Developing a Computational Science IDE for HPC Systems

David E. Hudak, Neil Ludban, Vijay Gadepally, Ashok Krishnamurthy
Ohio Supercomputer Center
{dhudak, nludban, vijayg, ashok}@osc.edu

Abstract

Software engineering studies have shown that programmer productivity is improved through the use of computational science integrated development environments (or CSIDE, pronounced "sea side") such as MATLAB. ParaM is a CSIDE distribution which provides parallel execution of MATLAB scripts for HPC systems. ParaM runs on a range of processor architectures (e.g., x86, x64, Itanium, PowerPC) and its MPI binding, known as bcMPI, supports a number of interconnect architectures (e.g., Myrinet and Infinband). In this paper, we describe our goals for the ParaM project, the current status of the project and report on initial software engineering successes and challenges.

1. Introduction

The adoption of traditional computer science tools (i.e., C and FORTRAN programming with compilers like gcc, UNIX debuggers like gdb and profilers like gprof) has been problematic for computational scientists. Mastering the complexity of these tools is often seen as a diversion of energy that could be applied to the study the given scientific domain. Many computational scientists instead prefer computational science integrated development environments (or CSIDE, pronounced "sea side"), such as MATLAB or Mathematica.

The popularity of CSIDEs in desktop environments has lead to the creation of many scientific solutions (coded as CSIDE commands commonly called "scripts") and, more importantly, the training of many computational scientists. It would be a great boon for computational science if those solutions and that expertise could be applied to our largest computational systems, also known as high performance computing (HPC) systems.

At the Ohio Supercomputer Center (OSC), we have developed a CSIDE distribution for HPC systems called ParaM. ParaM has been installed on a number of HPC systems at OSC and other HPC centers. In this paper, we recap the advantages of CSIDEs for computational science, describe the design of ParaM and analyze the development of ParaM as a software engineering project.

2. Computational Science IDE's (CSIDE)

A computational science integrated development environment (abbreviated as CSIDE and pronounced "sea side") is defined as suite of software tools, including (1) a numeric interpreter with high-level mathematics and matrix operations, (2) support for domain specific extensions (e.g., signal and image processing, control systems, operations research), (3) available graphics and visualization capabilities, and (4) a common user interface (typically including an editor) for interacting with the various tools. Commercial examples of CSIDEs include MATLAB, Maple and Mathematica. Notable open-source examples include GNU Octave (extended with OctaveForge and GNUPlot), Scilab and Python (extended with SciPy, NumPy, iPython and Matplotlib).

CSIDEs are used as alternatives to traditional programming languages such as FORTRAN and C. Advantages of using a CSIDE have been enumerated for educational purposes [8] [7]. These advantages include the interactive nature of the interpreter and the tight integration between the various tools in the CSIDE. Specifically, interactivity provides intuitive debugging and incremental development support as well as run-time inspection of complex data that may need sophisticated analysis (such as visualization) for validation. Tight integration between CSIDE tools removes the burden on the computational scientist to learn a number of different software packages and their (potentially complex) interactions.

3. Extending CSIDEs to HPC Systems

As CSIDEs have spread in popularity, they have been applied to data sets of increasing scale. In order to handle the necessary data processing requirements, users have written CSIDE scripts designed to work in parallel on a single data set. In order to support such parallel scripts, parallel interpreters have been developed. These parallel interpreters are typically

29th International Conference on Software Engineering Workshops(ICSEW'07)
0-7695-2830-9/07 $20.00 © 2007 IEEE

composed of (1) a sequential interpreter from a CSIDE (2) a job control mechanism for launching multiple copies of the interpreter on multiple processors and (3) communication libraries for interpreters to exchange results.

Software packages that provide parallel interpretation are informally referred to here as HPC CSIDEs. While not providing comprehensive CSIDE functionality on an HPC system, these parallel interpreters allow users to port scripts developed within a CSIDE to parallel execution environments, such as HPC systems.

3.1. Examples of HPC CSIDEs

For the remainder of this paper, we will restrict our discussions of HPC CSIDEs to ones executing scripts similar to MATLAB. There are a large number of both commercial and open source packages available to provide parallel interpretation for MATLAB scripts. For example, twenty-seven separate parallel interpreter systems based on MATLAB have been catalogued [3] in addition to our own project, ParaM. Commercial projects include Interactive Supercomputing Star-P and the Mathworks' Distributed Computing Toolbox (DCT) and Engine (DCE).

