

# Some technical remarks on the Hirlam forecast model

Kalle Eerola  
Finnish Meteorological Institute

Jussi Heikonen  
Center for Scientific Computing

## 1 Introduction

This note discusses three technical issues of the Hirlam forecast model. In section two a method of parallelizing the output is presented. The basic idea is to reserve one processor only for output and do it simultaneously with the computing. Section three discusses the problem of reproducibility, i.e., does the Hirlam forecast model produce exactly the same result when running with different number of processors. This is discussed in the light of one experiment. In section four the scaling of the Hirlam forecast model on T3E and Fujitsu is discussed.

## 2 Method of parallelizing the output

Numerical models produce more and more output. Normally the forecast fields are nowadays required with a time interval of one hour. However, in the reference Hirlam this requirement decreases the efficiency in parallel environment, because Hirlam uses the normal strategy for output: only one processor encodes the fields into GRIB and does the actual writing to disk, while the other processors are idle during this time.

MPI-2 includes parallel I/O, but this does not help very much, because MPI-2 is not yet very widely available. There is also another limiting factor in using MPI-2 with Hirlam. In addition to writing to disk, the output phase also contains GRIB encoding, which is not very easily parallelized, because GRIB is bit-oriented, and because global minimum value is needed for the reference.

We took another way of looking at the problem. In this approach one additional processor is reserved only for output and it can do it while the others continue with computing. Another basic requirement in our approach was that the parallelizing of output should not restrict the format of the output file. So the actual writing is done by a subroutine given by the user and can be changed if needed. The first implementation has been done using the the default Hirlam file structure (Asimof).

In the following we list the basic features of the new parallel output method.

- Only the output is parallelized.
- Because MPI does not allow creating new processes in the fly, the forecast model is started with one additional processor.
- The last processor is reserved for output. Processor zero, which in the reference version does the output, now only controls the additional output processor. This minimizes the changes in the code.
- In the very beginning, a new communicator, which excludes the output processor, is created. This new communicator is then used instead of `MPLCOMM_WORLD` in all communications between the computational processors. Again this feature minimises the changes in the original code.

- The computational nodes use non-blocking send in all communications to the output processor. This means that they only initiate sending and then go on computing.
- Because of using non-blocking send, additional buffers are needed. All the data to be sent is copied to these buffers so that the sending processor can proceed with computing without having to wait for the completion of the send. The buffers are allocated dynamically.
- The output process is started once in the beginning the program and it spends all its time in a loop (in a separate subroutine) waiting for instructions and data from processor zero and other processors.
- The output processor works in the following way:
  1. get parameters from processor zero for opening a file
  2. collect data from all computational nodes and create a field to be written
  3. encode the field into GRIB using standard Hirlam procedures
  4. write the GRIB field into a file using standard Hirlam procedures
  5. if there are more fields to be written to this file, continue from point 2
  6. get parameters from processor zero for closing the file
  7. continue from point 1 to check if new file is requested to be opened
- Processor zero (or all computational tasks) can be stopped until the output processor has completed all the pending writing. This allows reading previously written files, which is needed during initialization.

The parallel output has been preliminarily implemented and the first runs have been successful. The next step is to check the results and to evaluate the performance.

### 3 Reproducibility

The question of reproducibility in parallel environment is: does the model produce exactly the same result when running with different number of processors. From the meteorological point of view the important aspect is: are the differences (if they exist) so large that they have of meteorological significance and do the differences grow in time so that at some stage the forecast depends too much on the number of processors.

It has been observed in several situations that the parallel Hirlam forecast model does not produce exactly the same result when running with different number of processors. However, it has not been documented, how large the differences are. To test this two Hirlam forecasts on Cray T3E system with 32 and 128 processors have been run and the results of the 48 hour forecasts have been compared. The test area was the FMI operational ATA area with 194\*140\*31 gridpoints. These tests were run with Eulerian advection scheme and the default Hirlam physics.

