A Parallel Version of FORM 3

D.Fliegner*, A.Rétey**, J.A.M. Vermaseren†

*Abt. Nichtlineare Dynamik, MPI für Strömungsforschung, D-37073 Göttingen, Germany
**Institut für Theoretische Teilchenphysik, Universität Karlsruhe, D-76128 Karlsruhe, Germany
†NIKHEF, P.O. Box 41882, 1009 DB, Amsterdam, The Netherlands

Abstract. The parallel version of the symbolic manipulation program FORM for clusters of workstations and massive parallel systems is presented. We discuss various cluster architectures and the implementation of the parallel program using message passing (MPI). Performance results for real physics applications are shown.

1. The Sequential Version of FORM

FORM (1) is a program for symbolic manipulation of algebraic expressions specialized to handle very large expressions in an efficient and reliable way. It is used non-interactively by executing a program that contains several parts called *modules*. The execution of each of these module is divided into several steps:

- **Compiling:** the input is translated into an internal representation.
- **Generating:** for each term of the input expressions the statements of the module are executed. In general this generates a lot of terms for each input term.
- **Sorting:** all the output terms that have been generated are sorted and equivalent terms are summed up.

The special properties of FORM result from the fact that with a few exceptions it allows *local* operations on *single* terms only (e.g. replacing parts of a term by another term or a whole expression, multiplying a term by a certain expression). As a consequence expressions can be handled as "streams" of independent terms that can be read in from a file and processed sequentially. The generation of terms is done in a way that the output terms drop out term by term. The output terms are partially sorted and stored in intermediate buffers and temporary sort files on disk. At the end of a module all the existing sorted patches are merged into a single sorted output stream of terms which is written to a file and used as input source of terms for the next module. The procedure is not limited by the memory of the system, but the maximum size of the sort files, i.e. the total disk space. This is what makes FORM particularly useful for large symbolic problems. Obviously, the restriction to local operations also allows a straight forward parallelization.

2. Evaluation of Parallel Platforms

For the parallelization of FORM, message passing and a corresponding library MPI(CH) (2) was chosen, because of its availability on a wide range of different architectures. It was clear from the beginning that the performance of regular networking hardware would not be sufficient. To give an impression of the performance of special communication hard– and software we show results for various combinations that we have used instead:

- Digital Unix Alpha cluster with 32bit-Myrinet (3)
  MPICH/ParaStation II (4)
  (TTP Universität Karlsruhe (5))
- Compaq Linux Alpha Cluster 64bit-Myrinet
  MPICH/ParaStation II
  (ALICE Universität Wuppertal (6))
- IBM SP2, MPI (RZ Universität Karlsruhe)
- Compaq Tru64 Unix Alpha SMP server
  MPICH using shared memory segments
  (TTP Universität Karlsruhe)

The MPI(CH) libraries have been examined using the Pallas MPI benchmarks (PMB) (7), which determines the throughput and latency of the message passing operations under different circumstances. Here we only present the results for the simplest possible single transfer benchmark, a ping-pong test: one process sends a message of *n* bytes to another process, which immediately sends that message back. In contrast to more complicated benchmarks there is no concurrency with other messages passing activity during this test, i.e. the bandwidth and latency are measured under optimal conditions. A Fast Ethernet gives a maximum throughput of 100 Mbit/s and a minimum latency of about 200 μs.

