Parallel lines

David Fearon cites the search for ET as a prime example of the power of parallel processing.

If you've just got yourself a new PC system and you're wondering what to do with all the raw processing power that current CPUs offer you, then point your browser at www.setiathome.ssl.berkeley.edu and download the SETI@Home client.

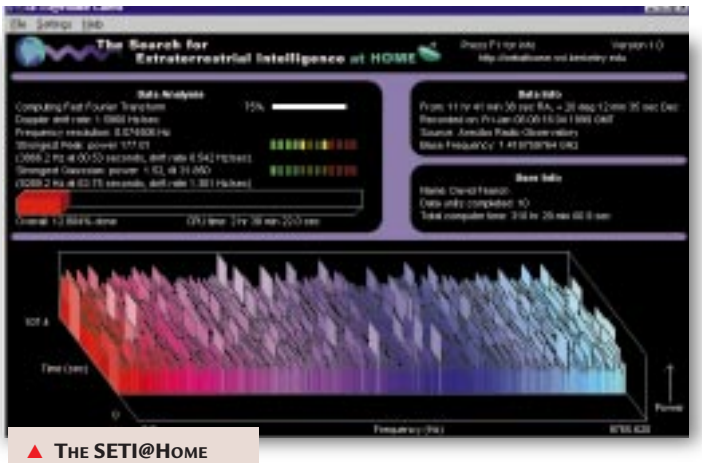
You've probably heard of SETI, the Search for Extra Terrestrial Intelligence, an organisation dedicated to hunting for ET. This is done by analysing data from radio telescopes pointed at the heavens, and looking for faint signals that could have emanated from civilisations beyond Earth.

But because of the vast quantities of data produced, even the fastest computers and dedicated hardware perform only relatively cursory examination of the data as it comes from the telescopes in real time. So the SETI@Home project has been set up to analyse the stored data generated by the Arecibo Radio Observatory, in Puerto Rico, in far greater depth than ever before.

THE STONE SOUPERCOMPUTER: PARALLEL PROCESSING ON A BUDGET

Hundreds and thousands

Thanks to that marvellous invention known as the internet, the hugely powerful computer required to perform this task is a virtual parallel device formed by hundreds of thousands of internet-connected client machines. All you need to do for your system to become part of the machine is download the client, which then downloads a chunk of data known as a work unit from the SETI@Home server and does some clever analysis. It sends the results back to the server and gets another chunk of data, and so on.



THE SETI@HOME CLIENT DOES SOME VERY IN-DEPTH SIGNAL ANALYSIS

The reason I'm talking about this in the *Hardware* column is because of the nature of the task: it's a superb example of the power of parallel processing. Each chunk of data sent to clients by the server is just 250kb in size and represents a 10kHz-wide slice of the 2.5MHz signal produced by the telescope, 107.4 seconds in length. Nevertheless, to process each chunk takes an average of 40 hours, depending on the system: even your brand spanking new Pentium III 550 is going to find itself working very hard. I've currently got one on loan, and it's taking up to 24 hours per work unit. Given that the telescope produces around 35Gb of data a day, you can see that it's a mammoth computing task.

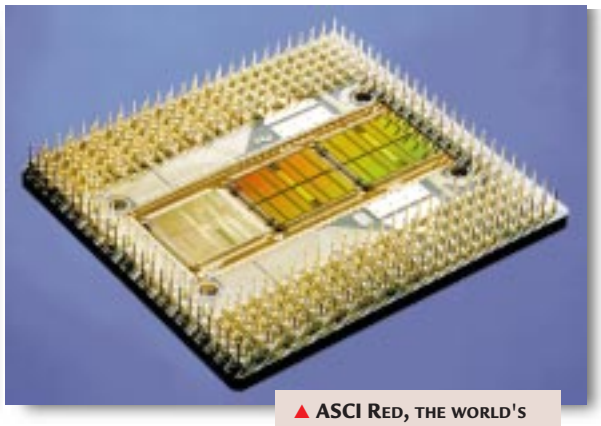
But because the data is being processed by a massively parallel computer with, at the time of writing, 500,000 processors (client machines), the data is being turned over at an incredible rate. The project began on 17th May, and after 18 days running, the accumulated CPU time dedicated to the task is already over 7000 years — and it's increasing at the rate of a millennium every few days.

