Overlapping communications with calculations

Gerard Cats
Royal Netherlands Meteorological Institute (KNMI)
cats@knmi.nl
Van Thieu Vu and Lex Wolters
Leiden Institute of Advanced Computer Science (LIACS), Leiden University

The communication overhead in a parallel implementation of a numerical weather prediction model may possibly partly be hidden behind calculation time. We tried this in a proof-of-concept study with the HIRLAM Eulerian explicit grid point dynamics. We found that savings in total execution time depend on the available hardware. If the network is fast, the additional calculation costs are not compensated by the reduction in communication overhead. But if the interconnection network is not very fast, or on a Grid computer, we are able to almost completely hide communication time. We anticipate several problems before our results can be implemented in an operational configuration of HIRLAM, not the least of them being the code complexity, in particular in a semi-Lagrangian scheme.

1 Introduction

The HIRLAM forecast model code has been designed to run on a massively parallel machine. Parallelization is by straightforward rectangular domain decomposition. A time step consists of a communication step, in which border data are fetched from the neighbouring processors, followed by a calculation step over the domain assigned to a processor.

In this paper we report on experiments to start the calculations while the communications are still in progress. The aim is to reduce overall execution time by hiding the communications behind the calculations. This obvious advantage of overlapping communications with calculations has a counterweight in increased calculation time and, perhaps more importantly, in increased code complexity. In particular the latter consideration made us decide to start with a proof-of-concept study based on the simplest HIRLAM configuration, i.e. the Eulerian explicit grid point dynamics. We are aware that our proof-of-concept will still be far from implementation within an operational context, even if we prove the concept.

One of the current challenges in ICT is found in Grid computing (Foster and Kesselman, 2004). The large communication costs incurred on a Grid inhibit the implementation of HIRLAM on a Grid. Perhaps, hiding the communication behind calculations will remove this inhibition. That is why we performed our experiments not only on a cluster, but also on a two-cluster configuration, as a prototype of a Grid computer.

In Section 2 of this paper we will explain the concept of overlapping in quite some detail. We do so because changing the order of calculations and communications is a sensitive and error-prone operation. This also motivates several decisions not to aim at a full dataflow programming model (see Wikipedia dataflow), but rather to avoid tight entanglement of communications and calculations to keep the coding simple.

We anticipate that overlapping will need additional calculation time, and that the balance between gains and losses may depend on the computer hardware. Therefore, in Section 3 we describe a range of strategies for overlapping, ranging from one that may be the fastest on vector type machines, to one that aims at maximum overlap, regardless the additional computational costs of overlapping. In Section 4 we describe our experiment configuration. In Section 5 we give the results. The results give rise to an extensive discussion, in Section 6, both on the results and on the problems that we anticipate to be encountered when extending our proof-of-concept study to an operational implementation. Section 7 gives the conclusions.

2 The overlap concept

Parallelization in the explicit dynamics part of the time step of HIRLAM is achieved by domain decomposition. Each processor performs the calculations for a rectangular domain. In the left panel of Fig. 1 this domain is indicated as “the processor domain”. The processor domains collectively form the HIRLAM domain. For the gradient and advection terms in the equations, information is needed from adjoining regions of the domains of neighbouring processors. Those regions form the “halo” around the processor domain. To collect the halo information there are 4 communication steps: first for the West and East neighbours, then for the North and South neighbours (or the other way around). The four corners of the halo come from four non-adjacent processors, indirectly: first the data are communicated to the West and East adjacent processors, and then to the North and South adjacent processors. Hence, the North and South halos in the left panel of Fig.1 include those corner areas. This indirect exchange will only work properly if the West and East communications are complete before the North and South ones start. Later on we will see that this requirement has implications for our study.

In HIRLAM, the information from the halo is collected before the processor starts its calculations. However, that information is needed only for calculations close to the boundaries of the processor domain. By “folding” the halo inward, we divide the processor domain into 5 subdomains: *M, W, E, N* and *S*. The right panel in Fig. 1 demonstrates this. The calculations in the middle domain M are independent of the information in the halo, and thus can be done while the information from the halo is being collected. This is the general concept of overlapping communications with calculations: while communicating data, already perform the calculations that do not depend on the communicated data. After communications have completed, the remaining calculations, which do depend on those data, are completed.

