AN EXTENSION OF PETSc FOR RECONFIGURABLE COMPUTER SYSTEMS: BLAS ON RASC

João Cleber Libório, Manoel Eusébio de Lima

Center of Informatics of Federal University of Pernambuco – UFPE – Brazil.

ABSTRACT

This work discusses the methodology to develop a library BLAS based, compatible with the PETSc library and working with the new reconfigurable logic platforms. High performance platforms have been used in many applications that requires massive data processing like genomics, proteomics, geophysics, mathematics, finance and so on. The text emphasizes is the BLAS and PETSc libraries and how they can be used in a software/hardware codesign to develop a reconfigurable library, using the SGI RASC System.

1. INTRODUCTION

Because the increase of scientific computational applications' complexity, the search for high performance processing of massive amounts of data has become a part of modern computer engineering. In latest years, the increase in functional density and performance of digital circuits has allowed the development of new techniques, capable of creating electronic products with greater speed, small size and low power consumption. Researches in areas like DNA structure, genomics, proteomics, simulations in geological and geophysical problems and calculations with huge matrixes in linear algebraic systems are examples with demand for this increase in computational speed.

The use of ASIC (Application Specific Integrated Circuit) technology in processing is discouraged due to the low level project difficulties and inflexibility, despite to the best performance and lowest energy consumption. The general propose processors are the most flexible platform and the easiest for an application to been developed, but their high clocking rates increase the power consumption. Also, their non-specific architecture allied with their computational paradigm decreases the performance of massive data processing. Between the two previous approaches, we have the FPGAs (Field *Programmable Gate Arrays)*. These devices provide a flexible architecture with low frequency, large data bandwidth and tools that allow their high level configuration.

2. SCIENTIFIC COMPUTATION

Some classes of scientific computation involves large dedicated libraries in several languages like *C, C++, FORTRAN*, etc. In general, all these libraries are implemented in software, in big machines with a big capacity of processing. The capacity and use of these machines however, based on general proposed *CPUs* or *ASICs*, start to be a problem due their limit in power, speed, flexibility and cost. Thus, this work discusses a hybrid architecture based on logic devices and general propose *CPUs*. In this particular case, the logics devices, the *FPGAs*, are used to accelerate the *PETSc* [1] functions in hardware, like an arithmetic co-processor.

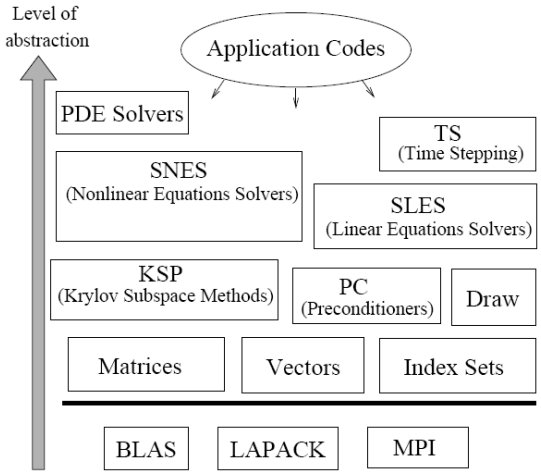

Figure 1 - The PETSc Structure.

2.1. The PETSc.

The *PETSc* emphasis is the algorithmic and discrete mathematics interface, where the programmer manipulates mathematical and algorithmic objects [2]. However, these modules were organized to provide many abstraction levels: the user can manipulates objects associated with the application through an application-specific interface, by using *PDE* Solvers or Linear Solvers, or he can use the low-level computation kernels, like *BLAS*-type operation or *FFTs* [Fig. 1].

The basic objects structures (vectors, matrices and index sets) can be distributed, and then will be resolved by the computation and communication kernels. These low level kernels are the *BLAS*, *LAPACK* and *MPI* sub-libraries.