Once you've registered with the server via the client software, the system keeps track of the number of data units sent to you, and the number of results posted back, along with the total CPU time spent by your systems. I say systems, because you can install the software on as many computers as you like; so if you're suitably equipped, you can partake of the joys of parallel processing yourself. Each data unit can only be worked on by one machine at a time, so the server sends a different unit to each machine and they're processed in parallel.

If you have two systems running the client, you can get through twice as many work units in a given time and increase your chances of being the person to discover intelligent life on another planet. The server keeps statistics, and maintains various league tables on who's processed the most units and how long each one is taking. But you'll need access to a whole lot of computers to stand any chance of getting to the top of the list.

Fast and loose

The type of parallel processing employed by SETI@Home is known as loosely-coupled. In other words, each client processor has its own, separate job to



▲ ASCI Red, the world's fastest computer, uses a whole heap of CPUs — and they're standard Intel Pentium Pros

do, and only communicates occasionally with the server, which dictates the organisation and hands out work units. Each processor has minimal interaction with the server, and no interaction at all with other processors (in the form of client machines).

This is the easiest form of parallel processing to set up, and it's extremely effective for computing jobs that require long-term processing of large amounts of data. 3D graphics and CGI (computer-generated imagery) for films are processed in this way. 3D animations lend themselves very easily to loosely-coupled parallelism, since rendering each frame of animation is a standalone task. Pixar Animation's *Toy Story*, for instance, was processed on a 'render farm' of 117 Sun SPARCstations.

Pixar's Toy Story was processed on a 'render farm' of 117 Sun SPARCstations

Loosely-coupled parallel processing is useful for predefined computing jobs with completion times of the order of days, weeks or longer, and has the advantage that increasing the number of machines (assuming all have equal spec) gives a directly proportional increase in speed. The disadvantage, obviously, is that using standalone machines takes up a lot of space, and not all computing tasks can be split into handy chunks for distribution via a client/server system.

For general-purpose parallel processing, you need to be able to split code at the machine instruction level, and you can't do that efficiently when the processors are connected by an internet link or even a fast network. So for less predictable tasks, the CPUs all need to reside in a local machine, with constant, tightly coupled co-operation and process integration controlled by the operating system.

The transputer trend

You may recall that ten or fifteen years ago, when the epitome of home computing power was a 32K BBC Micro, 'transputers' were hailed as the future of computing. Transputer was the trendy phrase for a parallel computer — in other words, one with two or more local processors. It seemed obvious: rather than spending vast amounts of money squeezing incremental speed improvements out of single-processor machines, simply wire up a bunch of processors, and hey presto! As much speed as you like. And it's true that using this method has led to some terrifically powerful machines: all of the world's fastest supercomputers are parallel computers.

The list of the top 500 fastest machines is maintained at www.top500.org, and king of the hill is currently Sandia National Laboratory's ASCI Red. This military monster currently has — wait for it — 9152 processors. And they're not some kind of exotic silicon either: they're standard Intel Pentium Pros, proving that massive parallelism is the most cost-effective way to the fastest machines.

Up to the limit

So, rather than shelling out to replace our old processor each time Intel comes up with something new, why aren't we all running parallel computers that we can upgrade just by bunging in an extra one? The answer, and the reason that the excitement of transputers is now just a memory to the normal user, is a limitation not of the hardware, but of us.

It's incredibly hard to write applications for parallel execution. With a standard single-processor computer, code is written the way it's executed, in a serial sequence. One thing happens after another, which is the way the human brain likes it.

But once you start breaking tasks up and executing them in parallel, things get very complex very quickly. For every step of the code, the programmer needs to take into account how to split the task, and what the consequences might be if one instruction is executed ahead of or behind another. And the more processors you have, the harder it gets, so massively parallel computing is still largely confined to research labs.

Specific operating systems are developed and tailored to the individual machine, and each application is written from the ground up with great care. But programmers writing mainstream applications to a marketing schedule don't have the time or the money for that. Having said that, those who aren't confined to schedules and applications for the mass-market have done some interesting things — check out www.esd.ornl.gov/facilities/beowulf for an interesting example.

