Asynchronous communication in the HIRLAM weather forecast model

Van Thieu Vu[1] Gerard Cats[2] Lex Wolters[1]

[1] Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands

[2] Royal Netherlands Meteorological Institute, De Bilt, The Netherlands

vtvu@liacs.nl cats@knmi.nl llexx@liacs.nl

Keywords: weather forecast model, halo data swap, asynchronous communication.

Abstract

In this paper we investigate alternative asynchronous communication strategies in the HIRLAM weather forecast model. We perform experiments to evaluate the efficiency of each approach on DAS-3 and compare it to the original communication approach in the HIRLAM system.

1. Introduction

HIRLAM [4] stands for HIgh Resolution Limited Area Model and is a state-of-the-art analysis and forecast system for the numerical weather forecast. The weather forecast model is one of the three components in the HIRLAM system.

There are several parallel implementations of the weather forecast model. These have been developed by hand from the HIRLAM reference code. The undesirability of hand-code is that it creates difficulty in maintenance. Furthermore, making these implementations efficient on different type of computer architectures is an impossible task, because each computer system requires computer architecture-dependent optimizations. A code generator can assist these problems.

In [6] van Engelen proposed the CTADEL code generator, which can generate code for the so-called *dynamics* of the HIRLAM weather forecast model. In this paper we used this generation-code as a basis for all the experiments performed in Section 3. However, the modifications needed for the experiments were hand-coded. The aim of this paper thus is not to extend the generator, but to investigate whether it may be fruitful to do so.

In parallelization, the domain is horizontally decomposed into sub grids, where each processor contains the grid points of a sub grid. To compute these grid points we need information of grid points from adjacent processors. Exchanging data with those processors is needed. In the HIRLAM system, this is done by the so-called *halo data swap* procedure [5]. In this paper we introduce new approaches for this halo data swap in the weather forecast model.

Our experiments are run on DAS-3 [1], a wide-area distributed system in The Netherlands.

This paper is constructed as follows: Section 2 describes the data communication methods, including the original halo data swap procedure in the HIRLAM system and the new asynchronous communication approaches. In Section 3 we discuss our experiments on DAS-3 and performance of the various data communication methods in the weather forecast model. Section 4 is reserved for conclusion and discusses future work.

2. Asynchronous communication

In this section, we firstly describe the original halo data swap procedure in the HIRLAM system. Then, we introduce new asynchronous communication approaches for the data swap.

2.1. Halo data swap in the HIRLAM system

In the HIRLAM system the sub grid is divided into two parts (see Figure 1): the halo zone and the inner zone.

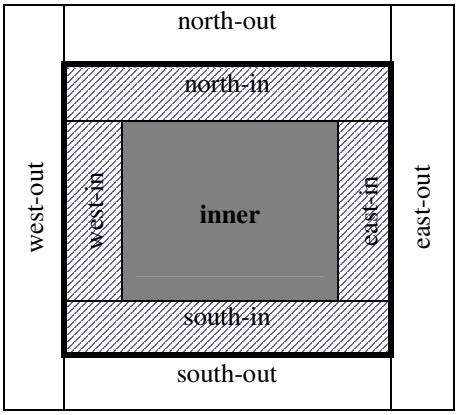

Figure 1: Sub grid of HIRLAM halo data swap. Gray area: inner zone; dashed area: inner halo zone; white area: outer halo zone.

The inner zone (gray area) is the un-swapped area, i.e. it can be computed without the need of the grid points from the neighbor processors.

The halo zone contains the grid points which are exchanged with the neighbor processors. It has two parts: the inner halo zone (dashed area), which consists of north-in, south-in, west-in, and east-in, contains the grid points which are *sent* to the neighbor processors; and the outer halo zone (white area) , which consists of north-out, south-out, west-out, and east-out, contains the grid points which are *received* from the neighbor processors.

In the HIRLAM system, all the grid points in a sub grid, except the grid points in the outer halo zone, are calculated first. Then, the data are exchanged. The swap process has two stages: first, the data are exchanged with the north and south processors. Once this exchange has been completed, the exchange is repeated with the west and the east processors.

