Chapter Z01

Library Utilities

Contents

1 Scope of the Chapter

This chapter provides utilities that simplify the task of safely creating and terminating parallel applications using the NAG Parallel Library. The Library assumes the Single Program Multiple Data (SPMD) model of parallelism (see [4]) and uses a logical grid of processors defined by the Basic Linear Algebra Communications Subprograms (BLACS, see [1]). This Chapter also provides routines that return information appropriate to the message-passing system being used such as the MPI rank (see [3]) or the PVM task identifiers (see [2]) of the logical processors. Routines in this chapter also provide a convenient way to determine the workspace requirements for some of the numerical routines included in the Library.

No examples of the use of these routines are given in this chapter because their use is illustrated in the example programs for other routines in the NAG Parallel Library.

2 Background to the Problems

The aim of Chapter Z01 is to provide a mechanism for allowing novice users to quickly develop parallel programs while allowing experienced users the flexibility to develop more sophisticated programs. Since managing the processor grid (Library Grid) and obtaining processor grid information differ in the PVM-based version and the MPI-based version of the NAG Parallel Library, this chapter provides separate sections that describe the ways in which a Library Grid is set up and operated upon in a PVM or an MPI environment. Hence, users should read the sections appropriate the version of the Library being used.

2.1 Managing a Logical Processor Grid

Exception to a few routines (as described in the relevant routine documents), users should set up a 'Library Grid' prior to a call to the NAG Parallel Library routines. The underlying mechanism in which this Library Grid is created is different in the MPI-based and PVM-based versions of the Library.

2.1.1 PVM-based Version

The most important use of the routines in this chapter is to set up a rectangular grid of logical processors executing identical programs (according to the SPMD model). A simplified mechanism allows users to execute a single program and a sufficient number of identical copies of this program are automatically executed (spawned) to form the requested processor grid. However, users also have the option of creating processes themselves and simply using Chapter Z01 utilities to declare a logical processor grid for use by NAG Parallel Library routines, although this option should not be used unless the user is familiar with PVM and the BLACS. This allows the user to multigrid (an example is given in the NAG Parallel Library Tutorial, PVM-based Version). Note that in the current release of the NAG Parallel Library, only one Library Grid is permitted. When the logical processor grid has been defined, NAG Parallel Library routines may be called.

The BLACS assume a static physical machine; i.e., the number of processors never varies. In keeping with this assumption, the first call to the Z01 spawning utility, Z01AAFP, creates enough logical processors to form the requested grid, and this number never changes throughout the execution of the program. The user is allowed to reshape the grid (i.e., vary the dimensions of the grid) subject to the condition that the total number of processors never exceeds the number initially created.

A processor grid is uniquely identified by a 'context' for the purpose of communications and global operations within the grid. A context should be thought of as a pointer to a processor grid; its value need never be examined and should never be modified by the user. A processor which belongs to a particular logical grid is said to be 'in context' for that grid. Processors which are not in context continue to the next synchronisation point in the program (a point in the code that all processors must reach together before continuing execution). This synchronisation point will usually be where a processor grid is terminated or reshaped.

When the user has completed calls to NAG Parallel Library routines or wants to reshape the processor grid, the existing grid should be 'undefined' using the Z01 utility routine Z01ABFP. The user must indicate through an argument to Z01ABFP whether further Library Grids will be declared or if there will be no more calls to library routines.

PVM offers the user some alternative options in the way that tasks are spawned. The simplified spawning mechanism provided in the NAG Parallel Library only duplicates one of these alternative PVM options,

for use in debugging parallel programs. The other spawning options that PVM provides can be utilised if the users choose to spawn tasks themselves and simply use the NAG Parallel Library utilities for declaring the logical processor grid.

2.1.2 MPI-based Version

The most important use of the routines in this chapter is to set up a rectangular grid of logical processors executing identical programs (according to the SPMD model). A simplified mechanism allows users to execute a single program concurrently on a processor grid. However, users also have the option of creating processes themselves and simply using Chapter Z01 utilities to declare a logical processor grid for use by NAG Parallel Library routines, although this option should not be used unless the user is familiar with MPI and the BLACS. This allows the user to multigrid (an example is given in the NAG Parallel Library Tutorial, MPI-based Version). Note that in the current release of the NAG Parallel Library, only one Library Grid is permitted. When the logical processor grid has been defined, NAG Parallel Library routines may be called.

The BLACS (and also MPI) assume a static physical machine; i.e., the number of processors never varies. In keeping with this assumption, the first call to the Z01 utility, Z01AAFP, claims enough logical processors to form the requested grid. However, the user is allowed to reshape the grid (i.e., vary the dimensions of the grid) subject to the condition that the total number of processors never exceeds the number initially created by MPI.

A processor grid is uniquely identified by a 'context' for the purpose of communications and global operations within the grid. A context should be thought of as a pointer to a processor grid; its value need never be examined and should never be modified by the user. A processor which belongs to a particular logical grid is said to be 'in context' for that grid. Processors which are not in context continue to the next synchronisation point in the program (a point in the code that all processors must reach together before continuing execution). This synchronisation point will usually be where a processor grid is terminated or reshaped. A call to Z01AAFP also returns a context for the Library Grid and all Library routines use this context for safe communication. However, facilities are provided which allow users to use a context created independently in the Library routines (an example is given in the NAG Parallel Library Tutorial, MPI-based Version).

When the user has completed calls to NAG Parallel Library routines or wants to reshape the processor grid, the existing grid should be 'undefined' using the Z01 utility routine Z01ABFP. The user must indicate through an argument to Z01ABFP whether further Library Grids will be declared or if there will be no more calls to library routines.

