THE Star-P HIGH PERFORMANCE COMPUTING PLATFORM

Alan Edelman
Dept of Mathematics, Massachusetts Institute of Technology
Edelman@Mit.edu
Interactive Supercomputing

ABSTRACT

The high performance MATLAB® user now has more choices than ever. Interactive Supercomputing's Star-P embraces this new world where, as an example, a MATLAB user who never wants to leave MATLAB might sit next to a C++ programmer at the office and both surf the internet for the latest high speed FFT written in yet another language. The MATLAB of the past now becomes one browser into a bigger computational world. HPC users need this bigger world. Other "browsers" can be imagined. The open Star-P platform gives users options never before available to programmers who have traditionally enjoyed living exclusively inside a MATLAB environment.

***Index Terms*—**MATLAB, parallel, high performance. prototyping

1. INTRODUCTION

Interactive Supercomputing's STAR-P platform represents a fresh approach to high performance computing whereby the MATLAB user has all the comforts and features of the familiar desktop environment, while gaining access to the emerging opportunities in hardware and open source software that represent the new reality of the modern generation of high performance computing.

The platform embraces open standards as illustrated in the proposed future architecture diagram of Figure 1. The current Star-P product only supports the MATLAB client but this is just the beginning.

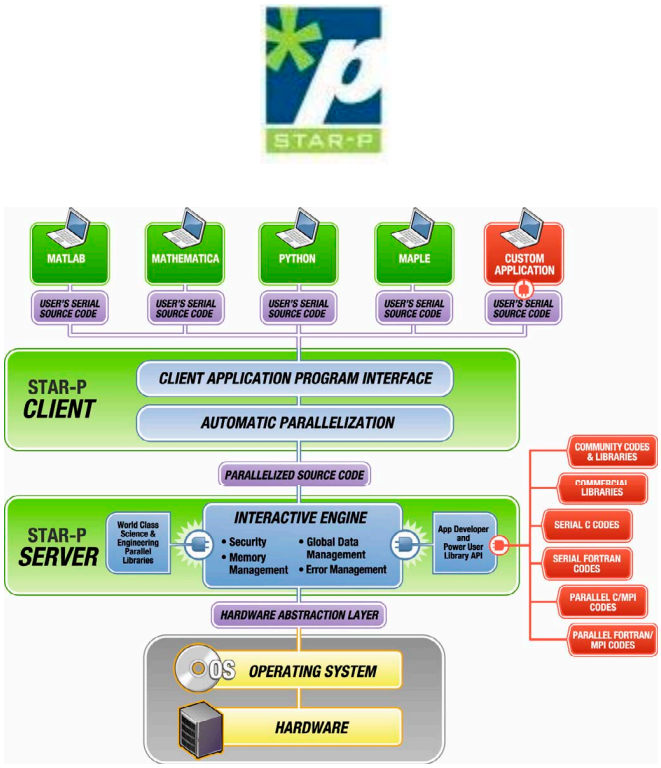

Figure 1: Star-P architecture with proposed future clients

To set the stage it is useful to put MATLAB in its correct context. In a recent survey, users of high performance computers were asked which application they used for prototyping codes. The results are histogrammed in Figure 1 below. The survey indicates clearly that by and large most users are prototyping in the C family of languages. Following this are slightly more MATLAB users than Fortran users. The open source language Python is emerging as a player in this space. Then Mathematica, Excel, Octave, R, and IDL all figure together as significant players.