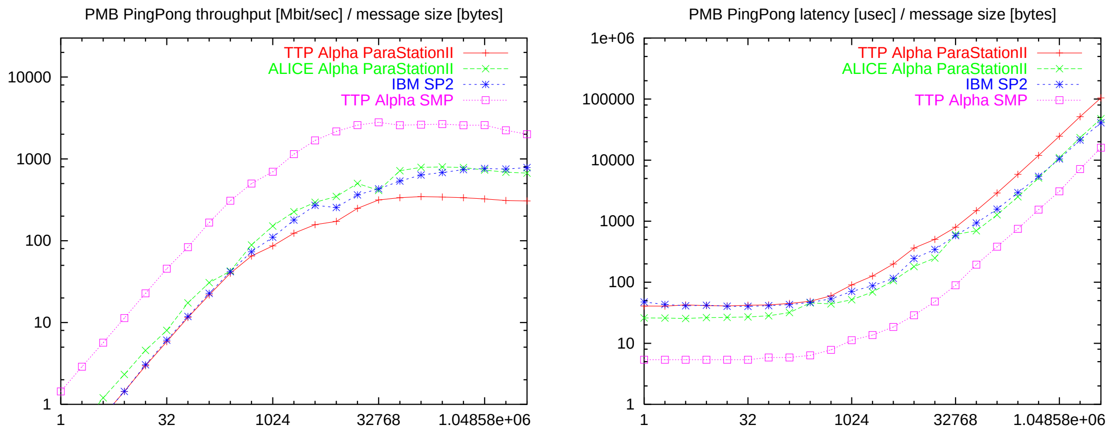

FIGURE 1. MPI(CH) bandwidth and latency results for the PingPong benchmark.

As can be seen in figure 1 the ParaStationII MPICH implementation on the 64bit Myrinet network yields about the same maximum throughput as the IBM SP2 (800 Mbit/s), but the maximum bandwidth on the Alpha SMP server is still considerably larger (more than 2.5 Gbit/s). In figure 1 (right) the measured latency is shown. The minimum latency (25μs) of the ParaStationII MPICH implementation using the 64bit Myrinet hardware is smaller than the minimum latency on the IBM SP2. The Alpha SMP server yields an even smaller latency (less than 7 μs) by using shared memory segments.

3. The Parallel Version: ParFORM

In a first step of the parallelization only the sorting procedure was parallelized. In this version the generated output terms are distributed by a master process among an arbitrary number of slave processes, which do a partial sorting. Finally the sorted patches are merged together by the master process.

In a second step also the generation of terms was parallelized. The input terms are distributed among all processes before generating the output terms. The distribution of terms is organized such that the master process sends a patch of input terms to each slave process at the beginning of the module and then waits for the slaves to ask for new terms whenever they have finished with their last patch of input terms. This concept also yields a decent load leveling automatically. In the current implementation the number of terms in a patch can be chosen at the start of the program. Of course choosing a too coarse grained distribution will result in the danger of running into worst cases, where all the work sits in only one of the input patches, and only one processor is busy. On the other hand, a fine grained distribution causes more overhead. The best setting turns out to be strongly dependent on the problem that is run: figure 2 shows the speedup for a combinatorics problem that amounts to ideal input for the parallel version. Beginning with the second processor a linear speedup is achieved even if single input terms ($N = 1$) are distributed. If a too coarse grained distribution is chosen ($N = 1000$ input terms per patch) the speedup is lost. In typical real physics applications (figure 3 shows the result for a complicated 3-loop Feynman diagram) things look different: the speedup is usually not linear anymore and there is obviously an optimum input patch size somewhere between $N = 10$ and $N = 100$.

Figure 4 shows the result for a SMP system with processors that are much faster than those of the clusters shown in figure 3. In spite of the faster inter-process communication through shared memory segments the speedup is smaller. This is a generic problem that can only be solved by omitting message passing and using threads instead.

4. Conclusion & Outlook

The parallel version of FORM based on MPI can be used on massive parallel computers, clusters of workstations and SMP systems. With only a few exceptions all the features of the sequential version are already operational in the parallel program as well. For ideal input programs ParFORM indeed yields a linear speedup beginning with the second processor. There is a considerable speedup in real physics applications, if the problem size is not too small. ParFORM is currently being optimized for huge clusters of workstations (e.g. ALICE (6)). We also work on a parallel version for SMP systems (e.g. TERAS (8)) using threads instead of message passing). The final aim is a version for clusters of SMP systems.