All the *PETSc* communication follow the *MPI* (*Message Passing Interface*) standard. In *MPI* the parallelism is explicit, that is, the programmer is responsible for jobs and data distribution. Thus, the problem is divided in small parts, and then distributed for the others nodes do the processing.

The *BLAS* routines provide standard building blocks for performing basic vector and matrix operations. It is subdivided in three levels: first level performs scalar, vector and vector-vector operations; the second performs matrix-vector operations; and the third level performs matrix-matrix operations.

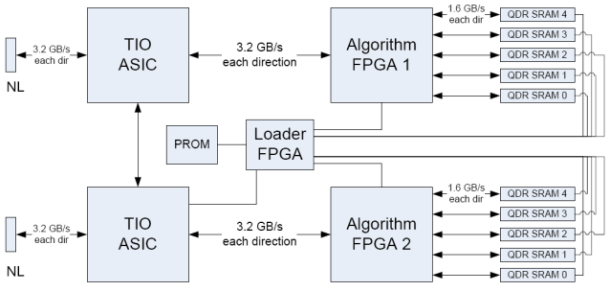

Figure 2 - RC100 blade achitecture.

Due to the efficiency, portability, and wide availability, the *BLAS* blocks is commonly used in the development of high quality linear algebra software [3]. The *LAPACK* (*Linear Algebra Package*) is one example. The *BLAS*/*LAPACK* don't follow the *PETSc* library, they are into an external package, that the user have to download in the *PETSc* installation [4] in accordance with the code language – *f2cblaslapack* to *C/C++* codes [5]. It is a *BLAS* in *FORTRAN* conversion to *C*, by freeware software called *F2C*, which are the level 1 *BLAS* and complete *LAPACK*.

2.3. RASC System

The SGI® RASC™ is a supercomputer with massive data processing proposes. The system consists of two distinct architectures: one is the Altix 350 and the other is the RC100 blade [Fig. 2].

The Altix 350 is an Intel based system, with two Itanium 2 processors, up to 12 GB DIMM RAM, four slots PCI/PCI-X and supports DVD-ROM, SCSI, Ethernet and USB. The system allows adding more processors and memory modules by expansion modules, up to 32 processors and 192GB memory [6].

The RC100 blade has two Xilinx® Virtex 4 LX200 FPGAs, 80MB QDR SRAM and is connected to Altix through the NUMAlink 4 technology, allowing the FPGAs extremely high bandwidth (up to 3.2GB/s per FPGA). Figure 4 shows RC-100 architecture [6].

All the necessary resources to develop to this platform are provided for the hardware/software codesign through the RASC Abstraction Layer (RASCAL) and RASC Core Services. The RASCAL is a set of API that abstracts the hardware to provide deep and wide scaling or direct and specific control over each hardware element in the system [6]. The Core Services is user independent and is pre-synthesized, made up of all the functional blocks excluding the algorithm block (*SSP port, global clock generation and control, independent read and write ports to each of the three logical or five physical random access memories, DMA engines, control and status registers and hardware debugging registers*) [6].

2.4. The Software/Hardware Codesign

Software/Hardware codesign can be defined as the simultaneous design of both hardware and software, involving cooperative specification, development, verification and management. The codesign techniques

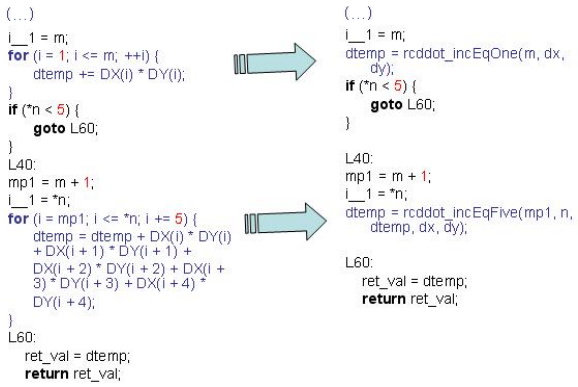

Figure 3 - Migration process.

includes: a Co-specification, where the function will be decomposed to create a conceptual model of the system and software and hardware functionalities are considered; a Co-development, where the hardware and software interfaces are developed; and Co-verification, where the SW/HW partitioning will be optimized and refined according to the requirements [7].