The surface pressure difference between the two experiments is shown in Figure 1. The maximum differences between the experiments are a few Pascals. So meteorologically they have of little importance. Also other fields both in the free atmosphere and at the surface were compared, and in all cases the differences were small. For instance, the maximum difference in 48 hour accumulated precipitation is less than one millimetre.

Repeating the same test without physics (adiabatic run) revealed that the cause for differences is in the physical parameterization and most probably in the moist part of it. This confirms that the parallelization itself works correctly.

Surface pressure difference (N32–N128) at +48h (Pa)

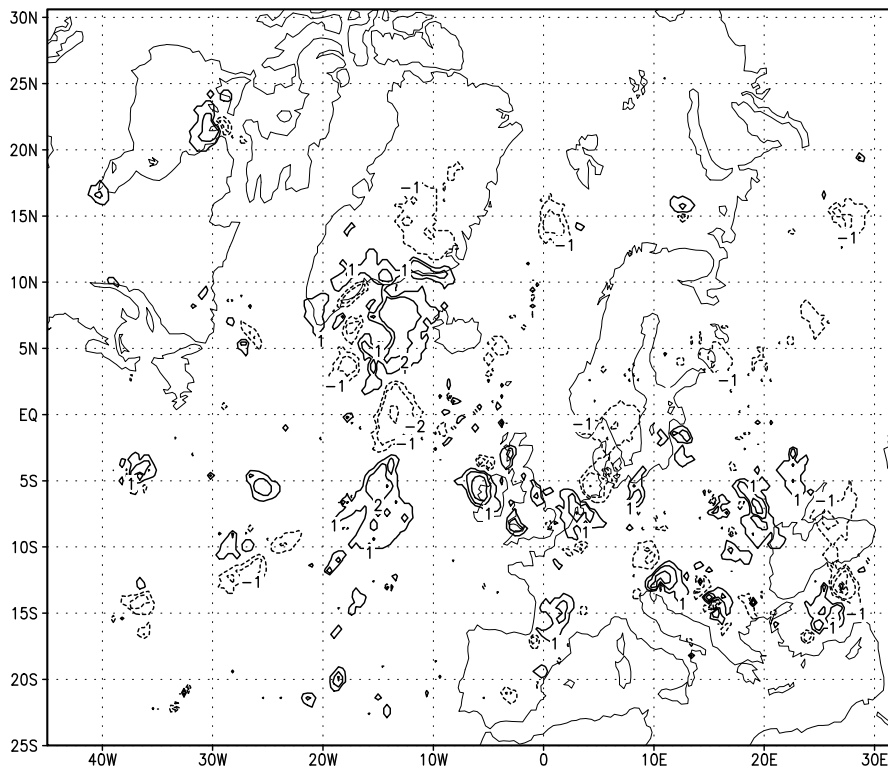


Figure 1: The difference in surface pressure from the experiments ran with 32 and 128 processors. Unit: Pa

## 4 Scaling of the forecast model

The parallelization strategy of the forecast model has originally developed to be used on the Cray T3D/T3E system using the SHMEM communication library [3]. Later the MPI interface has been added using the GC interface library [1]. It has been noticed that the scaling using MPI is not very good, especially when many processors are used. To see the possible bottlenecks some tests have been done both on T3E (using SHMEM) and Fujitsu (using MPI) with different number of processors. In these experiments semi-Lagrangian version of Hirlam 4.9.1 has been used with default physics on an area consisting  $366 * 260 * 31$  gridpoints. In this context it was also possible to compare the default direct Helmholtz solver to the iterative solver developed by Bjørge [2].

It is well known that Hirlam scales well on T3E. In this test the speedup when increasing the number of processors from 128 to 196 is 1.43, while the linear speedup would be 1.53. When comparing the timings of the direct and iterative solver there was not much difference with any number of processors.

