Realization of an HPF Interface to ScaLAPACK with Redistributions

Thomas Brandes
SCAI, GMD *
Thomas.Brandes@gmd.de

David Greco
ParComp, CRS4 †
David.Greco@crs4.it

Abstract

The High Performance Fortran (HPF) programming language provides the data parallel programming paradigm for high performance architectures with different hierarchies of memory. HPF programs are much easier to write and to read than conventional message passing programs.

Unfortunately, the data parallel programming paradigm is not sufficient for all kind of applications and message passing programs are more efficient in some cases. Therefore it is desirable to have an interface from HPF to existing parallel libraries based on the efficient message passing paradigm.

In this paper we describe the realization of such an interface to ScaLAPACK, a library of high performance linear algebra routines based on message passing. The interface is realized in such a way that it applies redistribution routines from the HPF runtime system to its arguments if ScaLAPACK does not support a certain HPF distribution or if it can increase the performance.

In fact, the high level specification and the powerful functionality of HPF will simplify the use of a parallel library dramatically and make its use much more convenient.

1 Introduction

High Performance Fortran (HPF) is the new de facto standard language for writing data parallel programs for shared and distributed memory parallel architectures [5]. HPF provides a much more friendly programming paradigm for users, who just want their results fast, without needing to worry about details of the parallelism.

An HPF program should run effectively on a wide range of parallel machines, including distributed memory machines. On these, programmers have become used to writing "message passing" programs, e.g. using the portable message passing interface PVM [8] or MPI [6]. HPF programs are much easier to write and to read than conventional message passing programs. They contain no message passing at all though the compiler has to insert some into the final code.

*Institute for Algorithms and Scientific Computing, German National Research Centre for Computer Science, Schloss Birlinghoven, P.O. Box 1319, 53754 St. Augustin, Germany

†Parallel Computing Group, CRS4 - Center for Advanced Studies, Research and Development in Sardinia, Via N. Sauro 10, 09123 Cagliari, Italy

It is obvious that the data parallel programming paradigm is not sufficient for all kind of applications and that message passing programs are more efficient in some cases. Therefore it is desirable to combine both programming models. Especially the interface from HPF to existing parallel libraries based on the efficient message passing paradigm is very important.

Usually, the interface to parallel libraries within message passing programs is rather complicated as the call to a parallel routine requires also descriptors specifying the mapping of the array arguments. As HPF uses already array descriptors internally, the HPF interface becomes simple and straightforward for the user.

The other advantage of using an HPF interface to a parallel library is given by the fact that HPF provides redistribution routines in its runtime system for all the data mappings of HPF. As in most cases parallel libraries do only support some mappings of their arguments, their use can be easily generalized for all the HPF mappings. Users have no longer to implement their own redistribution routines.

In this paper we describe how such an interface has been realized within the public domain HPF compilation system ADAPTOR [1] and the HPF runtime system HPParLib++ [4]. We discuss it for ScaLAPACK, a very efficient library for linear algebra routines. The CSCS provides also an HPF Interface to ScaLAPACK [7].

2 Used Compilers and Software Packages

2.1 The ADAPTOR HPF Compilation System

ADAPTOR is a public domain High Performance Fortran compilation system and has two major components (see figure 1). A source-to-source transformation translates the data parallel program written in HPF to an equivalent SPMD program (single program, multiple data) working on the local parts of the distributed arrays and exchanging the data via message passing. The generated program must be linked with the DALIB runtime system that is the second component of the ADAPTOR system.

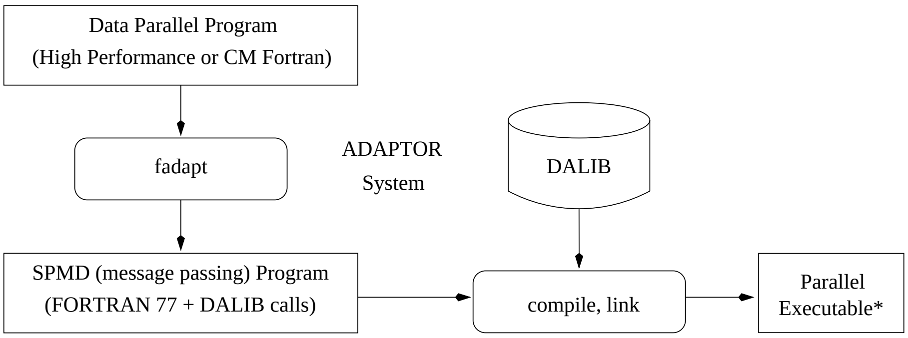

Figure 1: Overview of the ADAPTOR tool

2.2 HPParLib++

HPParLib++ [4] is a runtime system for HPF that is written in C++. The current version supports all the HPF models of data distribution as well as every kind of redistribution between these models. This also includes the block-cyclic distributions.

