Outstanding Issues in the MPIBLACS *

R. Clint Whaley †

November 6, 1997

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

Contents

*This work was supported in part by DARPA and ARO under contract number DAAL03-91-C-0047, and in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC05-84OR21400.

†Dept. of Computer Sciences, Univ. of TN, Knoxville, TN 37996, rwhaley@cs.utk.edu

1 Introduction

This note discusses some outstanding issues for the alpha release of the MPIBLACS. We hope that user input and the continuing MPI standardization effort will allow us to settle these issues, and thus proceed to a full release.

Most of these problems are closely related to issues being studied by the MPI committee. We have drafted a small note [4] describing our proposals for MPI, which expand on the sketches given here.

2 BLACS to MPI interface

It is important that users wishing to use the BLACS in conjunction with native MPI calls be able to do so. For this reason, the BLACS accept an MPI communicator when setting up the BLACS, allowing for an MPI user to call the BLACS. The reverse interface is also required. I.e., a user may be using the BLACS for the majority of his code, and then find he needs access to a MPI specific routine. The BLACS have therefore defined that the user can obtain an MPI communicator from a BLACS context through the support routine `BLACS_GET`.

What has not so far been defined is the specifics of this communicator. At the moment, the user knows that the returned communicator will have ranks corresponding to those returned by `BLACS_PNUM`, and little else.

Further refinement of this interface is probably required. It may be convenient, for instance, to define that two additional communicators will be cached on the communicator returned by `BLACS_GET`. These cached communicators would correspond to the process's process row and column.

Similarly, the BLACS/MPI interface might be further aided by specifying that the BLACS will have made the appropriate calls to `MPI_CART_CREATE` so that the communicator is set up such that calls to `MPI_CART_RANK` and `MPI_CART_COORD` will provide mappings between the BLACS' 2D view of the processes, and MPI's 1D view. It is not clear this will be required, as the BLACS have routines which can perform these operations (`BLACS_PNUM` and `BLACS_PCOORD`).

At any rate, we have not yet written code that requires this interface, and so it is difficult for us to specify exactly what this interface should be. We hope our continuing efforts in MPI usage, along with user input, will allow us to define this interface more completely.

3 Globally-blocking context operations

MPI has defined that when the user creates a new communicator from a previous communicator, the communicator creation routine will block until all processes in the original communicator call the routine. This means that if a new communicator is formed based on `MPI_COMM_WORLD`, for instance, then all processes must call the communicator creation routine.

Communicator creation occurs in the BLACS in the grid creation routines (`BLACS_GRIDINIT` and `BLACS_GRIDMAP`). These routines were originally designed so that only the those processes who are defined in the new grid need call the routines. This does not match what is required by MPI.

For now, we have little choice but to define that on MPI, all processes in the context passed to the grid creation routines must call the routine. It is our hope that MPI will relax this restriction.

4 Inter-language operations

The BLACS allow the user to mix Fortran77 and C code (via the Fortran77 and C interfaces). This means that you can, for instance, define a context using the Fortran77 routine `BLACS_GRIDINIT`, and then use that context as input for a C interface broadcast. This leads to problems on MPI, where inter-language operations are undefined. In particular, the translation between C and Fortran77 communicators is problematic.

At present, the BLACS use a variety of ways to handle this problem. First, regardless of whether the BLACS C or Fortran77 interface is called, the same MPI internals are used. The BLACS can call either the C or Fortran77 interface to MPI, based on user input. The problem of translating one communicator type to another still remains, however. The present (and unsatisfactory) solution to this problem is described below.

We anticipate that the next MPI standard will make allowance of inter-language operations, and that this will solve our inter-language problems.

4.1 Interim solution

The first thing that should be understood is when a communicator translation must take place. Communicators correspond to BLACS contexts, and are thus formed in the BLACS context creation routines, `BLACS_GRIDMAP` and `BLACS_GRIDINIT`.

As previously mentioned, the BLACS call the same MPI interface regardless of what BLACS interface is being called. Therefore, a communicator translation will only take place when the user calls `BLACS_GRIDMAP` or `BLACS_GRIDINIT` from a different language interface than the MPI interface the BLACS are internally using. Therefore, if the user calls `Cblacs_gridinit`, but the BLACS are calling the Fortran77 interface to MPI, a translation will have to take place. Similarly, if the user calls `BLACS_GRIDINIT`, and the BLACS are calling the C interface to MPI, a translation will have to take place.

