REDUCE on a Massively Parallel System
Extended Abstract

Winfried Neun
Konrad–Zuse–Zentrum
für Informationstechnik Berlin
Heilbronner Strasse 10
D–10711 Berlin-Wilmersdorf
Federal Republic of Germany
Email: neun@zib–berlin.de

1 Introduction

In this paper we present the installation of the computer algebra system REDUCE [1] on a massively parallel system, namely the Cray-T3D system [3]. This is the continuation of work which started with the usage of REDUCE on workstation networks and non-massively parallel multiprocessor systems, which was presented at the Rhine workshop Karlsruhe in 1994 [8]. At the same time this proves the usability of the system design on a variety of parallel systems. The aim is to develop a portable but nevertheless efficient system which allows the use of a piece of standard computer algebra software on a parallel machine without needing to change any basic parts of the algebraic engine.

2 Why do we want to use a massively parallel system ?

The usage of a supercomputer instead of a workstation or a PC for computer algebra calculations is only justified with the same argument as for numerical codes, if one encounters problems which are not soluble on today's powerful workstations. The reasons for this may be an enormous cpu or memory requirement typical for many computer algebra applications, but a lack of mass storage may as well hinder the solubility of a problem. For technical reasons the workstation clusters with (real) shared memory are limited in the number of processors to a relatively small number. This technical argument implies that building massively parallel machines requires the usage of local memories attached to the individual processor which can be grouped in such a way that they form a virtual shared memory.

The usage of massively parallel machines with local memories and hopefully fast networking capabilities leads to the need for coarse grain parallelization. In spite of this, one can try to use similar techniques (e.g. "multimodular calculations") which are used in some fine grain parallelizations e.g. on the Cray vector processors [7], [6].

3 REDUCE and PVM

In order to remain portable, we are using a quasi standard communication namely, PVM from Oak Ridge et al. [2]. For our purpose, PVM is an ideal tool. It implements a reliable transport layer, it is available virtually everywhere and it is easy to use. We can build our LISP to LISP communication layer on top of REDUCE and PVM since PVM requires only the usage of a C library. Moreover, for the Cray MPP systems (like other vendors systems) a special version of PVM is available which (without the need to use non-standard interfaces) benefits from the virtual shared memory capabilities [4].

The basic communication protocol uses the exchange of LISP forms between processors. Depending on the type of process communication these forms can be evaluated immediately by the receiving processor and the result will be sent back to the sender of the original message (master/slave) or the forms are processed and the result will not be sent back, or alternatively, the forms may even be neglected completely. The protocol is described completely in [5].

4 PVM on the Cray-T3D

The Konrad–Zuse–Zentrum has owned a Cray–T3D with 192 nodes since autumn 1994. Our design for a parallel computer algebra system had to work on this system of course, but its usage is not restricted to this type of system.

Due to some specialities of the Cray-T3D operating system, the PVM system works differently here than on workstation networks.

The task control under PVM is limited. The number of processors which a program uses is fixed and all processes are started simultaneously (Single Program, Multiple Data). Therefore the communication model cannot be set up at the time when the processes are spawned, but has to be defined in a resource file read at startup. The actual communication type is set up during startup of the processes in a appropriate script.

5 Design issues

A couple of decisions regarding the design had to be made. The result of our negotiations can be found below:

- We want to use stock hardware and software. In this way we hope to keep our investment into the technical part small and independent of the luck of the vendors of parallel machines.
- We have to look for coarse grain parallelism in computer algebra applications.
- The REDUCE algebraic engine has to be used as it is.
- The user interface (described in [5]) must be usable in the REDUCE algebraic (user) mode.
- Binary transfers between processors are used in order to reduce system overhead.

- Even though the machine supports (virtual) shared memory we do not take advantage of this fact for the sake of portability. But we hope that some layers of the software will do that for us.

6 Examples

Several examples are included in [5] and [8], which present different communication models between processors (master/slave, pipeline etc). Some problems from symbolic linear algebra (e.g. big determinant calculations) are currently being investigated. In cooperation with several research groups we are trying to parallelize prominent REDUCE packages.

References

[1] Anthony C. Hearn: REDUCE User's Guide, version 3.5, The RAND Corporation, 1993.

[2] Oak Ridge National Laboratories: PVM 3 User's Guide and Reference Manual, version 3.2, May 1993.

[3] Cray Research Inc: CRAY T3D System Architecture Overview, HR-04033, Cray Research Inc, Eagan MN, 1993.

[4] Cray Research Inc: NEWS, Nov 15, 1993.

[5] Herbert Melenk, Winfried Neun: RR: Parallel Symbolic Algorithm Support for PSL Based REDUCE, Konrad Zuse Zentrum 1995.

[6] Herbert Melenk, Winfried Neun: Parallel polynomial operators in the large Buchberger algorithm, in J. Della Dora, J. Fitch (eds.): Computer Algebra and Parallelism, Academic Press, 1989.

[7] Winfried Neun, Herbert Melenk: Implementation of the LISP - arbitrary precision arithmetic for a vector processor, in J. Della Dora, J. Fitch (eds.): Computer Algebra and Parallelism, Academic Press, 1989.

[8] Winfried Neun: Using PVM based software for parallel computation in Computer Algebra, in J. Calmet (ed.): The Rhine Workshop on Computer Algebra, Karlsruhe, 1994.