. NT's 20 million lines of code versus the 600 Kbytes of Jini. Bill versus Bill. Redmond versus Aspen - there's a pattern working here, and it almost surely has as much to do with philosophy or faith as it does with questions of mere technology.

For its part, Jini is gambling that a small nudge can actually relocate a mountain. "Our goal is to lose control over the network," says Jim Waldo, "and make everyone else - from Bell Labs to Redmond - lose control too." He's not talking about market share, not by itself anyway. "What we're trying to build are the mammals to compete with the big computational dinosaurs. You can imagine how the conversation went: 'They're too small. They're nothing - they're not enterprise scaled.' But the comet is coming. And when it does, we know who inherits the earth."

Jini is a set of new software layers that together create an overarching "federation" of computer devices and services.

On top is a **directory service**, based on a "lookup" mechanism that allows different Jini-enabled devices and applications to register and be seen on the network. The next-level service is persistence, provided by **JavaSpaces** technology, which stores objects so that other users or applications can retrieve them. Below that, a set of protocols based on Java's **Remote Method Invocation** enables objects to communicate and pass each other code. And finally a **boot, join, and discover protocol** allows Jini-compatible devices, users, and applications to announce themselves to the network and register in a directory.

Any device with an operating system capable of supporting a **Java Virtual Machine** - meaning, in practical terms, any modern computer - can be linked with a Jini network. Simpler devices can also join, though on a more limited basis.

JavaSpaces are **virtual "bulletin boards"** or "marketplaces" - the heart of Jini's distributed networking. Using a few simple programming methods, including "read," "write," and "take," JavaSpaces make software **objects** available to anyone in a network. The objects themselves can define a job to be done, a problem to be solved, or a service being offered. A JavaSpace can be as small as 10K and as large as 100 Mbytes.

Executive editor Kevin Kelly (kevin@wiredmag.com) is the author of New Rules for the New Economy (Viking/Penguin, 1998); Spencer Reiss (spencer@wired.com) is a senior editor at Wired