3. BLAS ON RASC: METHODOLOGY

Through *MPI* is possible to decompose the computation in three models [8]: the Multiple Program Single Data (*MPSD*), the Multiple Program Multiple Data (MPMD) and the Single Program Multiple Data (*SPMD*), that is the aim of this work.

In SPMD parallelism model, it was noted that BLAS kernel is not distributed, and will be integrally computed in one node. Thus, the strategy of acceleration is converting the BLAS in a hardware algorithm and defines software/hardware wrappers according with the chosen platform.

3.1. The Migration Process

The migration process is to identify the massive processing blocks in BLAS routines – generally inner loops – and convert this code blocks in dynamic call for new functions executed in *FPGA* [Fig. 3]. This design flow provides transparency to user's applications, keeping the main characteristics of *BLAS* library, with same signature of functions prototype. The use of *FPGA* implementations results in a more efficient execution with small computation time. There are, in literature, common performance increases about 20 to 50 times, with migration of sequential massive operations for hardware parallel execution [10].

The RASC System was chosen due to your hardware and software wrappers, and all peripheral subsystem, was developed through the Core Services and RASCAL API. It's only necessary to develop the hardware algorithm and the RASCAL API calls. It's only necessary to develop the hardware algorithm and the dynamic calls to the RASCAL API.

In this way, the new library with the Reconfigurable BLAS (RC BLAS) will be fully compatible with legacy

```
void function(DataType* A, DataType* C) {
    char resrv_name[] = "my_fpga";
    char alg_name[] = "alg";
    int num_devices = 2;
    int msg, id;

    msg = rasclib_resource_reserve(num_devices, resrv_name);
    msg = rasclib_resource_configure(alg_name, num_devices,
                                     resrv_name);
    id = rasclib_algorithm_open(alg_name, RASCLIB_BUFFERED_IO);

    msg = rasclib_algorithm_send(id, "a_in", Vector,
                                 sizeof(DataTpe) * SIZE);
    rasclib_algorithm_go(id);
    msg = rasclib_algorithm_receive(id, "c_out", C,
                                 sizeof(DataType) * SIZE);

    rasclib_algorithm_commit(id, NULL);
    rasclib_algorithm_wait(id);

    rasclib_algorithm_close(id);
    rasclib_resource_return(alg_name, num_devices);
    rasclib_resource_release(num_devices, resrv_name);
}
```

Figure 4 - Dynamic call hypothetic example.

code and will work in RASC. And the sentence containing only the dynamic call resolution, will be resolved in RC100 blade by RASCAL API.

3.2. The Dynamic Call and Algorithm Architecture

The main flow of execution, consists of queue a list of commands to scaling kernel (allocate resources, configure the resources, open the hardware algorithm, send data and parameters, starts the processing, receive the results) and wait the done signal [Fig. 6]. Then, the resources are unallocated and the results are delivered to main function.

A possible and simple architecture for hardware algorithm is one memory control, two data buffers and the algorithm hardware description (HDL) [Fig 7]. The memory control have be capable of request data to Core Services, receive it and feed the Input Buffer, withdraw the results from Output Buffer and starts (or not) the algorithm, according to his implementation. The Buffers can be from a register to a more complex structure, like *FIFOs* or caches, according to the algorithm's implementation.

5. CONCLUSIONS AND FUTURE WORKS

This work presents strategies, in scope of *HPCIn* project, for extension of the *PETSc* library applications for reconfigurable computer systems, used as high performance scientific problems solver. The *HPCIn Project* is a collaborative research, in development in the *Center of Informatics*. This research allows the implementation of data massive computation in high performance applications and integrates the *PETROBRAS*, in partnership with the *FINEP*.