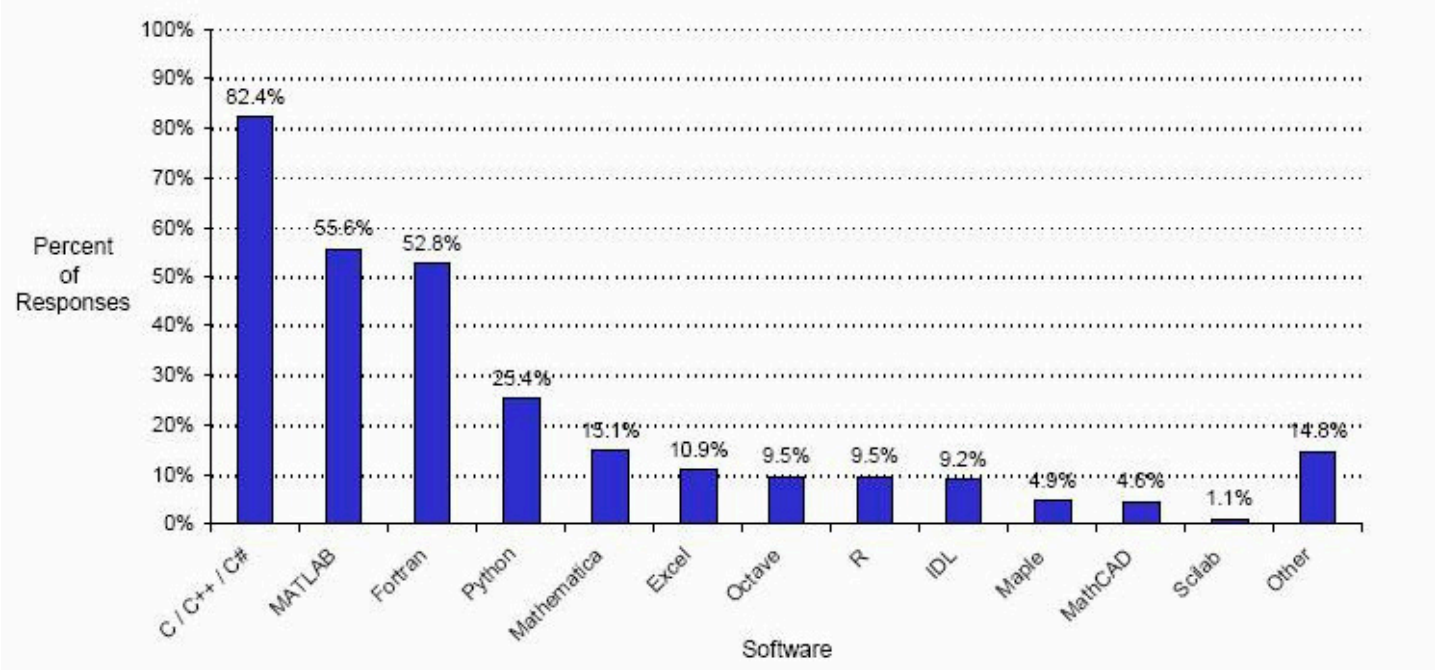

Figure 1: Software users begin with to prototype high performance computing applications? Users may employ multiple prototype languages giving a sum exceeding 100%. (Source: Simon Management Group study)

The significance of this survey is that MATLAB alone does not contain all the pieces of the puzzle needed by high performance computing users. This is a significant contrast from the desktop world, where by and large many users are happy to stay within the confines of the MATLAB environment and functionality. Furthermore, an organization purchasing an HPC product for MATLAB alone may not be addressing the full needs of its users. Soon these open source languages are likely to outstrip the features and performance of MATLAB alone. Other proprietary libraries continue to innovate providing users with sets of choices not readily available within the walls of the MATLAB environment. The goal of Star-P is to make these possibilities seamlessly available to MATLAB and non-MATLAB user alike.

2. NEW SOFTWARE STANDARDS

The Interactive Supercomputing Star-P platform addresses the difficulties faced by many high performance computing users with existing codes.

Scenario 1: Suppose a MATLAB user has inherited codes in C or Fortran MPI. This user believes it would be impossible to rewrite this code in MATLAB, but nonetheless really wishes the clock can be turned back and if it could all be started over again, the user can run in MATLAB. The Star-P environment has the hooks and abstractions to allow MPI programs to be seamlessly plugged into the platform.