The original halo data swap procedure in the HIRLAM system includes the following steps:

1) Do the computation in the whole sub grid;
2) Data in north-in and south-in are copied to the sending buffers;
3) Exchange data with its adjacent north and south processors;
4) Call *mpi_wait* to complete the north-south exchange;
5) Copy data from the receiving buffers to north-out and south-out;
6) Repeat steps 2, 3, 4, and 5 for the west-east exchange.

2.2. Asynchronization communication on data swap

In the previous subsection we have described the original halo data swap procedure in the HIRLAM weather forecast model. This procedure uses non-blocking MPI calls for data exchange. However, we observe that the exchange only starts when the computation is completed. Moreover, the data in the halo zone can be used only when the exchange is finished. This means that the communication can be seen as blocking MPI calls.

With non-blocking MPI calls [7], the computation can be done immediately after a MPI call, without waiting for the completion of the communication. Therefore, we may get an overlap between communication and computation. Our new approaches make use of this property of non-blocking MPI calls. The idea is to perform part of the computation while exchanging data. Firstly, we divide the computation into five parts: north-in, south-in, west-in, east-in, and inner zone (see Figure 1). Then, we overlap one of these computation parts with the exchange, with the requirement that the data have to be calculated before they are exchanged. With different ways of overlapping computation with communication we have different data swap methods, which are listed below:

Strategy-1:

The idea of the strategy-1 is to overlap the computation of the inner part with the west-east exchange and the computation of the west-in and east-in parts with the north-south exchange. In this way, we obtain an overlap between west-east communication and inner part computation as well as an overlap between north-south communication and west-east computation. The data swap is as follows:

1) Exchange data in the west and the east halo zones with their adjacent west and east processors;
2) Do the computation in the inner part;
3) Call *mpi_wait* to complete the west-east exchange;
4) Copy data from the receiving buffers to the west-out and the east-out halo zones;

5) Exchange data in the north and the south halo zones with their adjacent north and south processors;
6) Do the computation in the west-in and the east-in zones;
7) Copy data in west-in and east-in to the sending buffers;
8) Call *mpi_wait* to complete the north-south exchange;
9) Copy data from the receiving buffers to the north-out and the south-out zones;
10) Do the computation in the north-in and the south-in zones;
11) Copy data in the north-in and the south-in zones to the sending buffers.

Strategy-2:

In strategy-1, if the computation time of the west-in and east-in, which are small parts, is less than the north-south communication time then we do not make use of the overlap completely. In this case, overlapping the inner part with both west-east and north-south exchange processes is more efficient. To do this, we divide the computation of the inner part into two halves and overlap them with the west-east and north-south exchanges. This is the idea of strategy-2.

Now, the data swap is as follows:

1) Exchange data in the west and east halo zones with their adjacent west and east processors;
2) Do the computation in the first half of the inner part;
3) Call *mpi_wait* to complete the west-east exchange;
4) Copy data from the receiving buffers to the west-out and the east-out zones;
5) Exchange data in the north and south halo zones with their adjacent north and south processors;
6) Do the computation in the second half of the inner part;
7) Call *mpi_wait* to complete the north-south exchange;
8) Copy data from the receiving buffers to north-out and south-out;
9) Do the computation in the west-in and the east-in halo zones;
10) Copy data in the west-in and the east-in halo zones to the sending buffers;
11) Do the computation in the north-in and south-in halo zones;
12) Copy data in north-in and south-in to the sending buffers.

Strategy-3:

In strategy-1 and strategy-2 the west-in and the east-in parts are computed separately. These parts are short vectors. With the same total length, the computation of a set of short vectors can be more expensive than the computation of a set of long vectors. Therefore, in strategy-3, we compute the west-in, the inner, and the east-in parts at the same time while exchanging data with adjacent north-south processors.

The data swap is as follows:

1) Exchange data in the west and the east halo zones with their adjacent west and east processors;
2) Call *mpi_wait* to complete the west-east exchange;
3) Copy data from the receiving buffers to the west-out and east-out zones;
4) Exchange data in the north and the south halo zones with their adjacent north and south processors;
5) Do the computation in west-in, inner, and east-in part;
6) Copy data in the west-in and the east-in zones to the sending buffers;
7) Call *mpi_wait* to complete the north-south exchange;
8) Copy data from the receiving buffers to the north-out and the south-out zones;
9) Do the computation in north-in and south-in;
10) Copy data in the north-in and the south-in zones to the sending buffers.

