

# Genome Technology

## Parallel Code for Dummies

July/August 2007  
By *Matthew Dublin*

---

There is always a steady flow of high-performance computing vendors offering specialized hardware and software products that promise not to burden the researcher with IT headaches and just leave them to the science. But there's a surprising dearth of easy solutions that enable scientists to seamlessly parallelize their customized applications or algorithms for use on a cluster or grid.

Even with sophisticated workbench tools like MATLAB or PYTHON, researchers have traditionally been chained to their desktops. Such a predicament has obvious negative consequences in terms of limited processing power and available memory for crunching the ever-growing datasets researchers face. But programming customized applications to run in a parallel environment is not exactly a common skill among bioinformaticists or computer scientists, let alone the average biologist.

It is precisely this limitation that has forced many software developers and scientists to devise solutions for transforming such algorithms and applications into parallel implementations. And while effective, these solutions are hardly a walk in the park for those without a near-expert command of high-level programming languages such as C, C++, FORTRAN, and MPI (message passing interface), a protocol used to condition an algorithm to run smoothly and efficiently on a parallel, multiprocessor system.

If you want to parallelize your algorithm, say a long farewell, because you won't see it for a while. Investigators can sometimes wait up to a year before their specially parallelized algorithm or application comes back from an expert programmer, only to find bugs or other quirks that render the code useless. And that's not even including the long and labor-intensive task of testing the new parallelized algorithm on sample datasets to ensure its accuracy.

### Managing the Pain

As scientists like James Evans of the Computational and Systems Biology Initiative (CSBi) at MIT know, parallelizing your own code is no picnic. "In the past, we've had big supercomputers that require you to parallelize your code using MPI, and there was a lot of resistance by grad students and postdocs because it's a significant amount of work," Evans says. "We're developing prototypes, so you don't want to spend a week or month debugging or working out how the parallelization is going and then find out that the piece of code was no good anyway so it was a waste of time." Evans works with huge arrays of data such as 50 GB spreadsheets with hundreds of columns and millions of lines, so the idea of getting anything accomplished on a desktop was a difficult and often pointless task.

But earlier this year, Evans adopted a software solution by the technical computing outfit Interactive Supercomputing. Star-P is an open software platform that allows users to parallelize applications developed in popular desktop environments such as MATLAB, PYTHON, or R, and harness the compute power of a cluster without having to master the high-level languages required to do so. "Basically, Star-P just works in the background to parallelize MATLAB routines that we typically use, and you pretty much get linear speed up from running on your desktop," says Evans. "Instead of taking 10 minutes to wait for something to come back, it will take one minute, and we can basically evaluate whether a particular method we're using is going to pan out quickly, so it's just better."

One application for which Evans has found Star-P particularly useful is in the visualization and analysis of cells under various drug treatments. Researchers at CSBi used a function in MATLAB called K-means, a computationally intensive algorithm that clusters objects in a dataset. Using K-means, various drug effects can be grouped together according to drug treatment conditions and cell attributes. But a desktop PC buckled under the weight of the huge datasets required to conduct such an analysis. After implementing Interactive's product, Evans is now able to prototype parallelized K-means models right in MATLAB and then run those jobs on multiple processors, thereby utilizing the power of a cluster for the same jobs that were previously forced to slog their way through on a single desktop processor.

## The Pitch

Ilya Mirman, Interactive Supercomputing's vice president of marketing, says that many researchers who use desktop tools like MATLAB and PYTHON to customize algorithms to run on large datasets quickly run into problems. And when that happens, the only answer is high-performance computing. "Why do people use high-performance computing? They turn to it when they run out of steam on the desktop," says Mirman. "The typical workflow is this: the scientists develop the original algorithm on the desktop, runs into a brick wall from a performance standpoint, then he stops and a team of programmers spend on average 12 to 18 months to parallelize that algorithm." Mirman says that because of the daunting and time-consuming task of developing hand-coded parallelized algorithms, the majority of new Star-P users are actually using high-performance computing for the first time.

Researchers at the National Cancer Institute's Pediatric Oncology Branch are also using Interactive's platform to speed up a customized application called CORR4DB, used to correlate one genomic array against a database of 100,000 probes. CORR4DB was developed on a PC in MATLAB, but due to ever-increasing microarray sample sizes reaching the tens of thousands, the desktop's processors and available memory were reaching their limit. Instead of turning to a parallel programming expert to transform their application into a parallel program, CORR4DB was parallelized using Interactive's solution in MATLAB.

"You don't have a lot of PhD scientists who have parallel programming skills and understand high-performance computing," says Jack Collins, manager of the scientific computation and program development group at the Advanced Biomedical Computing Center, NCI's supercomputing resource. "We have other [computing] resources, but for the scientists, MATLAB was a comfortable environment where they had developed their code, they knew what it did and to rewrite that, there's a maintenance issue." Collins says that with Star-P, NCI researchers running jobs that were previously taking up to 50 hours on a desktop were reduced to half an hour.

Mirman acknowledges that there are some advantages to parallel coding an application line-by-line, but says those benefits are negligible when compared to the lengthy prototyping period. In most cases, Star-P is able to tweak an application to run at roughly 80 percent to 90 percent of the performance level of hand-coded parallelized application. "We're not going to magically make C++ and MPI code on the server run faster than if you were to hand-tune it yourself, but what we can do is be a productivity breakthrough," Mirman says. "So if an algorithm takes a year and a half to program and test it, we're going to get you to run it in a week."

© Copyright 2007 GenomeWeb Daily News. All rights Reserved.