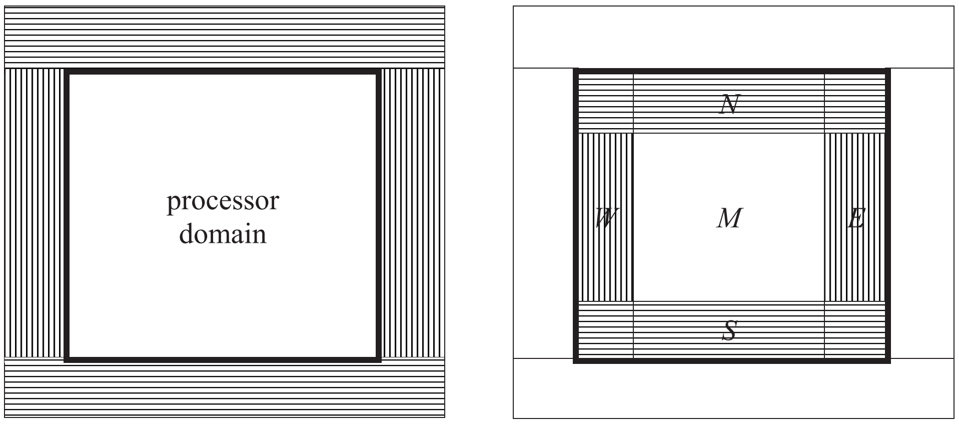

Figure 1: A processor domain and its halo (left); and the processor subdo mains (right) in case the W⇔E communication precedes the N⇔S one.

Overlapping has the obvious advantage of saving elapsed time by hiding communications behind calculations. There are disadvantages as well, though. We mention:

1 *More complicated coding.* The calculations over the processor domain cannot be executed anymore with straightforward double loops in the two horizontal directions. Instead, the calculations have to be executed separately for a series of subdomains. So the calculation code becomes more complicated. Furthermore, the communication code has to be interspersed with the calculation code, which also implies more complicated coding.
2. *More time spent in calculations.* The separate execution of loops over subdomains will probably result in longer calculation times. Furthermore, the West and East subdomains are very narrow. The loops over longitudes when doing the calculations for those subdomains become very short. These are the inner of the two horizontal loops. On vector machines short inner loops cost relatively much in calculation times. But on cache-based machines short inner loops may be relatively expensive as well. Also, the North and South subdomains are very slim, implying short outer loops, but probably, this is less harmful for performance than the short inner loops on the West and East subdomains.

A priori, it is not obvious that the time saved by overlapping communications with calculations will exceed the calculation time lost due to less efficient execution of the shorter loops and the larger number of subdomains. And even if it does, it still has to be judged whether the advantages outweigh the disadvantage of more complicated coding.

One of the considerations determining whether it pays to try to overlap communications with calculations is the size of subdomain *M* as compared to the total domain. If the halo width is more than half the processor domain size, the subdomain *M* will be empty, and there will be nothing much to overlap. In typical current implementations, a processor domain contains some 80 by 60 grid points, and the halo width for semi-Lagrangian is typically 10 grid points. Hence, subdomain *M* has typically 60 by 40 grid points, which means that in a semi-Lagrangian model about half the calculation time is available to hide communications behind. For this proof-of-concept study we used the Eulerian formulation of HIRLAM. The halo width is then 1 or 2 grid points, which implies that almost all calculation time is available for that hiding. On the other hand, we did our proof-of-concept experiments with a small HIRLAM domain, with in the most unfavourable case (120 by 120 HIRLAM grid points on 64 processors) only 15 by 15 grid points for a processor domain, and, with the halo width of 2 grid points, 11 by 11 points in *M*. So also in this proof-of-concept study subdomain *M* is about half the size of the processor domain.

In principle, calculations over subdomain (for example) *W*, where we exclude the 2 overlapping areas with *N* and *S* in the corners, can start as soon as the processor has received the halo data from the West neighbouring processor. Hence, it is conceivable that each processor starts sending data to the East neighbour, then starts receiving data from the West neighbour, - all this communication in parallel to calculations over *M* - , and then calculates over *W*. A similar sequence of events can also be coded for domains *E*, *N* and *S*, where *N* and *S* include the 4 corner areas. However, to reduce code complexity, we required the communications both from the West and from the East to be complete before the calculations over either subdomain *W* or *E* were allowed to start; and similarly for the North and South communications. Hence, we define two communication steps, W⇔E and N⇔S. For proper communication of the corner areas, the first of these two communication steps must be complete before the second starts; and the 4 calculation subdomains *W, E, N* and *S* must be defined such that the four corner areas are part of the subdomains that are communicated in the second step, to make sure that their calculations do not start before both communication steps have completed. So if the order is: first W⇔E, then N⇔S, then for calculation purposes the corners will be included in the *N* and *S* subdomains. For data exchange purposes, the four corner areas are always part of both subdomains they are in.

The width of the halo zones on the North and South are not necessarily the same, because HIRLAM uses a staggered grid: the v-component of the wind is kept at grid points that are shifted to the North by half a grid distance. Optimally profiting from this effect would again complicate code. Hence, we simply assume that all halo zones have the same width, namely 2 grid points, which is the minimax required for correct execution.