On Fujitsu VPP700 at ECMWF it was much more difficult to measure the efficiency, because other jobs affected the timing and hence the performance varied very much from timestep to timestep. Therefore the times from the timestep taking the shortest time in a 48 hour forecast were taken to give some estimate of the ideal circumstances. Figure 2 shows the (best) times per timestep with different number of processors from 4 to 32. For comparison, also the ideal curves are shown. With small number of processors the scaling is good, but it decreases, when the number of processors increases: the speedup from 24 to 32 processors is only 1.05, while the ideal number would be 1.33, as shown with dashed line in Figure 2. If we look for possible reasons to the bad scaling in case of many processors, Figure 3 shows

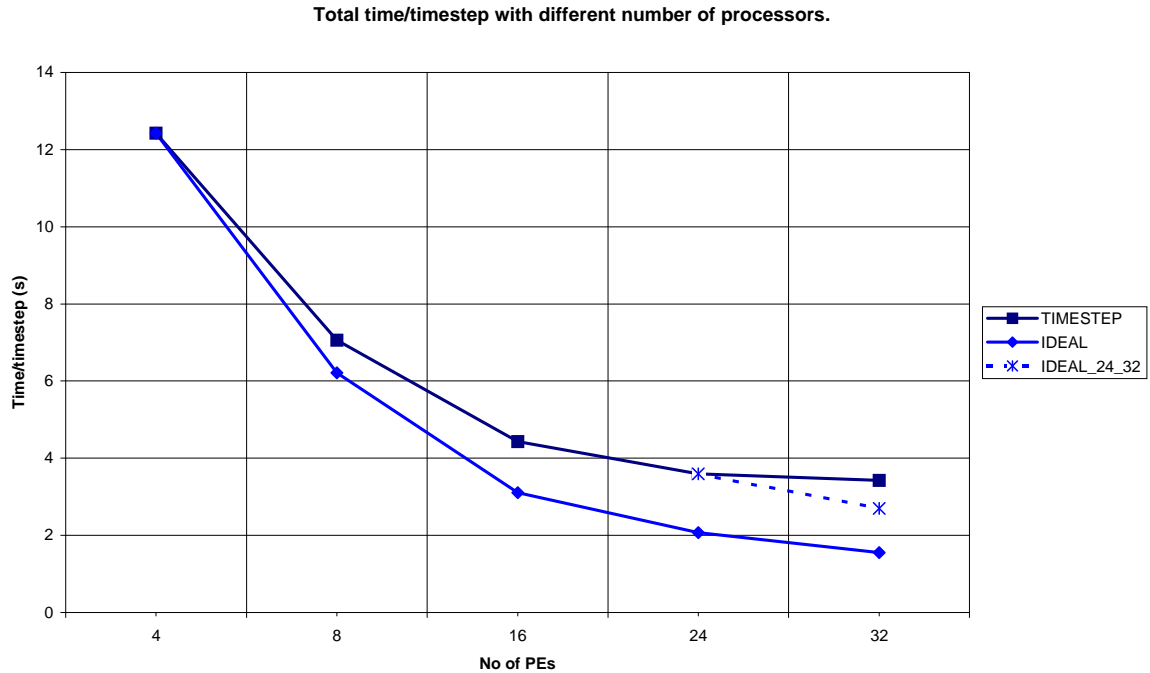


Figure 2: The best estimate of the total time of one timestep when running with different number of processors on Fujitsu VPP700.

the times of some subroutines with 16, 24 and 32 processors. The disappointing feature in this figure is the efficiency of the implicit diffusion: the time is increases when increasing the number of processors from 24 to 32 and with 32 processors it takes about 25% of the total computing time, while with four processors it takes only 10% of the total time. In the present implicit diffusion code there is, for every variable, several data transpositions, which certainly affects the bad scaling. So improving the implicit diffusion scheme certainly improves the overall efficiency. On the other hand, the physics seems to scale in a reasonable way. As on T3E, there was not much difference between the timings of the direct and iterative Helmholtz solver.

## 5 Concluding remarks

In this report we have described a method of parallelizing the output in such a way that it can be done simultaneously with computations. This method has been successfully implemented in the forecast model. The next step is to evaluate the performance.