The BLACS therefore offer two methods of getting around the translation problems. First, the BLACS support several methods of doing communicator translation. Second, we allow the user to indicate which internals the BLACS should call. The idea is that if translation can take place naturally on a user's machine, he is urged to set the translation method, and let the BLACS use the default internals. However, the user can also tell the BLACS to use a particular MPI interface. Thus, if he knows he will be calling mainly the Fortran77 interface, he can tell the BLACS to call the Fortran77 interface MPI, and thus avoid the translation altogether.

The BLACS offer preprocessor macros (which can be set in the makefile include file `Bmake.inc`) which allow the user to tune the BLACS to his system. These macros can be used to indicate how to translate communicators (the macro name is `TRANSCOMM`), or which

MPI interface to use (the macro is `WHATMPI`) The BLACS installation and testing guide [3] provides details describing these macros.

The BLACS have adopted the following strategy for the translation of communicators between C and Fortran77. If the user knows something about his MPI, he can tell the BLACS how to perform the translation. If the user is using MPICH or one of its derivities, the BLACS can call some MPICH internal internal routines to carry out the translation. On many systems, the C and Fortran77 communicators are the same anyway, and the BLACS can also handle this.

If neither of these translation schemes work, the BLACS do the following: when a translation must occur, all ranks in the context are translated to `MPI_COMM_WORLD`, and the new communicator is formed based on `MPI_COMM_WORLD` (it is assumed that ranks in `MPI_COMM_WORLD` are the same for both languages). This has the unfortunate side effect of making the routine doing the translation block on `MPI_COMM_WORLD`. Thus in situations where such a translation must take place, the user should be sure to have all processes call the grid formation routine.

5 Integer system contexts

A related issue is that a C communicator handle is not constrained to be an integer. This means that the user cannot standardly give it as input to the grid creation routines (these routines take integer system context handles as input). We hope that the next release of MPI will solve this problem by allowing translation of communicators between languages. If a language-neutral communicator is adopted, we may be able to take this as input. Otherwise, we may tell MPI users to translate to a Fortran77 communicator, which *is* constrained to be an integer.

As an interim solution, we have added routines which simply map C communicators to integer handles which can then be passed to the grid creation routines. These mapping routines are explained in turn below. Note that only user's using both MPI and the C interface BLACS will need these routines. User's using purely the BLACS will obtain a default system context (`MPI_COMM_WORLD`) via `BLACS_GET`, while user's using the Fortran77 interface already have integer context handles. Thus, in this scheme, users obtaining system contexts from `Cblacs_get` will get integer mapping handles, while users calling Fortran77's `BLACS_GET` will get the actual communicators. This means that users should be sure to call the same interface for their call to `BLACS_GET` and the grid creation routine.

5.1 The interim routines

5.1.1 SYS2BLACS_HANDLE

int Csys2blacs_handle(MPI_Comm SysCtxt)

SysCtxt (input) The system context to be mapped to an integer BLACS handle.

This function returns an integer handle which can be passed into the grid creation routines to indicate the MPI communicator (system context) SysCtxt.

5.1.2 BLACS2SYS_HANDLE

MPI_Comm Cblacs2sys_handle(int BlacsHandle)

BlacsHandle (input) The integer BLACS handle who's mapped system context is to be determined.

This function returns the MPI communicator (system context) which the integer handle `BlacsHandle` corresponds to.

5.1.3 FREE_BLACS_SYSTEM_HANDLE

void Cfree_system_handle(int BlacsHandle)

BlacsHandle (input) The integer BLACS handle who's mapping is no longer needed.

This routine may be called to dissassociate a BlacsHandle with a system context. Note that there is no need to do this, other than to keep memory usage down.