A proper dataflow machine would avoid the simplifications described in the two preceding paragraphs (see Wikipedia dataflow).

3 The overlap strategies

We investigated several strategies to overlap communications to calculations. Table 1 gives an overview. In the Table, communication is indicated by an arrow, and calculation subdomains are indicated by italics. Sequential execution is indicated by the table rows, top-down. Overlap of communications and computations is indicated by a communication entry and computations in the same row. Of course, a computation subdomain must only be processed after its communications have completed. In order to communicate the domain corners properly, a communication in one direction must have completed before the other one is allowed to start. The strategies, to be described below, are ordered by the number of subdomains for calculations. Hence, the order does not imply any general preference by us.

Table 1: Investigated overlap strategies

strategy	structure	
1s (HIRLAM) simple code	N↔S	
	W↔E	
	NWMES	
3s long vectors	W↔E	
	N↔S	*WME*
	N	
	S	
5s undivided *M*	W↔E	*M*
	N↔S	*W*
		E
	N	
	S	
6s both overlap *M*	W↔E	M/2
	N↔S	M/2
	W	
	E	
	N	
	S	

The first strategy is the one currently used in HIRLAM, *i.e.* no overlap at all. First, all data are communicated, and then computations are done over the entire domain. This strategy is expected to have the lowest calculation time, but its main advantage is its code simplicity.

The second strategy is a design that may be profitable if short inner loops are very expensive to execute. This strategy achieves long inner loops in all computations, at the penalty of not overlapping the W↔E exchange. It results in overlap of the N↔S communications only.

In the third strategy we accept short vectors when processing the West and East subdomains. We assume that those computations are slower than those for the North and South subdomains, because the latter have long vectors. Thus, more calculation time will be available for overlap if first W↔E is communicated, than if the N↔S communications would have been the first. Hence, the W↔E communications are done first, overlapped with the computations for the *M* subdomain. At the end of the next paragraph we will explain the possible advantage of this strategy.

In the fourth strategy, we try to overlap both communication steps with the calculations over the middle domain. The combination of Fortran and MPI does not simply permit to synchronize communications while calculating. This synchronization is required to ensure that the communications in one direction have completed before those in the other direction start. We resolved this by splitting the *M* subdomain into two equal parts. So halfway the progress through the *M* subdomain, we switch the program execution to MPI routines to wait for the W⇔E communications, and start the N⇔S ones. Then we continue with the remainder of the *M* subdomain. So in the strategy, we have to accept an additional calculation overhead by splitting the middle subdomain into two parts. If that additional overhead does not balance the advantage of increased overlap, the third strategy, described in the previous paragraph, is preferable.

In the Table we indicated that the calculations over non-contiguous domains are done sequentially, but of course the order in which we process the subdomains will be almost immaterial.

It so happens that the four strategies can be distinguished by the number of subdomains for computations. Hence, we code the strategies by that number followed by the letter ‘s’. The strategy 1s has 1 subdomain; it is the current HIRLAM strategy. Strategy 6s has 6 subdomains: the 5 as shown in the right panel of Figure 1 plus one because we had to split the middle subdomain.

4 Experiment configuration

4.1 Hardware

The experiments were conducted on the DAS3 (Seinstra and Verstoep, 2007). The DAS3 machine is built around AMD Opteron processors. Single cluster experiments were performed on the Vrije Universiteit (VU) cluster, which has 85 dual compute nodes, so 170 processors in total, each dual core. The clock frequency of each processor is 2.4 GHz. Our experiments on two clusters involved the VU cluster and the Leiden University (LU) cluster. The latter has 32 dual nodes single core processors, with a clock frequency of 2.6 GHz. We always use these clusters in single node, single core mode. The physical distance between the two clusters is approximately 40 km. A two-cluster configuration is a proto-type for a Grid computer (Foster and Kesselman, 2004).

Within a cluster, the nodes are connected with a Myri-10G (Myricom, 2006) switch and with 1Gbps Ethernet. Communication between clusters can be over common internet, to which the clusters are connected with 10Gbps Ethernet. The Myrinet hardware can also be used for communications between clusters, but then it does not support the Myrinet express protocol MX. Instead, the protocol available for inter-cluster communications over the Myrinet cards is TCP. Between the clusters there are 8 lines at 10 Gbps each, shared with other users of the clusters.