3. Experiments

In this section we perform experiments for data communication on DAS-3 using the original HIRLAM halo data swap procedure as well as new approaches strategy-1, strategy-2, and strategy-3.

3.1. Environment

Our experiments are run on DAS-3 [2]. This is a wide-area distributed system which consists of five clusters located at five universities in The Netherlands: Vrije Universiteit (*fs0*), Leiden University (*fs1*), University of Amsterdam (*fs2*),

Delft University of Technology (*fs3*), and the MultimediaN Consortium (*fs4*). All our experiments are run on clusters *fs0* and *fs1*.

To create environment we use command *module* to load all default modules for interconnection. Then we use *mpif77* to compile the Fortran-MPI code and use *prun* to submit job to DAS-3.

We use various numbers of processors *Nx* x *Ny*, where *Nx* and *Ny* varying from 1 to 8 are the number of processors in x- and y-direction, respectively. The computation domain is 128x128x16. The size of halo zones is one grid point for north-, south-, west-, and east-direction.

3.2. Results

To investigate the efficiency of the communication methods, we measure the total execution time. This is shown in Table 1.

We observe that the new approaches slow down the weather forecast model, although there is an overlap between communication and computation; this is not what we expected. Table 1 shows that the elapsed time of the original HIRLAM halo data swap is the smallest, followed by that of strategy-3 and strategy-1; whereas it is largest in strategy-2. Furthermore, it is observed that this order reflects the number of computation domains in each approach, that is 1, 3, 5, and 6 in HIRLAM, strategy-3, strategy-1, and strategy-2, respectively.

To analyze this result, we split the elapsed time in communication and computation time. Table 2 shows the measured communication and computation time of the original HIRLAM halo data swap and new approaches.

The communication time is the time for exchanging data and the time for copying to and receiving from the buffers. The computation time is the total computation time of all parts in the sub grid. We see that it increases with the number of computation domains (i.e. 1, 3, 5, and 6 in HIRLAM, strategy-3, strategy-1, and strategy-2, respectively).

In the paragraphs below, we will figure out the explanation for this observation.

Observe that the arrays in the original HIRLAM method are long vectors, with the order of accessing array similar to the order of storing

Nx x *Ny*	HIRLAM	strategy 1	strategy 2	strategy 3
1x1	48.891	50.441	50.555	49.633
1x2	24.816	25.531	25.641	25.086
2x2	12.553	12.871	12.949	12.555
2x4	6.375	6.645	6.813	6.609
4x4	3.309	3.520	3.602	3.402
4x8	1.746	1.867	1.930	1.840
8x8	0.926	1.008	1.023	0.953

Table 1: Elapsed times (in seconds) of the data swap approaches.

	Nx x *Ny*	HIRLAM	strategy 1	strategy 2	strategy 3
Communication	1x1	0	0	0	0
	1x2	0.191	0.207	1.117	0.227
	2x2	0.152	0.117	0.098	0.117
	2x4	0.137	0.137	0.082	0.082
	4x4	0.105	0.109	0.074	0.082
	4x8	0.125	0.051	0.070	0.059
	8x8	0.125	0.031	0.051	0.066
Computation	1x1	48.891	50.441	50.555	49.633
	1x2	24.625	25.406	25.508	24.957
	2x2	12.406	12.797	12.879	12.449
	2x4	6.238	6.605	6.770	6.535
	4x4	3.203	3.461	3.574	3.340
	4x8	1.621	1.848	1.895	1.805
	8x8	0.844	0.996	0.992	0.902

Table 2: Communication and computation times (in seconds) of data swap approaches.

Nx x *Ny*	north-in, south-in	west-in, east-in
2x2	0.680	0.715
4x4	0.332	0.352

Table 3: Computation time (in seconds) of north-south and west-east parts in strategy-2.

array in memory. In our new approaches, the computation is divided into parts. In some part, e.g. west-in and east-in, the computation is done with different order of accessing and storing array. The arrays used in the calculations are short vectors, the computation of which may be more expensive than the computation of long vector. To verify this, we make the following comparisons.