5.2 Example of usage for interim routines

```
   int bhandle;
   MPI_Comm mycomm, bcomm;
   .
   .
   .
/*
 * Translate mycomm to a BLACS handle
 */
   bhandle = Csys2blacs_handle(mycomm);
/*
 * Form BLACS context based on mycomm
 */
   icontxt = bhandle;
   Cblacs_gridinit(&icontxt, "r", 2, 2);
   .
   .
   .
/*
 * Free the handle associated with mycomm
 */
   Cfree_blacs_system_handle(bhandle);
   .
   .
   .
/*
```

```
 * Find the communicator the ranks returned by Cblacs_pnum are based on
 */
   Cblacs_get(icontxt, 10, &bhandle);
   bcomm = blacs2sys_handle(bhandle);
   .
   .
   .
```

6 MPI_INIT

In the present BLACS, there are some problems with MPI_Init. The problems are twofold. First, the standard states that the C interface MPI_Init takes pointers to argc and argv (arguments to main). It does not mention what happens if you wish to call MPI_Init from anything other than the main program. Many compilers define globally-addressable macros such as argc_ argv_ which allow the programmer to access these variables without explicitly passing them, but this is not ANSI standard. We are therefore unable to call MPI_Init from the BLACS, and so user's who's main is in C must explicitly call MPI_Init themselves. This is unsatisfactory, because the user no longer has a portable code, in the sense that if he wished to run on another BLACS platform, the MPI_Init call must be removed.

The second problem also involves language issues. In MPICH (the MPI implementation which has been used for BLACS testing), you must call the language version of MPI_INIT which matches your main program. I.e., if your main is in C, you must call MPI_Init(int *argc, char **argv), and if it is in Fortran77, the programmer must call MPI_INIT(IERROR). This feature makes it tough for a library such as the BLACS to operate correctly, as we cannot know in which language the user has written his main routine.

If the user's main routine is in C, but the BLACS go ahead and call Fortran77's MPI_INIT, MPICH calls some Fortran77 specific routines, which cause link errors. We have therefore split off the only BLACS routine which calls MPI_INIT, and place it into two seperate libraries. The C interface initialization library (the default name for this library is blacsCinit_<COMMLIB>-<PLAT>-<DEBUGLVL>.a) should be linked to if the user's main routine is in C; otherwise the user should link to the Fortran77 interface initialization library blacsF77init_<COMMLIB>-<PLAT>-<DEBUGLVL>.a).

We hope that the next draft of the MPI standard will define a way to cleanly call MPI_INIT in a non-language dependant fashion, as well as clearing up the argv, argc problems when the C version is called.

7 ANSI C

The MPIBLACS are released in ANSI C. We believe that most users have access to an ANSI standard compiler, and we can thus take advantage of the greater type and parameter checking available through ANSI C. This also avoids having to change our codes to avoid bugs in old C compilers which are no longer being actively supported, due to their replacement by ANSI C counterparts (an example is SUN's cc, replaced by acc or gcc).

These reasons, coupled with the fact that the MPI standard is specified in terms of ANSI C, have motivated us to provide the BLACS in ANSI C. User's unable to obtain ANSI C compilers should register their problem by mailing to `blacs@cs.utk.edu`.

8 Testing

The final reason we classify the MPIBLACS as an alpha release is simply that of testing. MPI itself is still relatively new, and finding a fully-featured and correct MPI is not always easy. However, MPI is already available on enough platforms that testing the MPIBLACS everywhere they might be ran is an impossible task. Further, the MPIBLACS have quite a few user-tunable parameters to allow them to acheive greater efficiency on the many different platforms supporting MPI. This means that user testing of codes will be absolutely necessary for us to become relatively confident in our implementation.

9 Conclusion

If you wish to offer suggestions, give a bug report, or have questions involving the BLACS, send mail to `blacs@cs.utk.edu`. Bug reports should be as concise as possible, with the smallest amount of code which can be used to demonstrate the error. You should consult the BLACS homepage (URL = `http://www.netlib.org/blacs/Blacs.html`), and the following papers for further insight into the BLACS: [1, 3, 4]. The MPICH homepage (URL = `http://www.mcs.anl.gov/mpi/mpich/index.html` privides good on-line MPI help, and [2] should be examined as well.

References

[1] Jack Dongarra and R. Clint Whaley. "A User's Guide to the BLACS v1.0". Technical Report UT CS-95-281, LAPACK Working Note #94, University of Tennessee, 1995.

[2] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard . *International Journal of Supercomputer Applications and High Performance Computing*, 8(3/4), 1994. Special issue on MPI. Also available electronically, the url is `ftp://www.netlib.org/mpi/mpi-report.ps`.

[3] R. Clint Whaley. "Installing and testing the BLACS". Technical Report UT CS-95-YYY, LAPACK Working Note #XXX, University of Tennessee, 1995.

[4] R. Clint Whaley. "Some Plebian Extensions to MPI", 1995.