4.2 Software

From HIRLAM, we used the DYN routine. It codes the grid point, explicit Eulerian primitive equations solver. However, we chose to do our experiments with the code generated with Ctadel (Van Engelen et al., 1997) rather than the original hand coded routine. We made this choice for the following reasons:

- *Flexible area selection.*
  The hand code assumes the South West corner of the domain to be the (1,1) point. This is awkward in case the DYN routine has to be applied on different subdomains. The code generator was straightforwardly adapted to generate code for which the South West corner can be adjusted.
- *The generated code is correct.*
  This is not a very strong argument, as the hand code is sufficiently correct for our purposes.
- *Future: inlining, loop unrolling.*
  The DYN routine is to be invoked over a number of subdomains. If subroutine calling overhead

turns out to be significant, we may want to decide to inline the series of DYN invocations to eliminate that overhead. The code generator can be modified to do so. The West and East domains have very short loops in the longitude direction. In the hand code, these loops are the inner loops. Probably, a lot of loop overhead can be avoided by unrolling the inner loops. Again, the code generator can be modified to do so.

- *In general, the generated code is faster;* hence, communication overhead is more significant. Because our research aims at optimizing performance, we should not start with sub-optimal codes.

Even though we motivated our choice for the generated code to a large extent on the possibility to extend the code generator to inline subroutine calls and to unroll loops, for this proof-of-concept study we did not attempt to do so. We coded the invocation of DYN for each subdomain from a main routine by hand. So we invoke the generated DYN code from a hand-coded main routine.

Our main routine also organizes the communications by invocation of the MPI non-blocking communication routines MPI_ISEND and MPI_IRECV. Depending on the strategy, the order of invocation of these routines and DYN ensures that the communication is parallel or sequential to the calculation. Synchronization is by MPI_WAITALL.

As compiler we used mpif77. The communication routines for our single cluster runs were from MPICH, and for the multi-cluster run from OpenMPI, because MPICH turned out not to work well between clusters. The performance difference between the two MPI implementations is small, but needs elimination in a follow-up study.

Timings were taken with the MPI wall clock routine MPI_WTIME.

4.3 Buffer size, domain size

If the size of the messages exceeds a certain system buffer size, the non-blocking MPI routines should switch to blocking mode, thus inhibiting overlap of communications with calculations. In this proof-of-concept study, we did not investigate whether the limiting system buffer size can be configured during the installation of MPI, or whether the buffered non-blocking MPI routines (MPI_IBSEND) can do without that system buffer. Instead, we simply circumvented the limitation by ensuring that our messages are smaller than the system buffer, which, in our case, was 64 kB. We reached this small message size by using a fairly small domain: 120 by 120 points in the horizontal, and 16 levels; and by using a sufficiently large number of processors: for this domain we need at least 16, in a 4 by 4 configuration. In the context of limiting message size, we profited from the fact that DYN does not need a halo that is wider than 2 points.

4.4 Reproducing results

In parallelization studies it is always crucial to avoid the usage of data from other processors before they have been communicated. We verified the correctness of our different strategies, and the correct implementation in our experiments, by checking that the tendencies, calculated with DYN, were bitwise identical to those obtained with a single processor run over the total domain. Hence, the meteorological results of our experiments are reproducible. Because of the different order of calculations, the Ctadel generated code gives results that differ slightly from the ones with the hand code, but that is not relevant for this study.

However, the main results of this study are not the tendencies, but the elapsed times it takes to obtain them. To get an impression of the reproducibility of the timings, we ran each experiment 5 times. It turns out that the pure calculation times are very well reproducible: they show variations of the order of 1%. This low variation is a consequence of the processor allocation strategy on DAS3: there is no time sharing. The communication times, however, were found to vary by much more. On a single cluster, variations by 50% were observed, and on two clusters even by a factor of 2. These variations arise because the networks were not dedicated to our experiments. Some variation within a cluster arises because the topology of the allocated processors is not always optimally matching the domain decomposition. The timings that we are going to report in this study are the medians out of each set of 5 reproducibility runs.

5 Results

5.1 Scaling of calculation times

The calculation overhead needed to hide communications behind calculations does not scale with the number of processors. Our first investigation therefore looks into the scalability of the calculation time, without looking at communication times yet. We ran the 120 by 120 domain on 4, 16, 36 and 64 processors, always in a square configuration, on a single cluster. The calculation times are shown in Fig. 2. HIRLAM (1s) scales almost linearly upward from the 4 processor run. Scalability decreases with the number of subdomains, which is consistent with the increased number of subroutine calls. Ethernet requires more CP activity than Myrinet, and this also decreases scalability.

The line “lin” is the linear extrapolation of the HIRLAM (hence 1s) data point at 4 processors. On 4 processors, the other strategies are less than 5% slower than 1s, and Ethernet is about 1% slower than Myrinet. At more processors, these differences grow, to almost 20% at 64 processors. This 20% reduction in scalability by overlapping communications does not deter us much, in particular because the number of processors is not normally increased to complete a certain problem faster, but to increase the problem size, i.e., the number of grid points.

