ParFORM: Parallel Computer Algebra

H.-M. Staudenmaier, M. Steinhauser, M. Tentukov (Univ. Kalrsruhe)
J.A.M. Vermaseren (NIKHEF, Amsterdam)

We describe the computer algebra program ParFORM *which is the parallel version of the symbolic manipulation program* FORM*.*

1 Introduction

FORM [1] is a program for the symbolic manipulation of algebraic expressions. It is specialized to handle very large algebraic expressions of billions of terms in an efficient and reliable way. That is why it is widely used, in particular in the framework of perturbative Quantum Field Theory, where often several thousands of Feynman diagrams have to be computed. However, the abilities of FORM are also quite useful in other fields of science where the manipulation of huge expressions is necessary.

FORM is constructed in such a way that the size of the expressions is not restricted by the main memory of the computer but only by the space available on the hard disk. Actually in modern applications in particle physics it happens quite often that the size of intermediate expressions for each Feynman diagram may reach several hundred Megabytes up to a few Terabytes. As a consequence, even with FORM such calculations require a CPU time of several years despite the steady advancement of the hardware and the continuous improvement of the algorithms.

One of the most efficient ways to increase the performance is based on parallelization. In fact, the project to obtain a parallel version of FORM has been started at the end of the nineties. In the recent years ParFORM [2] has become a reliable tool which shall be described in this contribution.

2 The concept of ParFORM

A FORM program is in general divided into so-called modules which are terminated by a "dot"-instruction. During the execution of the program, which is only possible in the batch-mode, each module is processed separately one after the other which essentially occurs in three steps

- Compilation: The input is translated into an internal representation.
- Generating: For each term of the input expressions the statements of the module are executed. This in general generates a lot of terms.
- Sorting: All the output terms that have been generated are sorted and equivalent terms are summed up.

Next to a sophisticated pattern matcher, it is the strength of FORM that only local operations on single terms are allowed, like replacing parts of a term by some other expressions. Non-local operations like replacing a sum of two terms are not allowed. Non-local operations are allowed only implicitly, e.g., in the sorting procedure at the end of the modules, where equivalent terms are combined. At first sight this seems to be a strong limitation for the formulation of general and efficient algorithms. It is usually possible to get around this limitation by designing algorithms in

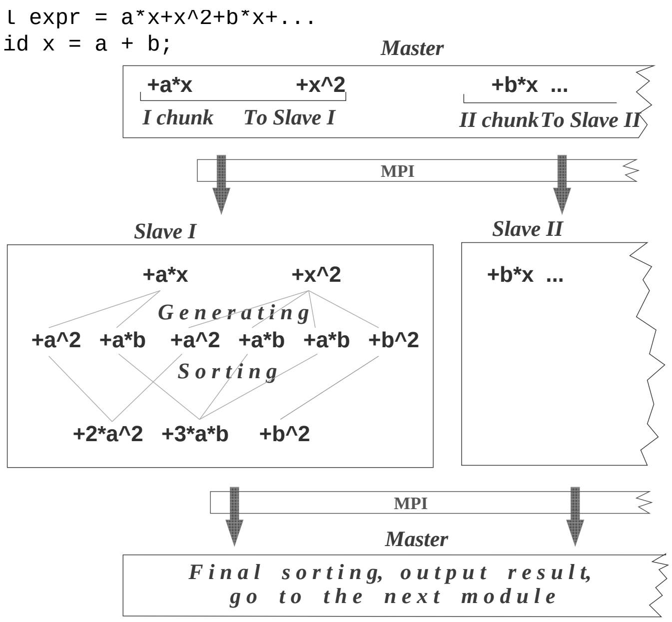


Figure 1: General conception of `ParFORM`.

clever and non-standard ways. Due to the locality of the operations it is possible to handle expressions as "streams" of terms that can be read sequentially from the memory or a file and processed independently. On one hand this enables `FORM` to deal with expressions that are larger than the available main memory and on the other hand it allows for parallelization.

The concept implemented for the parallelization is straightforward and indicated in Fig. 1: in a first step the master process splits the expression into pieces, so-called chunks. Each chunk is sent to one of the slaves where an independent `FORM` process runs, i.e. the module to be executed is compiled, the terms are generated, sorted and sent back to the master. Once all slave processes have finished their jobs the master performs the final sorting.

The communication between master and slaves is based on the message passing interface (MPI) standard[1] which provides a library for the data transfer between processes. Message passing permits to parallelize `FORM` on computer architectures both with shared memory, i.e. Symmetric Multi Processing (SMP) computers and on computer clusters. However, it should be mentioned that for our application there is a significant overhead in connection to MPI which is due to our relatively simple parallelization model.