Of course, there is already limited opportunity for mainstream use of multiple processors in a desktop system: Windows NT has native support for SMP (symmetrical multi-processing).

NT Workstation supports up to two processors, while Server will run with up to four. It's called symmetrical multiprocessing because you can only use an even number of identical processors — you can't mix and match clock speeds.

But as you may have gathered from the preamble, sticking another processor in your NT box won't magically make your computer twice as fast. Applications need to be specifically coded to be multithreaded — x86 machine code has no native support for parallelism — and because of the extra programming effort involved, few are.

Also, the overhead associated with thread synchronisation means that you won't get a straight 100 percent performance increase. Nonetheless, SMP can be an extremely useful and cost-effective way of getting a performance boost for a workstation.

Aside from the programming difficulties, SMP support is still relatively rare partly because most mainstream applications are run on Windows 95/98 machines, which are unable to use SMP. There's little point in expending the extra effort to write multithreaded apps when only a tiny percentage of the user base will see the advantage.

But that will change with the merging of the NT codebase into all incarnations of Windows when Windows 2000 is released. In fact, even games developers are starting to put the effort into investigating the possibilities of SMP: the production release of Quake 3 Arena will support it, which, according to John Carmack, Id Software's lead programmer, leads to a performance improvement of between 40 and 80 percent.

EPIC proportions

Further down the road, things are looking up for parallel processing. The near-mythical next-generation Intel processor, codenamed Merced, could make the prospect of adding processors to your system on an *ad hoc* basis a reality. Merced is now due in mid-2000 and will bring with it the new IA-64 architecture. IA-64 is, as the name suggests, a 64-bit architecture; in other words, data and addressing lines are 64 bits wide, as opposed to the 32-bit architecture of current Pentium-family CPUs.

More importantly, IA-64 is the first architecture to implement EPIC, or Explicitly Parallel Instruction Computing. Existing CISC and RISC processors already use various internal techniques to try to process more than one instruction at once where possible.

But the degree of parallelism in the code is only determined at run-time by parts of the processor that attempt to analyse and re-order instructions on the fly. This takes time and wastes die space that could be devoted to executing, rather than organising instructions.

With EPIC code, the compiler has already done a lot of work to tease out and organise instructions that can be executed in parallel, and instructions have specific flags to indicate whether they can be executed independently or whether they're part of a mutually dependent group. The compiler does the hard part, so the programmer doesn't need to specifically write parallel applications.

The result is that the processor can simply grab as large a chunk of instructions as possible and execute them simultaneously, with minimal pre-processing. This makes Merced inherently scalable, so more expensive versions of the chip could simply incorporate more instruction execution units to increase speed, and systems could potentially be designed with the ability to slot in new execution units whenever an upgrade is required, similar to plugging in more memory modules on existing systems.

FAT32 GETS THE BOOT ON 98/NT

After Roger Gann's advice on operating systems and filing-system compatibility in *PCW*'s July issue [*Hands On Workshop, p202*], Mark Lawrence emailed us to point out the existence of an immensely useful utility that gives Windows NT4 the ability to read and write FAT32 volumes. There are a couple of other utilities around, but this one is free if you only need to read FAT32 — ideal for accessing Windows 98 partitions on a dual-boot 98/NT machine. Surf on over to www.sysinternals.com and grab a copy. Bear in mind, however, that writing to FAT32 rather than NTFS volumes renders NT's file-system security useless, since all a nasty noser has to do is reboot the machine into Windows 98, or simply use a DOS boot floppy, to gain unrestricted access to the FAT32 partitions.

A couple of people have also pointed out a slight error in the file-system compatibility table. Linux does now feature kernel support for FAT32 under the Vfat file system, so anyone using Windows with FAT32 volumes can access them from Linux if required.

▶ **System Internals' FAT32 reader gives FAT32 access to NT**

WATCH THIS SPACE

Unfortunately, circumstances beyond my control have prevented me from bringing you the promised piece on remote-booting Windows 95 clients. I'll do my best to get it into a future issue, so keep an eye out if you're interested.

PCW CONTACTS

David Fearon welcomes your comments on the Hardware column. Contact him via the PCW editorial office, or email hardware@pcw.co.uk