5.2 Overlap on a single cluster

Fig. 3 shows the overlap between communication and calculation. The HIRLAM strategy (1s) does not have any overlap. On 1 Gbps Ethernet, the strategies 3s, 5s and 6s have at least one of the

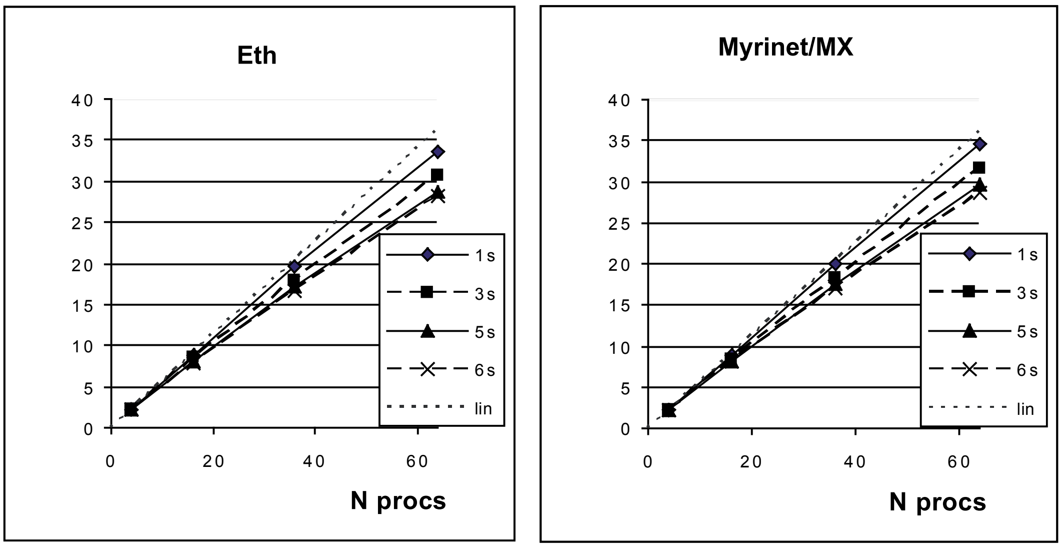

Figure 2: Inverse calculation times as a function of the number of processors, for the 4 strategies. The dotted line follows the linear speed up from the 4 processor, 1s measurements. Left for Ethernet, right for Myrinet with the Myrinet MX protocol

communication phases overlapped with calculation. This is seen in the figure by the much reduced time spent on that phase. To be precise, the time is still there, but it is hidden behind the calculation time.

We observe that communication is hidden behind the calculations, to a large extent, as far as the strategies permit: In 1s there is no overlap; in 3s the N⇔S communications are overlapped; in 5s the W⇔E exchange is overlapped, but the calculation time for the *W* and *E* domains is far too small to hide the N⇔S communications; in 6s, communication times almost disappear.

The increase in calculation time with the number of subdomains is clearly demonstrated by the figures. The calculation time is almost linear in number of subdomains; *e.g.,* with Myrinet, on 4 by 4 processors, the correlation between calculation time and number of subdomains is higher than 0.999. This, by the way, confirms the reproducibility of the calculation times. Communication times are less reproducible, as patently demonstrated by the W⇔E communication times on 8 by 8 processors with Ethernet: there is no overlap in either of the strategies 1s and 3s, and on a dedicated network the timings should have been similar in both strategies.