Nx x *Ny*	HIRLAM	strategy 1	strategy 2	strategy 3
1x1	0	0	0	0
1x2	0	0.082	0.984	0.098
2x2	0	0.043	0.027	0.012
2x4	0	0.098	0.039	0.008
4x4	0	0.051	0.047	0.020
4x8	0	0.031	0.036	0.023
8x8	0	0.020	0.020	0.016

Table 4: Overlap of asynchronous communication, is obtained by subtracting the execution time from the total of communication and computation time.

Firstly, in Table 3 we show the calculation time of parts in strategy-2 separately. We find that the computation time of west-in and east-in is larger than the computation time of north-in and south-in, although the number of grid points in west-in and east-in is smaller than the number of grid points in north-in and south-in (see Figure 1). This is also true for strategy-1, verifying our conclusion for the individual strategies.

Next, we compare different tests which use different types of arrays. We know that in strategy-3 and in HIRLAM, the calculation arrays are long vectors in all parts, meaning consecutive memory locations are accessed, resulting in a good performance. As we can see from Table 2, the computation times in strategy-3 and in HIRLAM are smaller than that in strategy-1 and strategy-2.

We conclude that the difference between the computation time in the original HIRLAM method and in the new approaches is due to the different ways of accessing data.

As it was mentioned, our idea was to make use of the overlap to speed up the execution. However, as we have seen in Table 2, the result was not what we expected. What could be the reason?

This is because the communication time is very small compared to the computation time. The overlap hence does not affect significantly the speed up of new asynchronous communication approaches. This is indeed shown in Table 4.

In summary, the above result can be explained by the fact that the overlap that we gain is smaller than the increase of the computation time due to dividing the domain into parts.

4. Conclusion and future work

We have introduced the new asynchronous communication strategies for the halo data swap procedure in the HIRLAM weather forecast model. Some experiments of alternative communication methods on DAS-3 have been shown. It turned out that the new asynchronous communication approaches slow down the weather forecast model. This is because the overlap that we gain from communication and computation is smaller than the increase of the computation time due to dividing the domain into parts.

In our experiments, the size of the halo zones is one grid point for north-, south-, west-, and east-direction. In future, we will study larger halo zones, as e.g. required in the semi-Lagrangian formulation of the model. Larger halo zone requires longer time for communication, hence returns larger overlap. Thus, larger savings by the asynchronous communication method are expected. Currently, we investigate the implementation of our model on more than one cluster of the DAS-3 computer, which serves as a prototype for a (wide-area) Grid computer implementation. We expect that communication times will become much larger than in the current one cluster implementation and hence, we expect to gain from asynchronous communication here as well.

Dependent on the outcome of these experiments we may decide to extend the Ctadel code generator, to the effect that it will be able to generate asynchronous communication code. In our current set-up, each sub-domain is processed by a separate subroutine call. If our further investigations warrant an extension of Ctadel, we will also modify Ctadel such that it will generate inline codes. Although the savings will be small, they may be big enough to make asynchronous communication interesting, even in the one-row halo zone, single cluster case that we investigated so far.

REFERENCES

[1] DAS-3: http://www.cs.vu.nl/das3.

[2] Execution job on DAS-3: http://www.cs.vu.nl/das3/jobexec.shtml.

[3] Haobo Zhou, Lex Wolters, and Gerard Cats, *Evaluation of the HIRLAM*

Forecast Module on the Grid Environment DAS-2, http://www.hirlam.org/open/publications/NewsLetters/48/72_GCa_rep.pdf.

[4] HIRLAM project: http://hirlam.org/.

[5] Reference HIRLAM Scalability Optimization Proposal – revision 1.2: http://www.hirlam.org/open/publications/NewsLetters/44/HIRLAMOptNewsletter.pdf.

[6] Robert A. van Engelen, *Ctadel: A Generator of Efficient Numerical Codes*, PhD thesis, Universiteit Leiden, 1998.

[7] The Message Passing Interface (MPI) standard: http://www-unix.mcs.anl.gov/mpi/.