MatlabMPI [1] and pMatlab [5] are popular open source packages for parallel interpretation. MatlabMPI supports a "message passing" programming model for MATLAB scripts. pMatlab supports a partitioned global address space (PGAS) programming model and relies on MatlabMPI. Both libraries are composed entirely of MATLAB scripts and can execute anywhere MATLAB does. However, the message passing constructs are implemented via file reads and writes to a globally shared file system, which incurs a high latency for messages.

3.2. HPCS findings supporting CSIDEs

Software engineering studies from the DARPA HPCS program (http://www.highproductivity.org/) point to the need for a CSIDE that provides access to HPC resources. Carver et. al. [2] describe a set of observations about HPC software development teams, including: (1) Development of science and code portability are of primary concern, (2) High turnover in development teams and (3) Iterative, agile development. Traditional CSIDEs provide interactive access, run-time data inspection and high-level source code (scripts) written at a higher level and (typically) easier for new developers to understand.

Wolter et. al. [10] found that time to solution is a limiting factor for HPC software development. Use of a CSIDE often reduces time to first solution when compared to traditional FORTRAN or C programming. Wolter et. al. [10] also found that steep learning curves associated with parallel tools inhibit their use by computation scientists. This is reminiscent of arguments supporting CSIDEs in sequential environments [7], namely, that computational scientists prefer an integrated environment with previously designed, well-understood interactions between tools.

Funk et. al. [4] describe the relative development time productivity metric (RDTP). HPC Challenge [6] is a series of benchmarks designed to exercise various aspects of an HPC system. The RDTP metric is applied to various implementations HPC benchmarks, and a CSIDE (MATLAB extended with pMatlab) implementation scored higher than C+MPI for three of the four benchmarks. These data indicate that a CSIDE can improve parallel programmer productivity.

4. ParaM

4.1. Overview and status

At OSC, a number of research scientists have been using MatlabMPI for parallel interpretation of MATLAB scripts. These researchers had a number of additional requests, including support for processor architectures not supported by MATLAB (e.g., Itanium), support for high-speed interconnects (e.g., Myrinet and Infiniband) and simplified installation.

In order to meet these requests, a development team at OSC developed ParaM. ParaM is a software distribution (a collection of internally and externally developed software packages) for parallel interpretation of explicitly parallel MATLAB interpreter scripts. ParaM can use existing MATLAB interpreter installations or is capable of using GNU Octave. It does not include full CSIDE functionality (for example graphing or visualization support is not included with Octave), but could be extended to provide such functions.

4.2. ParaMake package installer

ParaM is distributed as source code and is simultaneously developed on a range of UNIX platforms (Linux, NetBSD, Mac OS X). However, differences in the existence, version and/or installation location of software as well as varying linking conventions greatly complicate installation from source. In order to address these complications, certain conventions were adopted. First, a standard installation procedure was defined (a top level "ParaM" installation directory in which standard UNIX

distribution directories such as “bin”, “etc”, “include”, “lib” and “man” were placed). Second, GNU automake, autconf and libtool were adopted for all library construction. Finally, if the software needed to meet these conventions was not available on the system, it should be installed as part of ParaM.

In order to automate the process of locating, downloading, patching source (when needed), configuring, building and installing these packages, an installation utility called ParaMake was written. The goal was for end users (or system administrators) to customize ParaMake configuration files to complete a given installation. These configurations specify which toolchain, MPI library, interpreter and job control system to use. ParaMake installs UNIX tools (automake, autoconf, libtool), message passing libraries (OpenMPI), interpreter and support libraries (GNU Octave and fftw), communication support (bcMPI for message passing model, pMatlab for PGAS model), regression tests, examples and sample job control scripts.

ParaMake is only a package installer, not a full package manager like RPM (http://www.rpm.org) which is capable of identifying versions of installed packages and updating them if newer versions exist. We elected to write ParaMake because we wanted an installer that could be made to run quickly on many platforms (ParaMake is written entirely in python) and had simple configuration files that were intuitive to customize.