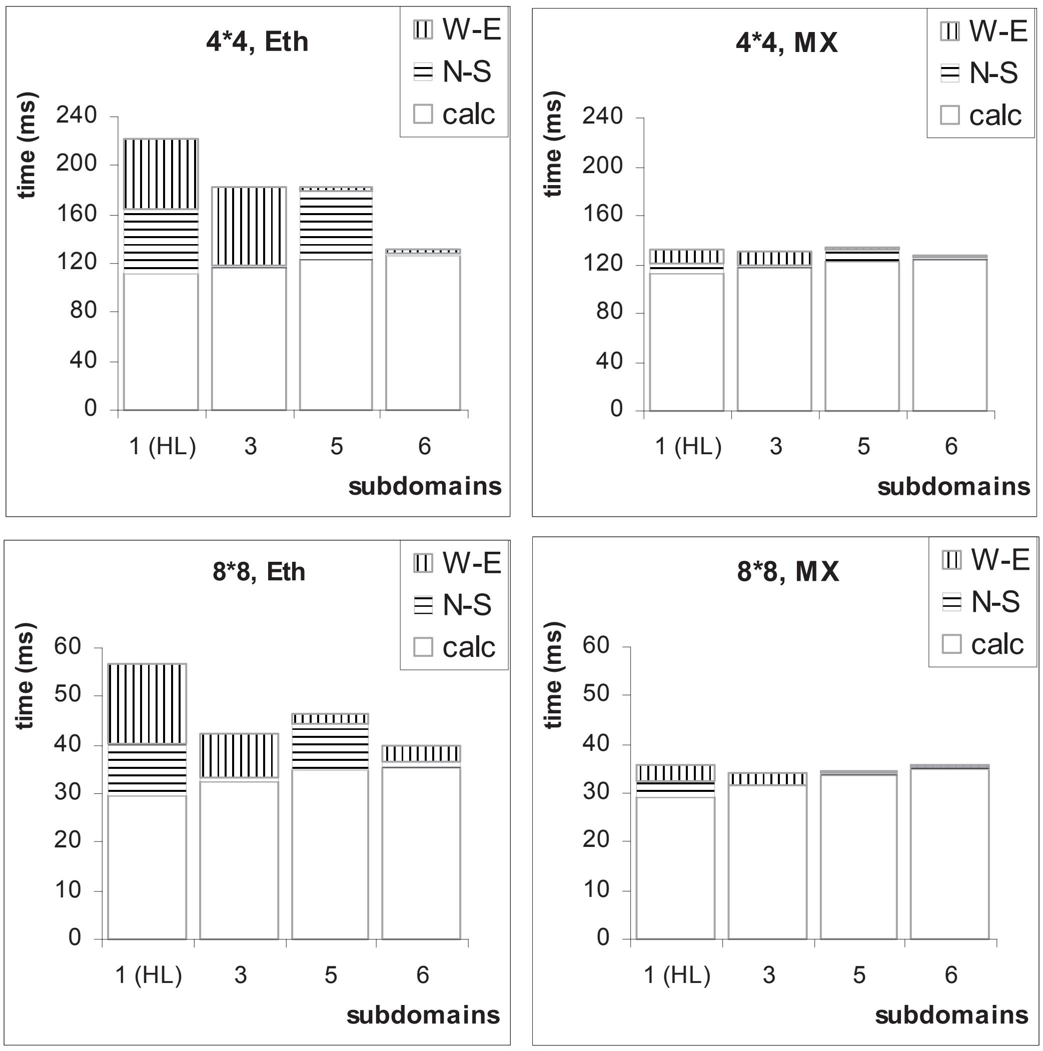

Figure 3: Calculation time, and non-overlapping parts of the communication times, for 16 (top) and 64 (bottom panels) processors. Left panels for Ethernet, right for Myrinet with the MX protocol. Single cluster results.Timings are for the first time step, but later steps are similar.

With Ethernet, the communication is so slow that each strategy to overlap communications to calculations is profitable. With Myrinet, on the other hand, the decreased communication time hardly compensates, if at all, the increased calculation time, simply because the communication times are very low. Ethernet also requires some central processor activity, which shows up as non-overlapped communication time; particularly so in the 8 by 8 processor case, where that activity is relatively more expensive than in the 4 by 4 case.

5.3 Overlap on two clusters

On two clusters, the picture changes on some points. We implemented the system on two clusters by assigning the Northern half of the total HIRLAM domain to one cluster, and the Southern half to the other. Fig. 4 shows the timings on the LU cluster, which has faster processors. Fig. 2 and 3, we bring into recollection, are the timings on a single cluster, hence at VU, with the slower processors.

The first point to note is that the calculation times have decreased by approximately 8%. This is a consequence of our choice to present the timings on the cluster with the faster processors. On the

