Very High Level Languages (VHLL) for No Pain Scalable Computing on High Performance Systems

Bracy H. Elton, Siddharth Samsi, Harrison Ben Smith, Laura Humphrey, Stanley Ahalt, Alan Chalker
Ohio Supercomputer Center, {elton, samsi, bsmith, humphrey, ahalt, alanc}@osc.edu
Niraj Srivastava, Roope Astala, Interactive Supercomputing Inc.
{nsrivastava, rastala}@interactivesupercomputing.com

Introduction

While Moore's Law continues to drive the density of integrated circuits, heat and geometry issues are making it impractical to create faster and faster processors. Instead, more and higher performance systems are being built using multi-core processors or accelerators like Graphics Processor Units (GPUs), Field Programmable Gate Arrays (FPGAs), often packaged into high-performance clusters that amplify computing power while avoiding single-processor constraints. As domain experts require even higher capability and capacity to keep up with data growth and the latest applications, they have little choice but to employ these parallel computing platforms. However, most current methodologies employ explicit parallel techniques such as MPI or PGAS paradigms that require a big initial investment in application development before a domain expert is able to test scalability of the application. In contrast, "productivity" is an "attribute of the entire process of computing that delivers ultimate value to the end user mission" [1]. Studies have shown large gains in productivity through the use of high level languages [2,3]. The purpose of this paper is to provide a study in scalability of a VHLL, namely Star-P®, on the MJM system at the Army Research Lab Department of Defense Supercomputing Resource Center (ARL DSRC). The results will focus on the use of Star-P software platform allowing transparent use of DSRC high performance computing resources from familiar desktop environment while providing scalable performance.

Star-P

Star-P is an interactive parallel computing platform from Interactive Supercomputing, Inc. (ISC). It leverages existing desktop simulation tools for simple, user-friendly parallel computing to a spectrum of computing architectures: SMP servers, multi-core servers, distributed memory clusters systems and cluster of GPUs. With Star-P, DoD users can continue to use the development and simulation tools they are already familiar with to solve larger, more complex problems that cannot be done on a desktop computer. Existing MATLAB® or Python scripts can be reused to run larger problems faster by decomposing problems or tasks to run in parallel with minimal modification. Star-P implements distributed memory parallelization of arrays and a client/server model that allows desktop users to take advantage of HPC resources.

Acknowledgement: This publication was made possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract No. GS04T01BFC0060. The opinions expressed herein are those of the author(s) and do not necessarily reflect the views of the DoD, Mississippi State University, the Ohio Supercomputer Center, or Interactive Supercomputing, Inc.

MJM Overview

MJM has 1100 two-processor socket dual-core compute nodes, each with 8 GB of memory, and has 8 two-processor socket dual-core login nodes, each with 16 GB of memory. The system utilizes a high-speed 4X DDR Infiniband interconnect and a global shared file system. Processors operate at a frequency of 3.0 GHz, resulting in a peak floating-point rate of 12.0 Gflop/s per core. In total, the 4400 compute cores deliver 52.8 Tflop/s.

Using Star-P on the ARL DSRC MJM System

Star-P is intended for interactive, parallel computing from MATLAB or Python.

Interactive: The session can start from user's local Windows/Linux client or from login nodes of MJM. Interactive mode assumes that the cluster is ready and available for computations or that resources are made available via a workload manager (WLM), such as Platform Computing's LSF, Sun Grid Engine (SGE) or Torque, among others. The Star-P client initiates Star-P server process on the cluster. The Star-P server, in turn, submits a request to the WLM for compute resources for parallel work. Once the WLM allocates the resources, the Star-P client manages execution of parallel work on these resources.

Batch: When using shared computing resources such as those available at the DSRCs and managed by WLM, the queue for resources can be long, especially for large allocations. The interactive mode of operation is thus not practical, and the established way to run simulations is to use the batch system for both the Star-P client and server. In a batch system, users are able to use standard LSF scripts to request the desired resources first and then run Star-P with a given M-file or a Python script on allocated resources.

Reservation: Users are also able to set up advanced reservations to run their analyses. This is particularly useful for users intending to run analyses interactively, say, on data as it is being collected in a lab or in the field, without having to wait for compute resources.

Scalability Results

We present three results on the MJM system. The first comprises results of HPL benchmark written in MATLAB script with Star-P modification for data parallel computation.

```
idx= 200000
x = rand(idx,idx*p);
```

```
ppchangedist(x,3);
y = rand(idx*p,1);
tic; z=x\y; toc;
```

This simple MATLAB script solves a (random) dense linear system in double precision (64-bit) arithmetic of a 200K by 200K random matrix by distributing the matrix across the cluster, as per the following:

- Adding the `*p` construct makes variables parallel.
- Related variables, via propagation, become parallel.
- Functions on parallel variables are transparently "overloaded" and themselves become parallel.