An interface between the ADAPTOR translation system and the HPParLib++ is now available. By this way, this runtime system can also be used as an HPF runtime system for ADAPTOR. Currently, there are some restrictions in the use that we want to get rid of in the near future.

2.3 ScaLAPACK

ScaLAPACK is a library of high performance linear algebra routines for distributed memory MIMD computers [2]. It contains routines for solving systems of linear equations, least squares and eigenvalues problems. One component of the ScaLAPACK software is the PBLAS (parallel BLAS) library whose interface is as similar to the BLAS as possible. These libraries operate on matrices distributed in a 2D block cyclic layout, the same distribution as incorporated in the HPF standard.

An example of a ScaLAPACK routine is the following one (`PDGETRF` computes an LU factorization of a general $M \times N$ distributed matrix A using partial pivoting with row interchanges):

```
      SUBROUTINE PDGETRF( M, N, A, IA, JA, DESCA, IPIV, INFO )

*     .. Scalar Arguments ..
      INTEGER            IA, INFO, JA, M, N
*     ..
*     .. Array Arguments ..
      INTEGER            DESCA( * ), IPIV( * )
      DOUBLE PRECISION   A( * )
```

3 Realization of the HPF Interface to ScaLAPACK

3.1 The User Interface

In the context of HPF, ADAPTOR provides an user-friendly high level interface to all the parallel routines.

```
      INTERFACE

         SUBROUTINE HPF_DGETRF (A, IPIV, INFO)
         DOUBLE PRECISION A(:,:)
         INTEGER IPIV(:)
!HPF$    INHERIT A
!HPF$    INHERIT IPIV
         INTEGER INFO
         END SUBROUTINE HPF_DGETRF

      END INTERFACE
```

The user can call all the routines of the ScaLAPACK via a corresponding HPF_xxx routine. The semantic of these routines is exactly the same as the semantic of the corresponding ScaLAPACK routines.

```
      PROGRAM TEST
      include "ScaLAPACK.h"
      DOUBLE PRECISION, DIMENSION (N,N) :: A, X
      INTEGER IP(N), INFO
!HPF$ DISTRIBUTE (*) :: IP
!HPF$ DISTRIBUTE (BLOCK,BLOCK) :: A, X
      ...
      CALL HPF_DGETRF (A, IP, INFO)
      CALL HPF_DGETRS ('No', A, IP, X, INFO)
      PRINT *, 'solved RHS, info = ', INFO
      ...
      END
```

There are the following differences between the HPF calls and the calls from a message passing program:

- The number of arguments related to one array is now exactly one argument. As in HPF an array or section descriptor is passed, the necessary information for the correct call to ScaLAPACK can be extracted from this descriptor.
- Many ScaLAPACK routines require workspace arrays of a certain size that must be provided by the user. These workspaces will now be allocated dynamically and is completely hidden for the user.

3.2 Realization of the Interface in DALIB, HPParLib++

HPF provides the EXTRINSIC mechanism for interfacing HPF programs with message passing programs. In fact, this mechanism could be used to write a portable interface. We decided to realize this interface by implementing it directly in C or C++ using the functionality of our HPF runtime systems. These are the reasons:

- Though the EXTRINSIC(HPF_LOCAL) mechanism supports redistributions if an interface is provided, it will not allow a flexible redistribution depending on the distribution of the input arguments.
- In some situations we can deal with subsections of arrays without creating temporary arrays.
- We can directly extract all the relevant information from our internal HPF array descriptors and save some additional overhead.
- The interface between our message passing system and the BLACS routines could be realized more convenient and more efficient.

```
void FUNCTION(hpf_dgetrf) (a_id, ipiv, info)

section_info *a_id;     /* section_info is a struct of the DALIB */
int ipiv[];
int *info;

{ int m, n, ia, ja;
  int error;
  int desc_A [8];
  char *A;

  error = dalib_get_section_info (*a_id, &m, &n, &A, &ia, &ja, desc_A);

  ....   /* check the argument, make a redistribution if necessary */

  pdgetrf (&m, &n, A, &ia, &ja, desc_A, ipiv, info);

  ....   /* redistribute again if necessary */

} /* hpf_dgetrf_ */
```

The routine `dalib_get_section_info` is responsible for translating the section descriptors used in our HPF runtime system to the parameters used in the ScaLAPACK routines.