4.3. bcMPI

bcMPI is a software library that implements a bridge design pattern between an interpreter and an MPI library (both the interpreter and choice of MPI library are interchangeable). Our goals included support for (1) either MATLAB or Octave interpreters, (2) different MPI implementations and (3) support existing API’s for message passing, e.g., MatlabMPI [5] API and MATLAB’s Distributed Computing Toolkit (DCT) message passing API.

bcMPI’s software organization is presented in Figure 1. At its core, bcMPI is a C library with a collection of data types (matrices and structs) that support MATLAB and Octave data types. It also contains functions to serialize those data types to MPI data types and to exchange their values via MPI. The core library is supplemented by separate “bindings” to MATLAB and Octave. A binding consists of a set of external-interface functions (MEX functions for MATLAB and OCT functions for Octave) and a toolbox (i.e., a set of interpreter scripts), which is directly called by the user.

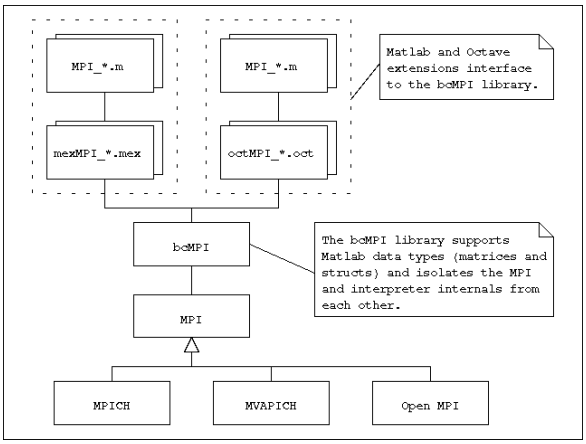

Figure 1. bcMPI software architecture

bcMPI is designed to supplement, not replace, MatlabMPI. A developer can use MatlabMPI on a multicore PC or departmental cluster to interactively develop and debug their parallel interpreter scripts. Since the same API is provided by bcMPI, the user can run scripts on a large shared cluster with a high-performance interconnect and a batch environment. In this way, the investment in code development time is preserved.

4.4. pMATLAB/bcMPI

As bcMPI supports the MatlabMPI API, it is possible to run MatlabMPI applications using bcMPI. In order to improve the performance of PGAS programming in pMatlab, the pMatlab library was incorporated into ParaM as a ParaMake package. Informally, this arrangement is known as “pMatlab over bcMPI” and abbreviated as pMatlab/bcMPI. Note that pMatlab/bcMPI is provided in addition to bcMPI, so users can choose between (or even combine) message passing and PGAS programming models. Also, the entire pMatlab API is preserved, so that users may develop their scripts on PC’s with the standalone pMatlab distribution, and then run them on an HPC system using pMatlab/bcMPI.

The RandomAccess benchmark is one of the HPC Challenge benchmarks and designed to test remote memory latency for fine-grained accesses. RandomAccess reports its results in Giga-Updates per Second (aka GUPS). Bliss and Kepner [1] reported a GUPS rate forty-six (46) times higher for C+MPI relative to pMatlab. The majority of this performance discrepancy is likely due to the latency of the filesystem used for message passing.

In order to compare pMatlab/bcMPI, C+MPI and pMatlab implementations of RandomAccess were run

on an AMD Opteron cluster at Ohio Supercomputer Center. The size of the RandomAccess table was 2^29 elements (4GB/node) and runs were conducted with 4, 8 and 16 processors using a gigabit Ethernet interconnect. The raw GUP rates for 16 processors are included in Table 1. The relative performance of C+MPI vs. pMatlab/bcMPI is presented in Table 2. Note that pMatlab/bcMPI performs close to (or better than) the GUP rate for C+MPI. Also, the GUP rate reported by pMatlab in included in Table 1 for completeness, but is not indicative of the maximum performance possible with standalone pMatlab, as the file system parameters on the test machine were not optimized for pMatlab.

Table 1. RandomAccess results

Language	Total GUPs
C+MPI	1.56E-03
pMatlab/bcMPI	1.86E-03
pMatlab	5.67E-07

Table 2. Relative GUPS performance

Processors	C+MPI/pMatlab+bcMPI
4	1.51
8	1.14
16	0.84

5. Software engineering observations

ParaM is intended to support a wide variety of platforms and multiple parallel APIs. The bcMPI software architecture has allowed us to rapidly deploy installations of bcMPI on a range of processors (AMD, Power, Itanium), MPI libraries (MPICH, MPICH2, OpenMPI) and interconnects (gigabit ethernet, Myrinet, Infiniband) out of a single source tree. In addition, support was added for the MatlabMPI API by extending the bindings with additional calls (no changes to the core library were required). Support for additional message passing and/or PGAS APIs (such as those provided by MATLAB's Distributed Computing Toolbox) can be added without changes to the bcMPI infrastructure.