Figure 1 shows the scalability of HPL as function of data size and number of cores. Simple code above is able to consistently get around 50% of the peak. We did notice some variability in the runs, which seems to be related to performance variation between the nodes. For the 512 core run, we observed 10% performance variation in the memory subsystem (see Table 1).

Table 1: STREAMS result from a 512-core benchmark showing minimum and maximum values. (N=Node#)

Function	Rate(MB/s)	Min @(N 29)	Max @(N 160)
Scale	4589.2	4185.7	4627.1

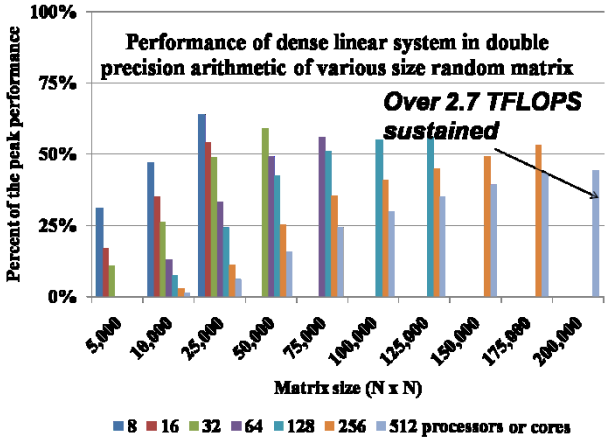

Figure 1: Scalability of parallel operations on distributed data.

The next example shows the scalability of task parallel algorithms using a simple Gaussian filter that is applied to many instances of a 1025x1025 matrix. Figure 2 shows scaling as a function of cores and number of tasks. The result shows near-perfect scaling with tasks and cores.

The final example is radio frequency (RF) tomographic imaging in the Air Force Research Laboratory Radar Signal Processing Technology Branch [4]. The VHHL and scalability results provide technical direction toward a shortened development cycle time for a RF tomography imaging experiments. The shortened development cycle results in increased productivity, reducing the time to develop a deployable system.

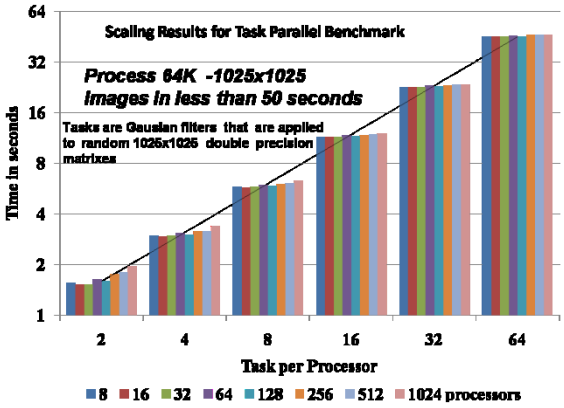

Figure 2: Time to execute tasks as a function of the number of tasks on various numbers of cores.

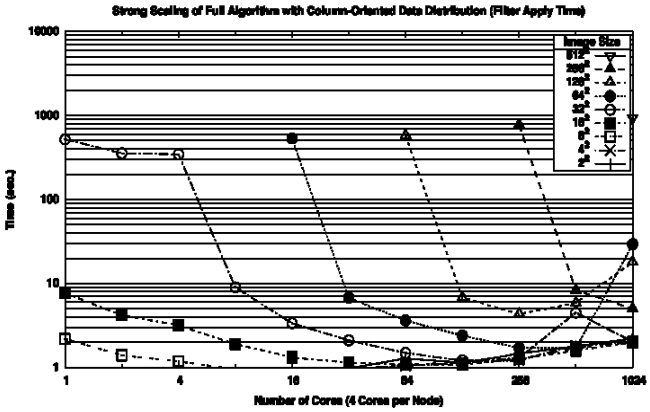

Figure 3: Application scalability for fixed problem size

Summary

As the results show, VHLL can hide the complexities of parallel programming, while providing high performance to the user. Furthermore, users are also shielded from the complexities of interacting with various high performance systems and their environments by providing a consistent client environment.

References

[1] Thomas Sterling, "Productivity Metrics and Models for High Performance Computing," *International Journal of High Performance Computing Applications*, Vol. 18, No. 4, pp. 433-440, 2004.

[2] J. Kepner and N. Travinin, "Parallel Matlab: The Next Generation," *Proc. 7th High Performance Embedded Computing Workshop* (HPEC 2003), MIT Lincoln Lab., 2003.

[3] A. Edelman et al., "Interactive Supercomputing's Star-P Platform: Parallel MATLAB and MPI Homework Classroom Study on High Level Language Productivity," *Proc. 10th High Performance Embedded Computing Workshop* (HPEC 2006), MIT Lincoln Lab., 2006.

[4] M. C. Wicks, B. Himed, J. L. E. Bracken, H. Bascom, and J. Clancey, "Ultra narrow band adaptive tomographic radar," *1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, IEEE, Dec. 2005, pp. 36–39.