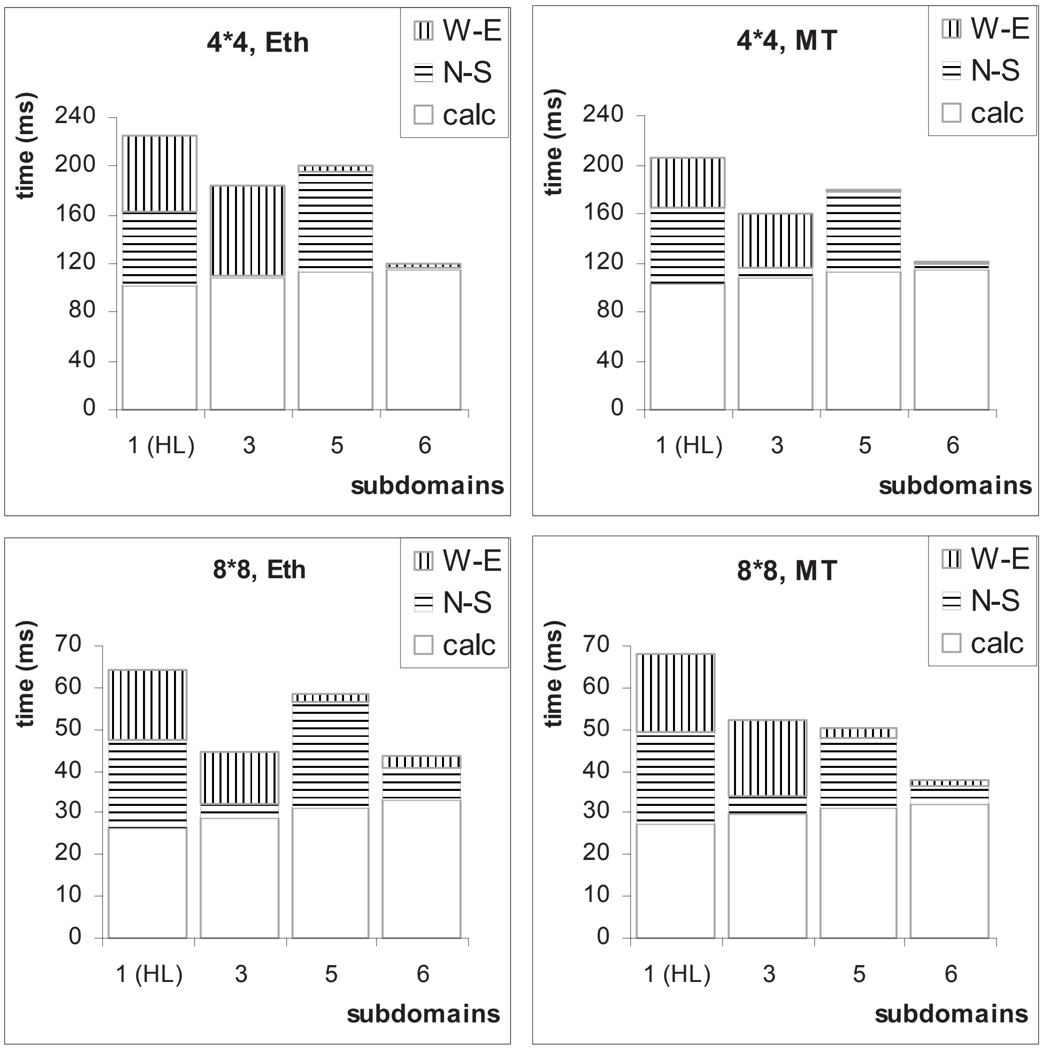

Figure 4: As Fig.3, for the two-cluster runs. The right panels now show the Myrinet over TCP protocol. Timings are taken on the cluster with the faster processors. Timings on the slower processor are the same, except that calculation times are approximately 8% larger. This gives additional N↔S communication time in later time steps, but the figure does not include that

other cluster, the calculation times are similar to those in Fig. 3. However, that cluster delays the faster cluster during communications. Indeed, on the faster (LU) cluster the N⇔S communication is less overlapped than on the slower (VU) cluster, the difference in overlap being equal in size, opposite in sign, to the difference in calculation times.

The second point we note is that Myrinet with the TCP protocol is not consistently faster than Ethernet, even though Ethernet is only over common internet.

The third point we mention is that the N⇔S communication takes more time than on a single cluster, even after correction for the delays caused by the slower processors on the other cluster. However, given the large physical distance between the two clusters (40 km) the additional delays are relatively small. Yet, by using two clusters, the total communication time has grown bigger than calculation time, in particular on 64 processors. Hence, it cannot be hidden fully anymore, not even in strategy 6s.

Discussion

If a fast interconnect is available, with straightforward domain decomposition nearest neighbour communication is fast. It will then be difficult to reduce total execution time by overlapping communications with calculations, because even if the little time needed for communication can be eliminated, it is doubtful whether the gain balances the additional calculation time. Knowing this, we started this investigation with designing a range of strategies to play with the balance of gaining by overlap and losing by additional calculations. Our experiments with the Myrinet MX protocol confirmed this: Little time is spent in communications if not overlapped at all (strategy 1s), and minute savings in total execution times are possible. On 64 processors, we found that strategy 3s is the fastest, but in total, when using MX, the gains do really not warrant the code complications needed for overlap.