ParaM is intended to support installation by users without UNIX system programming experience. ParaMake has made it possible for users to install ParaM automatically and without the need for root access on the target HPC system. In this regard, it can be considered a success. However, ParaMake has fallen short of providing an installation utility simple enough for non-expert users to customize for their systems. Such users are able to run an existing ParaMake installation, but lack the UNIX system experience to diagnose errors that occur during the build and installation process. Rather than spend (potentially large amounts of) development time trying to make ParaMake more sophisticated, we found it to be more effective to get an expert for a few hours on each new machine. The important part is that the entire process can be captured in a single configuration file for any user to repeat the installation.

Lastly, ParaM was designed with pragmatic software engineering practices such as source code management via version control (i.e., subversion), installation of regression tests and a consistent source code style. In the initial dissemination of ParaM to computational scientists, a widespread lack of knowledge of such tools and practices was observed. Note that this issue has been previously acknowledged and solutions have been proposed [9].

6. Conclusions and future work

ParaM is a software distribution developed at OSC that provides a computational scientist with the ability to run explicitly parallel MATLAB scripts on multiple processors. ParaM's message passing library, bcMPI, has a modular design that allows support for multiple interpreters (MATLAB and GNU Octave) across a range of HPC systems. In order to support computational scientists who may not be experts in UNIX system administration details, a package installer named ParaMake is included with the distribution. However, ParaMake, while automating much of the traditional UNIX configuration and build process, still requires extensive system knowledge to configure.

There are multiple avenues for future work. ParaMake can be improved. The ability to enable advanced MPI features, such as one-sided operations, would improve the performance for pMatlab scripts. Finally, HPC CSIDEs provide an integrated, high-level interface for data parallel mathematical operations. But, in order for computational scientists to develop parallel scripts that scale to large numbers of processors, user-friendly mechanisms for performance feedback must also be provided.

7. Acknowledgements

Thanks to Jeremy Kepner and the staff at MIT Lincoln Labs for providing a pMATLAB implementation of the HPC Challenge benchmarks.

8. References

[1] N. Bliss and J. Kepner, "pMatlab Parallel Matlab Library", *International Journal of High Performance Computing Applications: Special Issue on High Level Programming Languages and Models*, J. Kepner and H. Zima (editors), Winter 2006 (November)

[2] J. C. Carver, L. M. Hochstein, R. P. Kendall, T. Nakamura, M. V. Zelkowitz, V. R. Basili, D. E. Post, "Observations about Software Development for High End Computing", *CTWatch Quarterly*, Volume 2, Number 4A, November 2006.

[3] A. Edelman, "Parallel MATLAB Survey", http://www.interactivesupercomputing.com/reference/ParallelMatlabsurvey.htm

[4] A. Funk, V. Basili, L. Hochstein, J. Kepner, "Analysis of Parallel Software Development using the Relative Development Time Productivity Metric", *CTWatch Quarterly*, Vol. 2, No. 4A, November 2006.

[5] J. Kepner, and S. Ahalt., "MatlabMPI", *Journal of Parallel and Distributed Computing*, 64(8):997–1005, Aug 2004.

[6] P. Luszczek, J. Dongarra, J. Kepner, "Design and implementation of the HPC Challenge Benchmark Suite", *CTWatch Quarterly*, Vol. 2, No. 4A, November 2006.

[7] P. Webb, "Response to Wilson: Teach Science and Software Engineering with Matlab", *IEEE Computational Science and Engineering*, vol. 04, no. 2, Apr-Jun, 1997, pp. 4-5.

[8] G. Wilson, "What Should Computer Scientists Teach to Physical Scientists and Engineers?", *IEEE Computational Science and Engineering*, vol. 3, no. 2, June 1996, pp. 46-55.

[9] G. Wilson, "Where's the Real Bottleneck in Scientific Computing?", *American Scie*ntist, Jan.-Feb. 2006.

[10] N. Wolter, M. O. McCracken, A. Snavely, L. Hochstein, T. Nakamura, V. Basili, "What's Working in HPC: Investigating HPC User Behavior and Productivity", *CTWatch Quarterly*, Volume 2, Number 4A, November 2006.