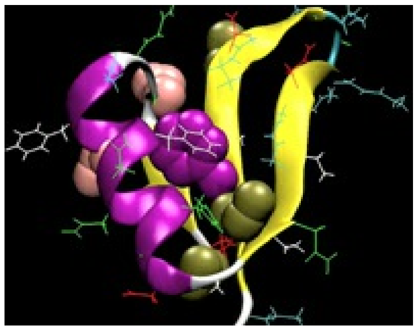
Speeding up Science

Speeding up Science
Many of today's most difficult scientific computations include some gain from GPUs

The Folding@home project often uses GPUs to compute simulations of the folding of proteins, such as the one displayed here.
Courtesy of Vijay Pande



Scientific simulations can tackle larger problems and produce more accurate answers, as computing grows increasingly parallel. This need for parallel processing proves particularly crucial for some questions, including how to make better drugs and how elements formed after the Big Bang. Such simulations may require millions of iterations, with results feeding back into more calculations. In the past, the parallelism came from using more CPUs — either from a cluster or from a multicore chip — but graphics processing units (GPUs) now offer an alternative. As a result, today's scientific calculations run faster than ever.

In short, GPUs offer much more parallelism than a multicore CPU. "A GPU has tens or even hundreds of cores," says Dinesh Manocha, professor of computer science at the University of North Carolina at Chapel Hill. "If a specific algorithm can make good use of GPU-based parallelism, then perhaps you can achieve higher performance on a GPU."

Supercomputers also can benefit from GPUs. On the June 2010 Top10 list of supercomputers, the second fastest machine — the Nebulae at the National Supercomputing Center in Shenzhen, China — consists of a combination of multicore CPUs and an NVIDIA Tesla C2050 GPU.

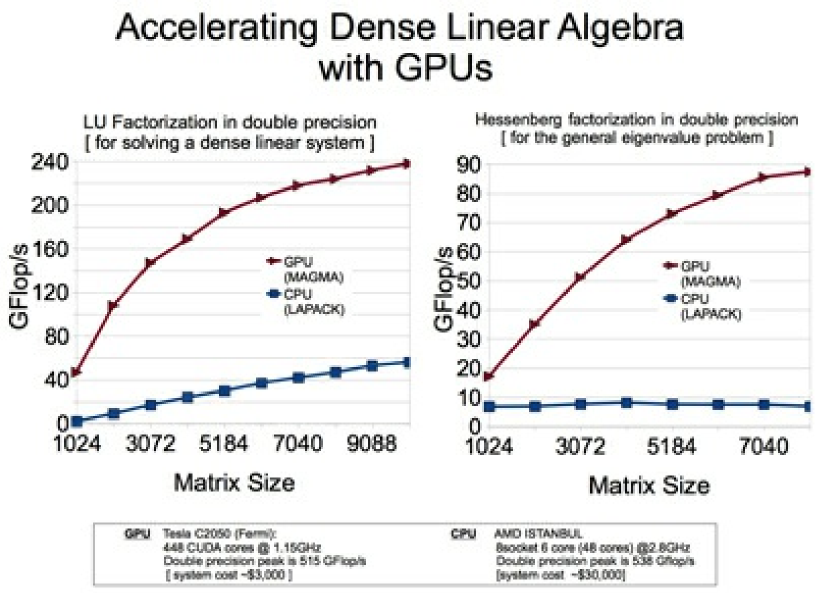
Such hybrids will probably take over in the future. "If we want bigger or faster machines, this is what the future looks like: undoubtedly multicore and almost undoubtedly heterogeneous multicore," says Bronson Messer, acting director of science at the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory.

The question is: How can scientific algorithms take advantage of this evolving hardware?

Processor Pros and Cons
GPUs emerged as graphics accelerators, especially for rasterization algorithms. That capability made these processors very hot in the gaming world. Today, GPUs appear on video cards and motherboards, so systems can combine a CPU and GPU for numeric computation. In such a case, some aspects of an algorithm can run on the CPU while the highly parallelizable parts get ported to a GPU.

For now, NVIDIA is the leading GPU vendor, and AMD makes a close second. Still, the best-known chipmakers, including Intel, have focused on CPUs, although often designing many-core processors. Moreover, AMD and Intel have both announced that they will come out with hybrid — CPU/GPU — chips in the next year.

In running dense linear algebra, GPUs can run some algorithms much faster than CPUs can.
Courtesy of Stan Tomov



Nonetheless, it’s not always clear what makes the best choice for a specific algorithm. As explained by Vijay Pande, associate professor of chemistry at Stanford University: “In terms of the science output, more-modern multicore CPUs and GPUs both can do much more powerful tasks than a single-core CPU, but it’s not easy to say one is always better than the other.” He continues, “To make a car analogy: GPUs are like a drag racer, very fast at one task, but not good as a general utility vehicle; multicore CPUs are much more general and still very powerful.”

