ARGONNE/GMD MACROS AND SUPRENUM COMMUNICATIONS LIBRARY: TOOLS FOR PORTABLE PARALLEL FORTRAN PROGRAMMING

R. Hempel
GMD (German National Research Center for Computer Science)
St. Augustin, Germany

1. EXTENDED ABSTRACT

The applications programmer's interest in software portability is obvious. He wants to code his problem in a form which is as independent as possible from the computer which he happens to use at the moment. In the world of sequential (von Neumann type) machines this portability has more or less been achieved with the definition of language standards like FORTRAN 77. For vector or parallel machines with shared memory usually there are vectorizers or parallelizers which automatically transform a sequential program to special code for the particular system. So, FORTRAN 77 still serves as a basis for portability, although this easy way of programming often does not yield optimal efficiency. The reason is usually an inefficient use of the memory hierarchy of the vector/parallel machine.

With local memory parallel machines the situation is much worse. Since languages as FORTRAN 77 do not contain any means for expressing the locality of arithmetic operators, an automatic parallelizing tool has the extremely difficult task of detecting this locality itself, and to split up the data among the local memories accordingly. There are prototypes for this kind of tools (cf. Bast et al. (1989)), but at the moment there is no satisfying solution for general programs. So, the partitioning of data and instructions has to be programmed explicitly by the user. Access to non–local data is provided by messages which the user also specifies. However, there don't exist any standards for syntax or semantics of the extensions to the sequential language, and, as a consequence, parallel programs for local memory machines usually are not portable.

At Argonne National Laboratory already in 1983 a solution was found for this portability problem (cf. Lusk et al. (1983)), in that case, however, using the shared memory programming model of process synchronization and monitors. The approach was to define machine independent macros for all parallel extensions and to implement those macros on different machines. The user writes his code in terms of the macros which then are expanded before compilation by the m4 macro expander (a standard UNIX tool). Later a similar macro package was defined for the local memory programming model (cf. Lusk et al. (1987)). In the beginning only C was supported, but in cooperation of ANL and GMD a corresponding package was developed for FORTRAN. These macros meanwhile are implemented on a broad variety of parallel systems, including not only the Intel iPSC's and SUPRENUM, but also workstation networks and shared memory machines.

From the user's point of view, writing a program in terms of the message passing macros is very similar to using the vendor–specific constructs. A careful implemen-

tation of the macros also results in a small time overhead as compared to the direct programming. This is particularly true for the Intel iPSC/2, where the worst case overhead in a send/receieve was measured to be less than 4 % (cf. Bomans et al. (1990)).

In the SUPRENUM project many parallel application programs had to be developed, mainly in the field of partial differential equations (PDE's) using multigrid techniques. It turned out that the communication parts of many programs were very similar, while the arithmetic parts obviously were problem dependent. The reason is that in most cases the same parallelization technique called "grid partitioning" was used, and this technique together with the underlying geometry determines the communication more than the particular form of the equation to be solved. In order to avoid multiple work, a central SUPRENUM Communications Library was defined which contains subroutines for all communication tasks arising in the parallelization of PDE solvers using finite difference discretizations. The first version of the library could only handle logically rectangular domains in two and three dimensions, but later routines were added for more general geometries as they occur in CFD codes.

If one compares the way of direct programming with the use of the Communications Library, the latter solution is advantageous in more than one sense. First, it saves the user much work. In typical situations up to 50 % of the total code can be taken from the library. Also, debugging is much easier, since usually the communication bugs are those which are the most difficult to find, and debugging the library routines is not the job of the user. Last but not least, it is possible to implement the library on different parallel machines. Since the user program is completely free of parallel language extensions, it is then portable among all systems where the library is available. However, the SUPRENUM Communications Library itself contains some 50,000 lines of code, so that a new implementation is very much work. For that reason the newest version of the library has been based on the Argonne/GMD macros. Thus, only the macros have to be implemented, and the Communications Library itself is portable.

The combination of macros and library has another advantage. Sometimes a user has a communication task which is not included in the library. He can then use the library routines for all the other communication parts of his program, and write the missing routine in terms of the macros. So, he saves programming work by using some library routines and keeps his program portable.

2. REFERENCES

Bast, H, Gerndt, M., Thole, C., 1989: SUPERB – the SUPRENUM Parallelizer. SUPERCOMPUTER 30 (1989) 51–57.

Bomans, L., Hempel, R., Roose, D., 1990: The Argonne/GMD macros in FORTRAN for portable parallel programming and their implementation on the Intel iPSC/2. Parallel Computing 15 (1990) 119–132, North–Holland.

Hempel, R., 1987: The SUPRENUM Communications Subroutine Library for Grid–oriented Problems. Argonne National Laboratory Technical Report ANL–87–23.

Hempel, R., Schüller, A, 1988: Experiments with Parallel Multigrid Algorithms, Using the SUPRENUM Communications Subroutine Library. GMD–Studie 141, St. Augustin.

Lusk, E., et al., 1987: Portable Programs for Parallel Processors. New York, Holt, Rinehart and Winston, Inc.

Lusk, E., Oberbeek, R., 1983: Implementation of Monitors with Macros: A Programming Aid for the HEP and Other Parallel Processors. Argonne National Laboratory Report ANL-83-97.