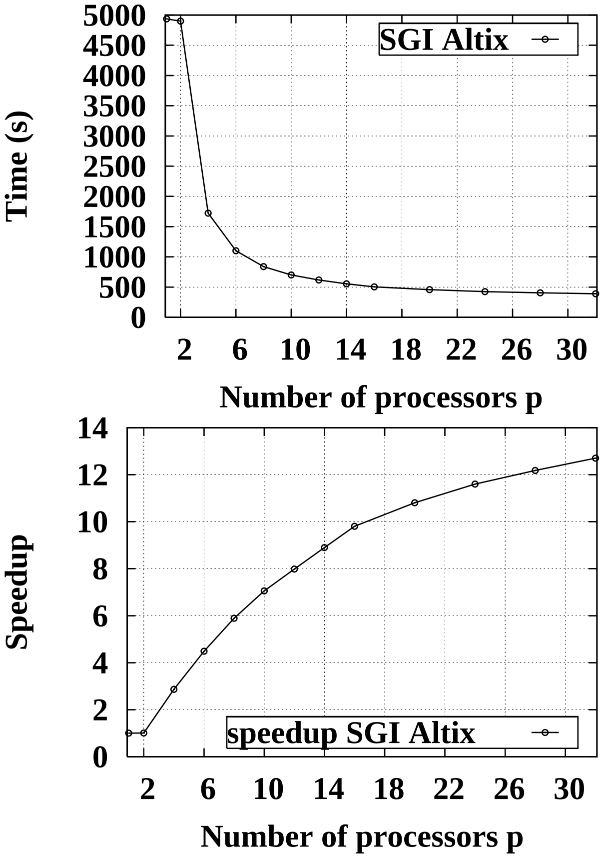
Recently this disadvantage could be overcome on SMP architectures with the help of a new model for the communication. In this approach the master explicitly allocates shared memory buffers which can be accessed both by the master and the slaves. In these memory segments the master prepares the chunks for the slaves, they are doing their job and the master collects the results again from the shared buffers. Thus, copying huge amounts of data is not necessary any more. The use of the shared-memory model on SMP machines led to an increase in the speedup of 20-25%. This concept is taken even further in `TFORM`, a multithreaded version of `FORM` which will be released shortly [3].

It is worth mentioning that the parallelization does not require any additional efforts from the user. It is possible to run the programs written for the sequential version using `ParFORM` and adding a specification concerning the number of processors. It is clear that different codes show a different performance and efficiency in the parallel version. In particular, in the current version of `ParFORM`, not all instructions are parallelized and thus the modules containing such commands are executed in the serial mode.

3 Results

At present, there are a number of calculations of physical quantities [4] which would not have been possible without the gain in performance and speedup provided by `ParFORM`. Most of the applications are connected to the evaluation of four-loop Feynman integrals which occur in the context of perturbative quantum field theory. In particular, there are algorithms which transform

[1] see `http://www.mpi-forum.org`



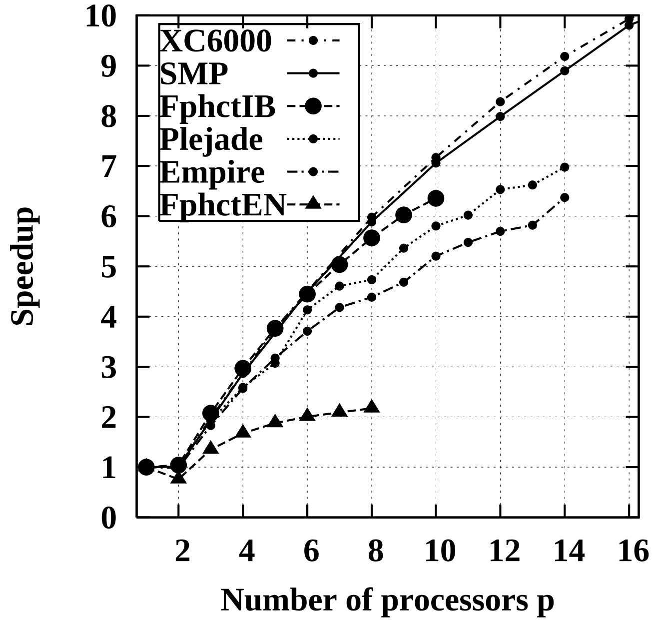


Figure 3: The speedup for the test program on different clusters in comparison to the SMP computer (cf. Fig. 2).