Scenario 2: Suppose a user who generally prefers the MATLAB environment has some serial C or Fortran codes that needs to be run with a multitude of parameter choices and then later the large collection of data needs to be analyzed inside the MATLAB environment. The Star-P environments makes this possibility particularly easy.

Scenario 3: Suppose a user wants a new HPC tool from a propriety library vendor. These users all in all would prefer transparent connections to their products when they overlap MATLAB's functionality but realize superior performance.

The Star-P environment allows the development of wrappers for this purpose.

Scenario 4: With the emerging growth of multicore software with multithreaded products, users of clusters of SMPs who wish to run the multicore software seamlessly will be able to do so in the Star-P environment.

3. STAR-P HISTORY AND PERFORMANCE STUDY

Star-P began ten years ago as an academic research project at the Massachusetts Institute of Technology. Based perhaps loosely on lessons learned at Thinking Machines Corporation, and with inspiration from the MultiMatlab Project at Cornell University, [9] we built what may well have been the first global array based parallel MATLAB in the mid-nineties. A few years ago, a survey of all the parallel MATLABs was undertaken to understand the use of compilers, compiler/server technology, master/slave technology, and message passing approaches. [1]

Every year the Star-P technology was tested in MIT's graduate course on high performance computing. The first author is proud that among his students have been both authors of FFTW, some of the authors of pMATLAB,[7,8]. Over the years the project was known as MITMATLAB, pMATLAB itself, MATLABp, and MATLAB*p.

In a recent classroom experiment reported in [11] and reprinted here for the benefit of the ICASSP audience, Students participated in performance studies as part of the development time study experiment of the HPEC program [6]. What has become increasingly clear from these studies is that a few very talented students who have the knack, can find ways to improve the performance of codes, but even the most talented and inclined still expend a great deal of time.

The students were given a by now standard programming assignment in parallel computing classes, the two dimensional Buffon needle problem. A typical parallel MATLAB solution in Star-P looked like:

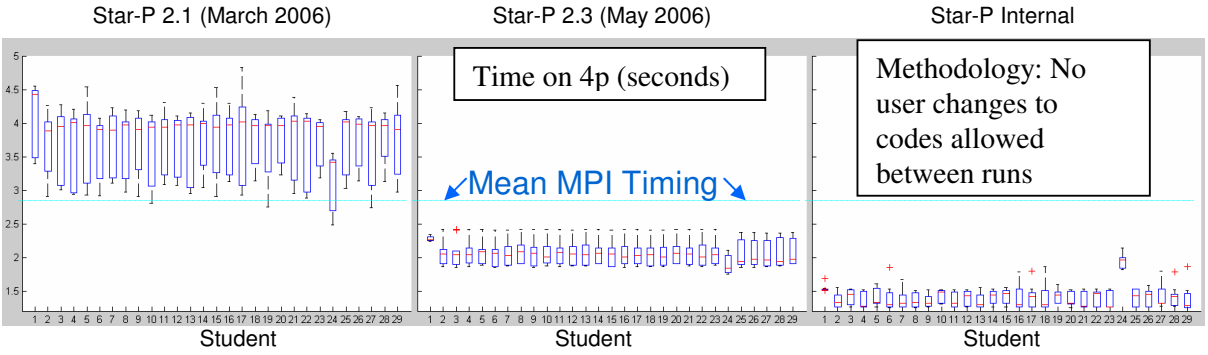

Fig 2: The Buffon Needle Problem executed by 29 students in three evolutionary versions of Star-P each executed ten times and compared with MPI runs written by the same students. The mean MPI timing was 2.8 seconds. We have not here normalized per student but we should report that a handful of students who worked hard achieved what might be considered the optimum of 1 sec on 4 processors in MPI. In a boxplot, the blue box ranges from the 25th to 75th percentiles of the ten data points. The red line is at the median. The whisker is the full extent of the data omitting outliers which are the red plusses. Writing message passing code was widely considered an unpleasant chore while the insertion of the two characters "*p" hardly seemed to be worthy of an MIT problem set.

```
function z=Buffon(a,b,l, trials)
r=rand(trials*p,3);
x=a*r(:,1)+l*cos(2*pi*r(:,3));
y=b*r(:,2)+l*sin(2*pi*r(:,3));
inside = (x >= 0) & (y>=0) & (x <= a) & (y <= b);
buffonpi=(2*l*(a+b) - l^2)/ (a*b*(1-sum(inside)/trials));
```

The serial MATLAB code differs from the parallel one by the "*p" in red above. We ran each code ten times in three revisions of STAR-P. Figure 2 plots the students timings on 4 processors (ten million trials).

