pMatlab Takes the HPCchallenge

Ryan Haney, Hahn Kim, Andrew Funk, Jeremy Kepner, Charles Rader, Albert Reuther and ***Nadya Travinin***

MIT Lincoln Laboratory, Lexington, MA 02420

Phone: 781-981-2514

Email Addresses: {haney, hgk, afunk, kepner, rader, reuther, nt}@ll.mit.edu

Abstract[1]

The HPCchallenge benchmark suite has been released by the DARPA HPCS program to help define the performance boundaries of future Petascale computing systems. The suite is composed of several well known computational kernels (STREAM, Top500, FFT, and RandomAccess) that span high and low spatial and temporal locality. These kernels also encompass key aspects of embedded signal processing: vector computations, matrix multiplies, corner turns and random selection operations. MATLAB®[2] is the primary high level language used within the signal processing community and is increasingly used for large system simulations and quickly processing data in the field. The pMatlab parallel MATLAB toolbox provides the necessary global array semantics to allow HPCchallenge to be implemented. The results provide a unique opportunity to probe both the relative (pMatlab vs. MATLAB) and absolute (pMatlab vs. C/Fortran+MPI) merits of pMatlab. Specifically, for each kernel in HPCchallenge we examine code size, maximum problem size, and performance. We find pMatlab code to be approximately 10x smaller than the equivalent C/MPI code. The problem sizes possible using pMatlab scale linearly with the number of processors (e.g. we are able to FFT a 2^{28} complex vector on 16 CPUS), and are comparable to the corresponding C/Fortran+MPI code. Finally, the scalability of the kernels approaches that of the C/Fortran+MPI code.

Introduction

The HPCchallenge

The DARPA High Productivity Computing Systems (HPCS) program has initiated a fundamental reassessment of how we define and measure performance, programmability, portability, robustness and, ultimately, productivity in the HPC domain [1]. With this in mind, HPCchallenge is designed to approximately bound computations of high and low spatial and temporal locality for Petascale systems. Figure 1 illustrates the approximate spatial/temporal relationship of the different kernels and the connections to important operations in the embedded

signal processing community. In addition, because HPCchallenge consists of simple mathematical operations, this provides a unique opportunity to look at language and parallel programming model issues. This paper compares traditional C/Fortran+MPI with MATLAB using global array semantics.
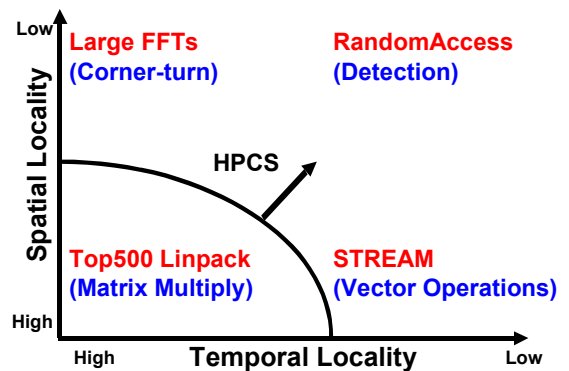

Figure 1: HPCchallenge kernels are plotted relative to spatial and temporal locality.

The pMatlab Parallel Toolbox

The pMatlab toolbox implements global array semantics in MATLAB. pMatlab provides high-level parallel data structures and functions without removing the fast prototyping capability and ease of use for which MATLAB is well known [2]. This is achieved by combining operator and function overloading with the concept of parallel data and task mapping to provide implicit data and computational parallelism. pMatlab is currently being used for simulating signal processing chains and for rapid analysis of sensor data in the field. The implementation of the HPCchallenge using pMatlab offers a means for more detailed performance analysis of pMatlab.

Parallel Implementation

STREAM consists of four local operations performed on distributed vectors: copy, scaling, addition, and scaling with addition. All of these operations are important in signal and image processing. The STREAM benchmark requires no interprocessor communication and is implemented using simple distributed matrices.

RandomAccess is designed to measure the random access capabilities of a computer system. This is accomplished by effectively computing the histogram of a random number generator, replacing the typical addition

---

[2] MATLAB is a registered trademark of The Mathworks, Inc.

update with a bit level XOR operation. The ability to randomly access data and perform logical operations are standard "post detection" signal processing operations. RandomAccess requires dynamic communications among all the processors and is implemented using parallel sparse arrays.