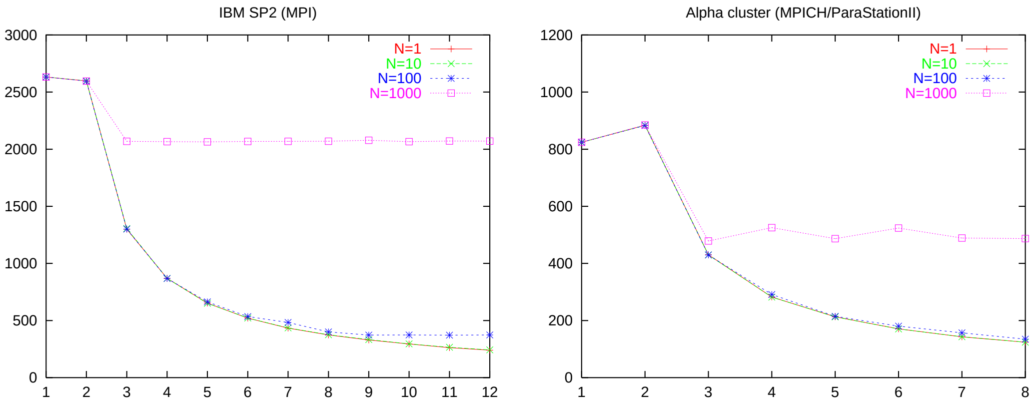

FIGURE 2. Runtimes [sec] vs. number of processors for ideal input on a IBM SP2 (left) and a Digital Alpha cluster using ParaStationII and a 32bit Myrinet network (right). Shown are the runtimes for different granularities of the input-term distribution.

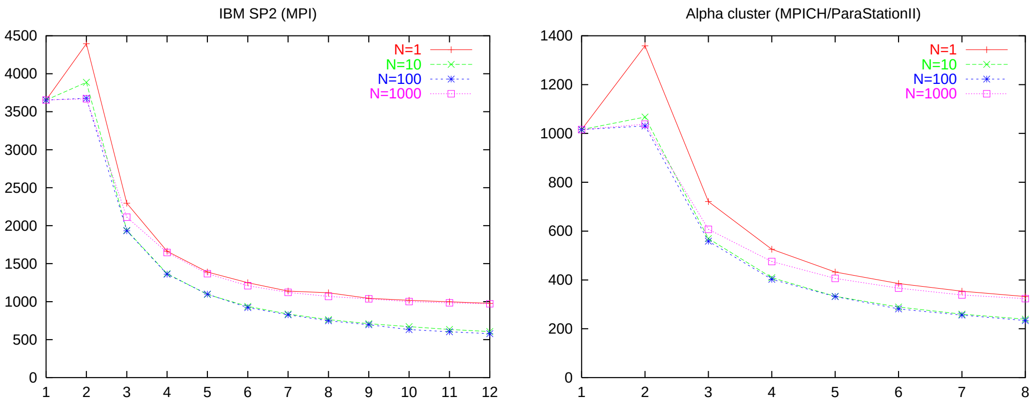

FIGURE 3. Runtimes [sec] vs. number of processors for a real physics application on a IBM SP2 (left) and a Digital Alpha cluster using ParaStationII and a 32bit Myrinet network (right). Shown are the runtimes for different granularities of the input-term distribution.

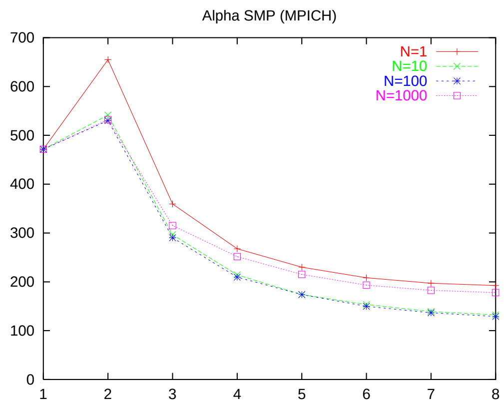

FIGURE 4. Runtimes [sec] vs. number of processors for the physics application of figure 3 on a Compaq Alpha SMP Server.

REFERENCES

1. FORM homepage: http://www.nikhef.nl/form
2. MPI(CH) homepage: http://www.mcs.anl.gov/mpi
3. Myricom homepage: http://www.myri.com
4. ParaStation homepage: http://parastation.ira.uka.de
5. TTP homepage: http://www-ttp.physik.uni-karlsruhe.de
6. ALICE homepage: http://alice.iai.uni-wuppertal.de
7. Pallas MPI Benchmarks: http://www.pallas.de/PMB2
8. TERAS homepage: http://www.sara.nl