We can only report anecdotal evidence about the human time for all 29 students, but overwhelmingly the students preferred adding the two characters "*p" to their code as compared to writing the MPI code. The mean time was 2.8 seconds on four processors. A handful of the students who were determined to performance tune their MPI code reached times close to 1 second. Thus the Star-P system brings users to within 40% of the hand coded optimum. The Star-P design allows for even this overhead to be shaved down further in future releases.

To understand scalability, the following times are the mean run times on the internal version of Star-P. (We note that the other versions of Star-P indicate similar scalability characteristics:) Each number is the average of 290 runs, 10 runs for each of 29 student codes.

Processors	1	2	4	8
Avg Seconds	5.7	2.9	1.4	0.7

Our view of this experiment is best illustrated as in the cartoon in Figure 3 which follows the productivity methodology introduced by Kepner and colleagues.

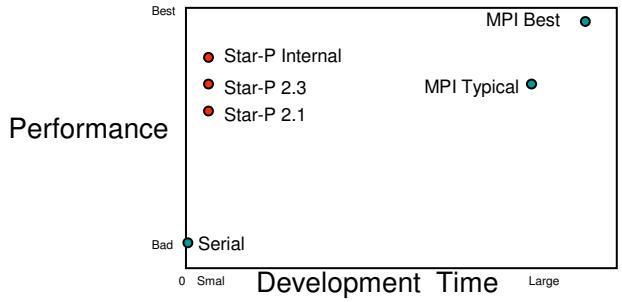

Fig 3: Kepner diagram illustrating the main point of this study. Productivity may be thought of as best slope on line to the origin. The vertical rise in performance of Star-P may be thought of as riding the technology curve as students expended no additional effort. Typical methodologies only report MPI vs serial on the vertical axis. The Kepner methodology provides the means of seeing productivity on a two dimensional scatter plot.

.

4. REFERENCES

.

[1] R. Choy and A. Edelman, "Parallel MATLAB doing it right," Proceedings of the IEEE, Vol.93, No.2, Feb 2005, pages 331-341.

[2] P. Husbands and C. Isbell, "The Parallel Problems Server: A Client-Server Model for Large Scale Scientific Computation." Proceedings of the Third International Conference on Vector and Parallel Processing. Portugal, 1998.

[3] P. Husbands, Interactive Supercomputing, PhD Thesis, Massachusetts Institute of Technology, Cambridge, 1999.

[4]Interactive Supercomputing: http://www.interactivesupercomputing.com.

[5] A Edelman, MIT Course 18.337: http://beowulf.csail.mit.edu.

[6] J. Kepner, http://www.highproductivity.org

[7] J. Kepner and S. Ahalt, "MatlabMPI," Journal of Parallel and Distributed Computing (JPDC), 64(8): 997-1005 (2004). (http://www.ll.mit.ede/MatlabMPI).

[8] N. Travinin and J. Kepner, pMatlab Parallel Matlab Library IJHPCA 2006. (http://www.ll.mit.edu/pMatlab).

[9] Vijay Menon and Anne E. Trefethen, MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing

[10] Sudarshan Raghunathan: Making a aupercomputer do what you want: High level tools for parallel programming (Computing in Science and Engineering)

[11] A. Edelman, P. Husbands, S. Leibman, Interactive Supercomputing's Star-P Platform:
Parallel MATLAB and MPI Homework
Classroom Study on High Level Language Productivity, Proceedings of HPEC06.