Figure 2: Runtime and speedup for the test program running on a SGI Altix 3700 server with 32 Itanium-2 processors (1.3 GHz).

the mathematical complexity of the original problem to the need of simple manipulations of rather large polynomial expressions which have billions or even more terms. Manipulation of this type constitute the basis of the speedup curves which are discussed in the following.

The results for the test program running on a SGI Altix 3700 server with 32 Itanium-2 processors are shown in Fig. 2 where both the runtime and the speedup (as compared to the sequential version) is shown as a function of number of processors, p, involved in the calculation. The almost horizontal line between $p = 1$ and $p = 2$ is due to the fact that for $p = 2$ one of the processors takes over the role of the master and the other one of the slave. Thus a real reduction of the CPU time only starts from $p = 3$. It is interesting to note that the speedup is almost linear up to twelve processors. Furthermore, for 16 processors the program is faster by an order of magnitude. As a consequence instead of years one only has to wait a few months in order to obtain the results of a calculation. This provides the possibility to consider qualitatively new kinds of problems, since in practice it is impossible to run a job for years whereas a few months are feasible nowadays. Beyond $p = 16$ the curve becomes more flat, however, the speedup is still considerable up to 32 processors.

Recently, `ParFORM` was successfully installed on several clusters. In Fig. 3 the corresponding speedup curves are shown and compared to the curve from Fig. 2 obtained on the SMP computer.

The cluster `XC6000` is a Hewlett Packard Itanium-2 QsNet interconnected cluster. This is the only tested cluster which demonstrates a better behaviour than the SMP computer, however, it is also significantly more expensive. `Fphctl` is a cluster consisting of 32-bit Xeon nodes. This cluster has been tested both with an Infiniband (`FphctlIB`) and a simple Fast Ethernet (`FphctlEN`) interconnection. Whereas the latter is not of interest in practice the former shows a quite reasonable behaviour following closely the SMP curve for a smaller number of processors. `Plejade` and `Empire` are both dual Opteron clusters. However, `Plejade` is interconnected using InfiniBand whereas `Empire` uses Gigabit Ethernet. Both clusters show a reasonable behaviour leading to a speedup of about six for ten processors.

We want to mention that the SMP curves shown in Fig. 2 and 3 are based on the shared-memory model mentioned above. On the other hand, for the clusters one has to rely on the MPI library which for our applications has a significant overhead.

4 Summary

The computer algebra program `FORM` is designed to handle huge expressions in a quite effective way. Still, for some physical applications even `FORM` would take several years which make a practical calculation impossible.

In the recent years a parallel version of `FORM`, `ParFORM`, has been developed and in the meantime it has become a reliable tool to perform computer algebra in parallel. `ParFORM` has demonstrated a good speedup behaviour both on SMP computers and on different cluster architectures. Furthermore, for the current version of `ParFORM` the `FORM` programs written for the sequential version need not to be modified.

In the next years the development of `ParFORM` will continue. In particular, both the slope of the speedup curves shall be increased and the flattening shall be pushed to higher number of processors. Furthermore, more `FORM` commands and options will be parallelized.

This work was supported by the DFG through SFB/TR 9.

References

[1] `FORM` version 3.0 is described in: J. A. M. Vermaseren, arXiv:math-ph/0010025;
for recent developments, see also: M. Tentyukov and J. A. M. Vermaseren, "Extension of the functionality of the symbolic program FORM by external software," arXiv:cs.sc/0604052;
see also: J. A. M. Vermaseren, Computeralgebra-Rundbrief, Nr. 36, März 2005.
`FORM` can be obtained from the distribution site at http://www.nikhef.nl/~form.

[2] M. Tentyukov, D. Fliegner, M. Frank, A. Onischenko, A. Retey, H. M. Staudenmaier and J. A. M. Vermaseren, "ParFORM: Parallel Version of the Symbolic Manipulation Program FORM," arXiv:cs.sc/0407066;
M. Tentyukov, H. M. Staudenmaier and J. A. M. Vermaseren, "ParFORM: Recent development," Nucl. Instrum. Meth. A **559** (2006) 224.
See also http://www-ttp.physik.uni-karlsruhe.de/~parform.

[3] M. Tentyukov and J. A. M. Vermaseren, "FORM version 3.2", to appear shortly. See also the distribution site of `FORM` at http://www.nikhef.nl/~form.

[4] For a recent application, see, e.g., P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, "Scalar correlator at O(alpha(s)**4), Higgs decay into b-quarks and bounds on the light quark masses," Phys. Rev. Lett. **96** (2006) 012003 [arXiv:hep-ph/0511063].