2.2 Processor Grid Information

2.2.1 PVM-based Version

Each processor in the logical processor grid is identified by the coordinates of its position in the processor grid. The size of the processor grid and the position of each processor can be determined by the BLACS routine BLACS_GRIDINFO, but there is additional information that may be relevant for using the NAG Parallel Library. In particular, it is often necessary to identify the 'root' processor since this is the processor that is used to access standard input and standard output. If the NAG Parallel Library mechanism for spawning the processor grid is used, then the 'root' processor is always labelled {0,0} in the grid.

It is also useful to know which processors are 'in context' for the library routines and which are not, since it is possible (but not recommended for novice users) to give the out-of-context processors other useful work (but not to call additional NAG Parallel Library routines).

Additional information from the underlying PVM system can be obtained, in particular the task identifiers of each logical processor in the grid. This allows communication between logical processors using direct calls to PVM routines, although the use of the BLACS for inter-processor communication is strongly recommended.

The most efficient logical topololgies for broadcasts and global operations using the BLACS are also returned by Z01 routines.

2.2.2 MPI-based Version

Each processor in the logical processor grid is identified by the coordinates of its position in the processor grid. The size of the processor grid and the position of each processor can be determined by Z01BGFP. In particular, it is often necessary to identify the 'root' processor since this is the processor that is used to access standard input and standard output. If the NAG Parallel Library mechanism for creating the processor grid is used, then the 'root' processor is always identified by the process with rank 0 (this is usually labelled as {0,0} in the grid). The NAG Parallel Library provides a routine which returns the coordinates of the root processor in the logical processor grid.

It is also useful to know which processors are 'in context' for the library routines and which are not, since it is possible (but not recommended for novice users) to give the out-of-context processors other useful work (but not to call additional NAG Parallel Library routines).

The most efficient logical topololgies for broadcasts and global operations using the BLACS are also returned by Z01 routines.

2.3 Workspace Functions

Some of the routines in the NAG Parallel Library require a workspace array which must be suitably dimensioned. Typically the minimum size of the array depends on the size of the problem, the size of the logical processor grid and occasionally, the position of a logical processor within the grid. The size of the workspace is usually defined in terms of functions which are implemented in Chapter Z01.

2.4 Relationship to PVM and the BLACS

It is possible to make calls to PVM and BLACS routines from a program which uses the PVM-based version of the NAG Parallel Library. The PVM task identifiers returned by Z01BDFP and the BLACS context returned by Z01AAFP may be utilised to make direct calls to PVM and BLACS routines. See the NAG Parallel Library Tutorial (PVM-based version) for an example.

2.5 Relationship to MPI and the BLACS

It is possible to make calls to MPI and BLACS routines from a program which uses the MPI-based version of the NAG Parallel Library. The MPI rank returned by Z01BGFP and the BLACS context returned by Z01AAFP may be utilised to make direct calls to MPI and BLACS routines. However, if MPI routines are being called, users should first create an MPI communicator from the BLACS context for safe communication in their Library Grid. This can easily be achieved by a call to the BLACS routine BLACS_GET. See the NAG Parallel Library Tutorial (MPI-based version) for an example.

2.6 References

[1] Dongarra J J and Whaley R C (1995) A users' guide to the BLACS v1.0. *LAPACK Working Note 94 (Technical Report CS-95-281)* Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA.
URL: http://www.netlib.org/lapack/lawns/lawn94.ps

[2] Geist A, Beguelin A, Dongarra J J, Jiang W, Manchek R and Sunderam V (1994) *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing* The MIT Press, Cambridge, MA, USA

[3] Gropp W, Lusk E and Skjellum A (1994) *Using MPI: Portable Parallel Programming with the Message-Passing Interface* MIT Press, Cambridge, MA

[4] McBryan O A (1994) An overview of message passing environments *Parallel Comput.* **20** 417–444

3 Recommendations on Choice and Use of Available Routines

Note: Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Selection of Routine

Grid Management	
Purpose	**Routine**
Declare a processor grid	Z01AAFP
'Undefine' a processor grid or terminate library usage	Z01ABFP
Inform library mechanism of a user-spawned grid in PVM environment (only available in the PVM-based version of the Library)	Z01ADFP
Inform library mechanism of a user-created grid in MPI environment (only available in the MPI-based version of the Library)	Z01AEFP
Spawn debug sessions in PVM environment (only available in the PVM-based version of the Library)	Z01BFFP
Grid Information	
Identify 'root' processor	Z01ACFP
Grid coordinates of 'root' processor	Z01BAFP
Identify processors 'in context'	Z01BBFP
Return PVM information (only available in the PVM-based version of the Library)	Z01BDFP
Return BLACS topologies	Z01BEFP
Return MPI information (only available in the MPI-based version of the Library)	Z01BGFP
Workspace Functions	
Number of rows/columns of a cyclic 2-d block distribution held locally	Z01CAFP (NUMROC)
Workspace size for F08AEFP (PDGEQRF), F08AFFP (PDORGQR), F08ASFP (PZGEQRF) and F08ATFP (PZUNGQR)	Z01CBFP
Workspace size for F08AGFP (PDORMQR) and F08AUFP (PZUNMQR)	Z01CCFP
Identify the processor row/column that holds a matrix row/column in the cyclic 2-d block distribution	Z01CDFP (INDXG2P)
Workspace size for F08FEFP (PDSYTRD)	Z01CEFP

3.2 Programming Advice

The user must be careful to program according to the Single Program Multiple Data model of parallelism. It is possible to mimic other models of parallelism from within the SPMD model but this should be done with caution.

Examples of the use of the routines in this chapter are given for each numerical routine document and also in both versions of the NAG Parallel Library Tutorials. It is strongly recommended that you read the Tutorial before attempting to use the Library.