

MATLAB Embraces HPC

By Silvina Grad-Freilich

New development tools quicken the migration of high-level technical applications into high-performance computing environments.

As requirements for technical computing applications become more complex and development time shrinks, engineers and scientists must solve problems of increasing computational intensity with many involving huge data sets. These domain experts often find that the key challenge is not only the inherent difficulty of the problem; it is also the fact that the computational intensity of the problem exceeds the capabilities of their computers. One solution is for these domain experts to use high-performance computing environments.

In the past, high-performance computing was available mainly to government agencies and big research labs because only these groups had the means to purchase supercomputers. Today, most supercomputers have been replaced by commercial-off-the-shelf (COTS) computer clusters that provide affordable high-performance distributed environments. For example, a Cray supercomputer available 15 years ago cost about \$40 million and provided a performance of about 10 gigaflops (10 billion instructions per second). In 1998, the same performance from a Sun HPC server cost about \$1 million. Today, a four-processor Dell configuration can provide the same level of performance for only \$4,000.

PROBLEMS WITH TRADITIONAL WORKFLOWS

While domain experts are attracted by the availability and low cost of this technology, the complexity of developing software applications for distributed environments creates a substantial practical barrier. Most domain experts work in high-level languages, such as MATLAB or Mathematica. Until recently, there were no commercially available high-level tools for prototyping and developing technical computing applications for distributed environments from vendors of these high-level computing languages and elsewhere. As a result, a technical computing application developed in a high-level language would have to be rewritten in C or Fortran for distributed computing, an arduous and time-consuming task for which many engineers and scientists do not possess the required skills. While C or Fortran programmers might be more proficient at translating the software to a lower-level language and adapting it to distributed hardware, this approach presents its own difficulties, as few programmers are domain experts.

There are other disadvantages: Converting the application from a high- to a low-level language results in two versions of the application. It is difficult and inefficient to go back to the original high-level language version if requirements change, but if code changes are implemented in the low-level program, the advantages of working in the intuitive high-level environment are lost.

Moreover, to achieve maximum performance, programmers

usually tune the application written in the low-level language to the specific hardware on which it will run. This means that if the hardware changes, as is frequently the case, the program must be retooled. Any processing time gained through using distributed computing is thereby offset by the time spent retooling the program to run in a distributed environment.

Problems like these could be avoided if high-level languages used by domain experts directly supported a distributed computing environment. Domain experts could then prototype and develop distributed applications from within the environment used to develop conventional technical computing applications, with considerable reductions in time and cost. The application could also be redeployed to any hardware platform supported by the high-level language, not to mention less frustration and less error-prone results. While a program written and implemented in a high-level language might perform less efficiently than a program written by an expert C or Fortran programmer, the reduction in time-to-solution far outweighs the minor loss of computing speed.

DISTRIBUTED COMPUTING TOOLKITS

Several third-party toolkits for high-level languages are already available for developing distributed and parallel applications. A survey conducted by the Supercomputing Technologies Group at Massachusetts Institute of Technology identified almost 30 toolkits for use with MATLAB. Most of these toolkits were developed for research projects and are in the public domain. Few are commercially supported, which means that no technical support or help is available. Most are no longer being developed, and, therefore, do not work with the latest versions of MATLAB and application-specific MATLAB toolboxes. While the actual toolkits are often free, most require full MATLAB and associated toolbox licenses for each computer in the distributed computing environment, raising the cost of ownership.

The MathWorks has responded by providing products that enable domain experts to prototype and develop distributed com-

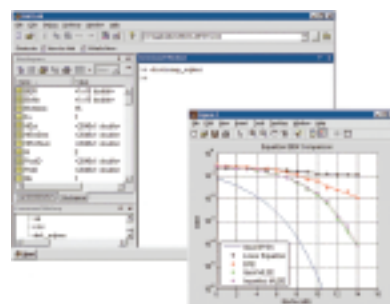


Figure 1: Plot displaying the performance of five equalizers. Using the distributed computing products, this MATLAB application was divided into several tasks, each of which performs the same calculation for different signal-to-noise ratios, and is executed on a remote cluster.



Figure 2: Processing 30 GB of geospatial information across the US with The MathWorks distributed computing products running on a cluster of four 2.8 GHz Pentium IV machines, achieves an almost linear speedup.

puting applications in MATLAB. Version 1 of these products supported coarse-grained, or embarrassingly parallel applications in which the same algorithm runs independently on several nodes, without communication, shared data, or synchronization points between these nodes. Parameter sweeps and Monte Carlo simulations are examples of applications that may fall within this category. Version 2 addresses finer-grained applications containing interdependent tasks that require communication between tasks during processing. It implements key Message Passing Interface (MPI) function calls, the industry-wide standard protocol for point-to-point communication in a parallel program.

In version 1, users could prototype and develop distributed application interactively and access distributed computing resources at development time. Version 2 provides support for pluggable schedulers, such as LSF from Platform Computing, enabling users to also send applications to a batch queue and receive resources for execution only when the scheduler allocates them. Pluggable schedulers provide support for integrating the distributed computing tools to users existing distributed computing environments.

SIMPLIFYING THE PARALLELIZATION TASK

Using The MathWorks distributed computing tools, domain experts can develop applications in MATLAB and then divide them into tasks that are evaluated remotely on cluster nodes. In the simplest case, where the problem can be divided into tasks consisting of the same function with the same number of input and output variables, a single function call parallelizes the problem for a distributed computing environment. In other words, one line of code is sufficient. In more complex cases, a few functions, or several lines of code, are required.

Because these tools can execute algorithms that include any toolbox for which the user is licensed, there is no need to purchase additional toolboxes licenses for the cluster nodes. As a result, the cost of ownership is a fraction of what it would be with third-party toolkits, which would require purchasing licenses of each toolbox for every node where it might be used.

Just as COTS clusters have reduced the cost of distributed computing, so will programs written by domain experts in high-

level languages reduce the time and effort required for application development. This shift is being driven by users who expect not only the performance of high performance computing applications but also their programmability and portability. High-productivity programming tools can dramatically reduce the time to solution by making it easy for domain experts to develop their own distributed computing applications. The more prevalent these tools become, the more companies will invest in hardware, software applications, and the training required to support the development of distributed computing applications.

Silvina Grad-Freilich is the Product Manager for distributed computing and application deployment products at The MathWorks. Silvina holds both B.Sc and M.Sc. degrees in Computer Science from the National University of La Plata in Argentina and a M.Sc. in Management from the Massachusetts Institute of Technology.

COMPANIES MENTIONED

Cray, Inc.
Seattle, WA

Dell, Inc.
Round Rock, TX

Platform Computing, Inc.
Markham, ONT

Sun Microsystems, Inc.
Santa Clara, CA

The Mathworks, Inc.
Natick, MA

Supercomputing Technologies Group
Massachusetts Institute of Technology
Cambridge, MA

Desktop Engineering Magazine
PO Box 525, Dublin, NH 03444

Telephone (603) 563-1631 Fax (603) 563-8192

Copyright© 2006 Desktop Engineering Magazine. All Rights Reserved.



91407v00 08/06