ScaLAPACK provides also some efficient routines for array redistributions [3] to increase the performance of many routines.

3.3 Implementation Problems

The following problems have been identified during the implementation of the interface:

- The combination of the BLACS in ScaLAPACK with the message passing in our HPF runtime systems required some adaptation. Especially the processor grids in the BLACS must be matched to the abstract and physical processor arrays of HPF.

  Furthermore, our interface does not work with PVM. Both message passing systems require the PVM taskids and therefore both systems want to start the PVM processes. In other words, the `BLACS_setup` cannot be called as PVM has already been started by the HPF runtime system.

- If one array argument is not legal for the call, a redistribution takes place. The functionality for the redistribution itself is already available. But finding a good strategy for choosing the correct distribution is still an open problem.

- Error messages coming from the ScaLAPACK are not of any use. Therefore it is necessary that checking the arguments is already done one level higher in the interface routines.

4 Conclusions

Using HPF to provide an interface to parallel libraries is a desirable goal. The simpler interface and the high capabilities of HPF for redistributions will help to get the libraries better used.

Parallel libraries are expected to support a certain subset of distributions known from HPF. If one HPF distribution is not supported, the HPF runtime system provides the necessary functionality for redistributions.

The main problem is that the HPF runtime system and the parallel library have to use the same message passing context. In other words, a given mapping of the array in one system must have its counterpart in the other system. Some kind of standardization might be required. Also the mapping of abstract processors to physical processors that is not standardized in HPF must have its counterpart.

Good algorithms for choosing the best redistribution are necessary. For very good solutions these algorithms might need information about the machine.

Acknowledgements

For the many valuable technical discussions regarding this paper we are indebted especially to Bernard Tourancheau (LIP, Ecole Normale Supérieure de Lyon, France), Frédéric Desprez (LaBRI, University of Bordeaux, France), and Jack Dongarra (University of Tennessee, USA).

References

[1] Th. Brandes and F. Zimmermann. ADAPTOR - A Transformation Tool for HPF Programs. In K.M. Decker and R.M. Rehmann, editors, *Programming Environments for Massively Parallel Distributed Systems*, pages 91–96. Birkhäuser Verlag, April 1994.

[2] J. Choi, J. Demmel, I. Dhillon, J. Dongarra, A. Ostrouchov, S. Petitet, K. Stanley, D. Walker, and R.C. Whaley. ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance. Technical Report LAPACK Working Note 95, Department of Computer Science, University of Tennessee, 1995.

[3] J. Dongarra, L. Prylli, and B. Tourancheau. Array Redistribution in ScaLAPACK using PVM. In J. Dongarra, M. Gengler, B. Tourancheau, and X. Vigouroux, editors, *EuroPVM'95: Second European PVM Users' Group Meeting, Lyon, France*, pages 271–276. Hèrmes, Paris, September 1995.

[4] D. Greco and G. Cabitza. HPParLib++: A run time system for HPF. Technical Report CRS4-PARCOMP-95/1, Centre for Advanced Studies, Research and Development in Sardinia, Cagliary, Italy, February 1995.

[5] High Perforamnce Fortran Forum. High Performance Fortran Language Specification. Version 1.1, Department of Computer Science, Rice University, November 1994.

[6] Message Passing Interface Forum. Document for a Standard Message-Passing Interface. Draft Version February 1994 1.0, University of Tennessee, Knoxville, February 1994.

[7] Y. Murakami. High Performance Fortran Interface for ScaLAPACK. Extended abstract, sipar workshop on parallel and distributed systems, biel-bienne, switzerland, CSCS, Mann, Switzerland, 1995.

[8] V. Sunderam. PVM: a Framework for Parallel Distributed Computing. *Concurrency: Practice and Experience*, 3(10), December 1990.