The Top500 Linpack Benchmark uses an LU Solver to solve a dense linear system of equations such as $Ax=b$. Such an algorithm requires selecting and communicating arbitrary parallel sub-matrices typical of many dense linear algebra operations. At the core of LU are matrix-matrix multiplies typical of multi-element beamforming operations.

The FFT kernel performs a 1-D Fast Fourier Transform. The 1-D FFT is performed by computing two 2-D FFTs, and then corner-turning the distributed matrix in between the two computations. Both the local 2D FFTs and large matrix corner turns are among the most important operations in multi-sensor signal processing.

Results
For each kernel in the HPCchallenge, we examine code size, maximum problem size, and performance on a Linux cluster consisting of dual 3.0 GHz Xeon processors connected with Gigabit Ethernet. Examining code size, we find pMatlab code to be approximately 10x smaller than the equivalent C/F77+MPI code. Approximate software lines of code numbers for the HPCchallenge kernels are shown in Table 1.

The maximum problem sizes possible using pMatlab scale linearly with the number of processors used and are comparable to the corresponding C/F77+MPI code. Figure 2 illustrates this for the Top500 kernel. The maximum input matrix size run on 16 processors (28K x 28K) is 16x the maximum size that can be run on a single processor (7K x 7K). Figure 3 shows the performance and maximum problem size achieved in the pMatlab FFT code relative to serial MATLAB, which uses FFTW [4] to implement its Fourier Transform. The performance scalability is typical of that seen in C/F77+MPI implementation.

Lines of code	C/F77 + MPI	pMatlab	C/F77+MPI / pMatlab
STREAM	441	51	~8
Random Access	225	101	~2
FFT	~1100	72	~15
Top500	~5000	200	~25

Table 1: C/Fortran + MPI vs. pMatlab software lines of code for four of the HPCchallenge benchmarks.
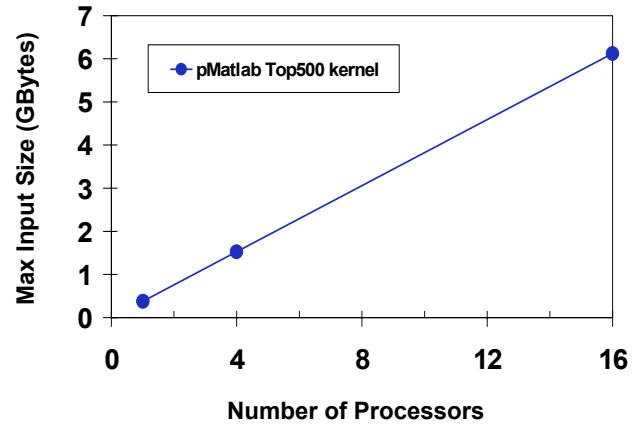

Figure 2: Maximum input matrix data sizes are plotted for the Top500 kernel. Each matrix contained real double-precision data.
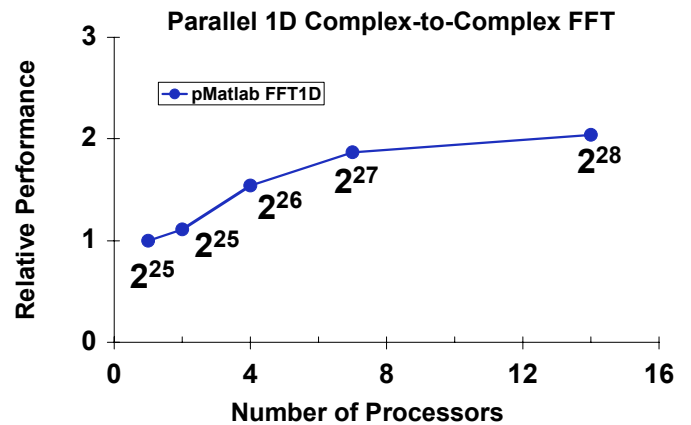

Figure 3: Performance (Flops) and scalability results are plotted for the FFT kernel. Results are relative to the serial MATLAB performance. Numbers next to the points indicate the size of the complex vector used.

References
[1] HPCS - High Productivity Computer Systems. http://www.highproductivity.org, 2004.
[2] Jeremy Kepner and Nadya Travinin. "Parallel MATLAB: The Next Generation". HPEC 2003 Workshop, 2003.
[3] Jack Dongarra. "Performance of Various Computers Using Standard Linear Equations Software". University of Tennessee, Knoxville TN. http://www.netlib.org/benchmark/performance.ps, 2004.
[4] FFTW Fastest Fourier Transform in the West. http://www.fftw.org, 2004.