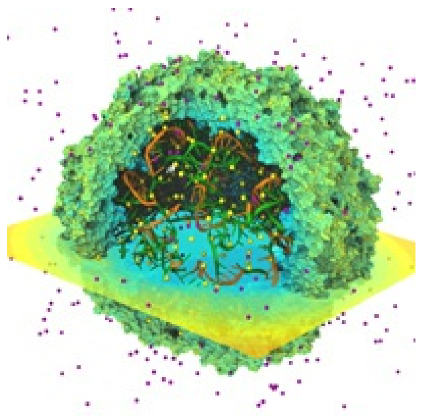
Pande truly understands the need for computing power. As director of the Folding@home distributed computing project, he is always looking for faster ways to study the folding and misfolding of proteins, which is crucial to their function. “There are limitations to what we can learn from traditional experimental research, and simulations allow one to make detailed models and predict how biological systems would behave,” Pande says.

As a rule of thumb for picking or passing on a GPU, John Stone, a senior research programmer in the theoretical and computational biophysics group and NIH Resource for Macromolecular Modeling and Bioinformatics at the University of Illinois at Urbana-Champaign, says, “If an algorithm is largely sequential and branchy, it might not make a good match for a GPU.”

Riding the Waves Faster
Many areas of research — including acoustics, electromagnetics and fluid dynamics — require scientists to solve the wave equation, which is a second-order linear partial differential equation. For example, petroleum companies use this equation when searching for oil. Recently, Manocha’s research team solved the acoustic wave equation entirely on GPUs for sound propagation. “This generated a speed improvement of a factor of almost 100 over prior methods,” Manocha says.

In addition, scientific computation often relies on other basic capabilities, including linear algebra. Stan Tomov, a computer scientist at the University of Tennessee, works in this area. “Our software allows the user to specify their

Here the satellite tobacco mosaic virus was colored by the electrostatic field created by the virus capsid and contained RNA. The small purple and yellow atoms surrounding the virus structure were added to neutralize this electrostatic field in the completed model in preparation for a molecular dynamics simulation. The electrostatic field was calculated using a linear-time multilevel summation method electrostatics algorithm running on an NVIDIA Tesla C1060 GPU, at a speed just over 26 times faster than a single CPU core, or 6.6 times faster than a typical 4-core CPU. The GPU-accelerated multilevel summation method algorithm is implemented in the freely available molecular visualization and analysis tool VMD (www.ks.uiuc.edu/Research/vmd).
Courtesy of Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign



hardware context — like five cores and two GPUs, say — and the software is designed to use the maximum capability of those hardware resources," Tomov says.

When Tomov started porting linear algebra algorithms to GPUs, he admits that he was skeptical. "Previous research was showing that older GPUs were not that good for some algebra," he says. For instance, some applications need so-called dense linear algebra. For it to run faster than the limits imposed by the speed of the main GPU memory, these algorithms need intermediate memories, or cache, that is closer to the processor and, thereby, faster to access. Today, some GPUs include a memory cache and, in general, have well-developed memory hierarchy. As a result, Tomov says that GPUs can run many algebra algorithms 10 times faster than even a high-end multicore system.

Where CPUs didn't do a very good job with a scientific algorithm, the speed up from going to a GPU could be even better. For example, Stone and his colleagues recently ported a histogram function — one that shows the distances between atoms in a molecular simulation — and the code ran 92 times faster. In talking about this work, Stone says, "When the speed up is greater than about 30, you're in a situation where the algorithm hit a weak spot on a CPU or the implementation lacked something."

Simulating the Stars

The key to making GPUs central to scientific computing is moving from one-offs to general solutions. A key issue is keeping the data near the GPU. "With a heterogeneous architecture, you need to cast your data structures to maximize data locality and exploit as much parallelism as possible," says Messer at Oak Ridge. "You need to minimize data movement as much as possible, keeping the data close to where they will be processed."

Moreover, Messer says, "We want a code that a scientist or developer can work with and, whether running on a machine with GPUs or not and one core or many, it's the same code base, and it runs the same everywhere." He adds, "That's the aim of lots of compiler vendors and tool vendors."

Among other things, Messer and his colleagues generate code that simulates the synthesis of elements in supernovae, which are exploding stars. "It takes every sort of physics you can think of and all at once," Messer says. "To get a precise element synthesis, we knew we'd need to ramp up the computational intensity." In one component of those calculations, for example, Messer expected a 700 percent increase in runtime when enhancing the physical fidelity of the element production but, with CPU-GPU hybrid code, he expects to run the more realistic simulation in the same amount of time that the lower fidelity one took on CPUs alone.

Other scientists will face similar battles when turning gaming hardware into research tools. The future battles, though, should be simplified as programming tools hide the hardware in the background.

Mike May is a freelance writer for science and technology based in Houston, TX. He may be reached at editor@ScientificComputing.com