The *SPMD* parallelism allows migrate only the *BLAS* kernel to hardware processing. The proposed methodology (identify and convert data massive processing inner loops into dynamic call) provides transparency and portability to user's applications, and the performance can be increased about 20 to 50 times.

The *RASC* platform uses *FPGA* technology to develop a full-featured reconfigurable computer, connecting *FPGAs* into the *NUMAlink* fabric making them a peer to

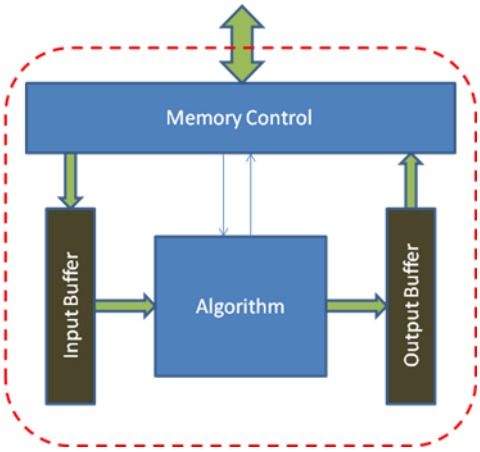

Figure 5 - Algorithm Block architecture.

the microprocessor and providing both high bandwidth and low latency. The Core Services is the key component of *RASC* which facilitates execution of the user algorithm in the algorithm block. It helps in synchronizing the I/O, memory, and Algorithm Block operations.

The migration of a new dynamic call for RASC system consists to develop a software control with a well defined flow and a hardware block that will execute the algorithm.

In the near future we intend: apply the methods for the `ddot` and `dswap` BLAS functions (vector-vector dot product and vector-vector swap operation) for the first use case; measure increases performance; develop the design flow and hardware implementation for the f2cblaslapack.

6. ACKNOWLEDGEMENTS

We would like to thank the *National Council of Technological and Scientific Development (CNPQ)* and *Funding for Studies and Projects (FINEP)* for funding, to the *Petrobrás Ressearch Center (CENPES)* and to the *Cooperative Research Net in Computational Modeling (RPCMod)* coordination, and to *Center of Informatics (CIn - UFPE)* for the physical structures (laboratories and PCs).

7. REFERENCES

[1] *PETSc Release Version 2.3.3*. Hosted in <ftp://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-2.3.3-p3.tar.gz>. Access in 07/25/2007

[2] *Introduction to PETSc: Presentation Objectives*. Hosted in <https://lineal.developer.nicta.com.au/support/workshops/slidesNIPS05/barry_NIPS2005.pdf>. Access in 07/25/2007.

[3] *BLAS*. Hosted in <http://www.netlib.org/blas/>. Access in 07/26/2007.

[4] *PETSc Installation*. Hosted in <http://www-unix.mcs.anl.gov/petsc/petsc-as/documentation/installation.html>. Access in 07/25/2007.

[5] *External package f2cblaslapack*. Hosted in <ftp://ftp.mcs.anl.gov/pub/petsc/fblaslapack.tar.gz>. Access in 07/25/2007.

[6] *RASC User's Guide*. SGI, 2007.

[7] J. Takalo, J. Kääriäinen, P. Parviainen, and T. Ihme. *Challenges of software-hardware co-design*. Access in 08/27/2008. Hosted in <www.vtt.fi/inf/pdf/working papers/2008/W91.pdf>.

[8] William GROPP, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, 1994.

[9] L. Zhuo and V. K. Prasanna. *High Performance Linear Algebra Operations on Reconfigurable Systems*. Hosted in < ieeexplore.ieee.org/iel5/10435/33129/ 01559954.pdf?arnumber=1559954 >. Access in 07/09/2007.

[10] *FPGA Acceleration in HPC: A Case Study in Financial Analytics*. Hosted in <http://www.xtremedatainc.com/pdf/FPGA_Acceleration _in_HPC.pdf>. Access in 07/26/2007.