The tests with different number of processors revealed that the forecast model does not exactly produce the same results with different number of processors. It also appeared that the cause for this is in the parameterization of the physical processes. However, the differences are so small that they do not have any meteorological significance.

Finally, it was found that the MPI version does not scale very well to very many processors. Although the tests were performed only on Fujitsu, a general feeling is that the same is true to other platforms using MPI, too. Especially the implicit diffusion seems to be very ineffective. The comparison between the direct and iterative Helmholtz solver did not reveal any advantages of one or the other. However, there may be some possibilities to improve the efficiency of the iterative solver.

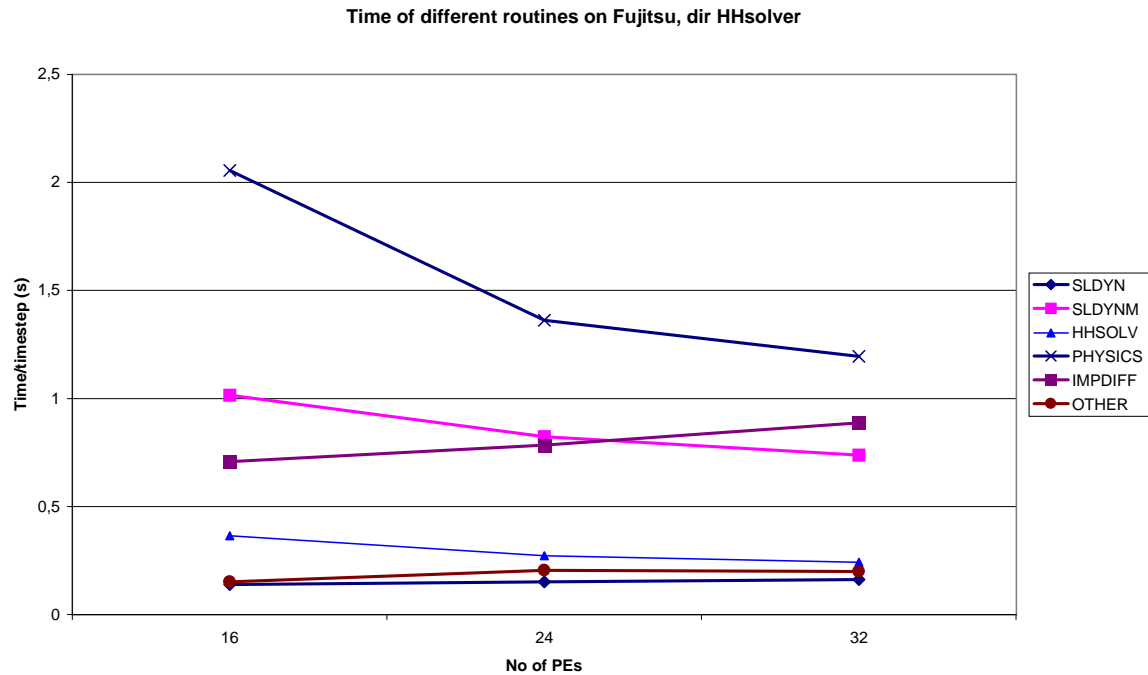


Figure 3: The best estimate of the timing of some routines when running with different number of processors on Fujitsu VPP700.

## References

- [1] J. Amundsen and R. Skålin. *GC - Generalized Communication*. SINTEF Applied Mathematics, 1996. Manual available at <http://www.sima.sintef.no/software/GC>.
- [2] D. Bjørge. Implementation of the semi-implicit scheme in a message passing version of hirlam. Technical Report 132, Det Norske Meteorologiske Institut, 1994.
- [3] K. Eerola, D. Salmond, N. Gustafsson, J. A. Garcia-Moya, P. Lönnberg, and S. Järvenoja. A parallel version of the HIRLAM forecast model: Strategy and results. In Hoffmann and Kreitz, editors, *Making Its Mark, Proceedings of the Seventh ECHWF Workshop on the Use of Parallel Processors in Meteorology*, pages 135–143. World Scientific, 1998.