On the other hand, if the interconnect is not very fast, considerable savings are possible by overlapping. On our hardware, the additional computational costs of processing several subdomains, some with very short vectors, are limited. All our experiments then show that strategy 6s is preferable; on two clusters we found even that calculations are too fast to enable full overlap. After inspection of 6s, we realized that a refinement should be possible, to provide more calculation time to hide the N⇔S communication behind. Table 2 and Fig. 5 illustrate this refinement: The middle subdomain *M* is divided into two parts, not necessarily equal in size, and the calculations over *W* and *E* are also done parallel to the N⇔S communication. To maximize vector length, we join the second

Table 2: A refinement of 6s

6s(α)	W⇔E	α *M*
	N⇔S	α *W*
		α *E*
		(1-α) *WME*
	N	
	S	

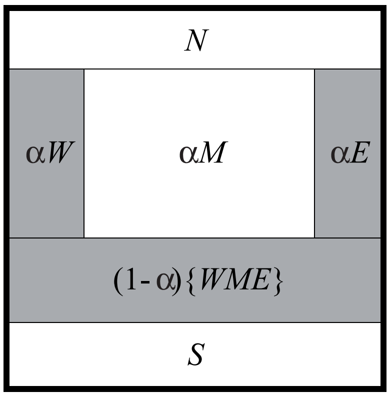

Figure 5: Strategy 6s(α). The N⇔S communications overlap the calculations in the grey area.

part of *M* and the matching parts of *W* and *E*. Parameter α can be tuned to achieve maximum overlap. We introduce the notation 6s(α) for this strategy, and note that 6s(0) and 6s(1) become our old strategies 3s and 5s, resp., after elimination of the redundant code for the empty subdomains. On our hardware the calculations over *W* and *E* do not take much time; hence 6s(1/2) is not much better than 6s; we found an optimum at $\alpha = 0.3$, saving 2% of total execution time over 6s.

Once we have overlap, we observe that there is not much advantage of having a state-of-the-art interconnect. With a relatively cheap 1Gb Ethernet total execution times are comparable to those obtained with Myrinet/MX.

Also, we observed not more than slight performance decay when we went to two clusters, even if the two clusters are only connected over the internet. Part of the performance loss on our two clusters is due to the difference in processor speeds, of approximately 8%. When we used 16 processors, we assigned a domain of 30 by 30 grid points to each processor. Measurable improvements are possible by a redistribution, to let the faster processors calculate 31, and the slower processors 29 lines of latitudes. The speed difference is then almost fully compensated, and HIRLAM needs 3% less of execution time. On 64 processors, it is hardly worth assigning 14 lines of latitudes to the slower, and 16 to the faster processors, because then the faster processors get too heavy a load: this distribution overcompensates the speed difference. We note that the speed difference can be made useful by increasing the HIRLAM domain size from 120 by 120 to 120 by 124, where the 4 additional lines of latitudes are assigned one to each of the faster processors. In this way we get an increase of domain size for free.

The calculations over the *W* and *E* subdomains have inner loops of length 2. This calls for elimination of those loops, by writing them out. Probably a further optimization will then be reached by merging the codes for the *W* and *E* subdomains. Actions of this kind are best performed by the Ctadel code generator. On our hardware we will not profit much from these modifications, because the additional computation time needed to achieve overlapping is small. But on hardware where short inner loops are expensive, *e.g.,* on vector machines, these optimizations may be essential to achieve advantages of overlapping. Similar loop elimination and code merging for the outer of the horizontal loops over the *N* and *S* subdomains will be profitable, though less than those for the inner loops on *W* and *E*.

Our study seems to suggest that possibly HIRLAM can be modified, to achieve reasonable performance also on clusters with slower (cheaper) interconnects, or on a wide area computer (a Grid computer). However, first we will have to resolve the issues raised by the global communications, needed for the implicit scheme, by the semi-Lagrangian scheme, and by the limited MPI buffer size. We will start with the last.

Our experiments were configured to fit the MPI ("eager-limit") buffer size requirements to allow overlap. *E.g.,* our experiments used 16 levels. Before more levels can be used, or before the halo width can be extended (as needed by the semi-Lagrangian scheme), or before more parameters like trace gases can be added, we will have to find ways around this limit. Possibly, the buffer size can be configured during installation of MPI; possibly buffered communication by MPI can be used; and possibly the message size can be reduced by message splitting. The last of these options will only work if the middle domain is split as well, resulting in more calculation overhead.

On the issue of global communications we make the following remarks. The nature of the calculations that need global communications inhibits implementing overlap straightforwardly. Possibly, algorithms can be designed that permit starting calculations before all data needed to complete the calculations have been received, but we doubt that the time taken by those calculations is significant as compared to the communication times. Even if some overlap can be achieved, results will lose the bitwise reproducibility, because the order of calculations will depend on the domain decomposition. Fortunately, global communications happen infrequently as compared to the communications needed by the semi-Lagrangian HIRLAM implementation, so their optimization is not very urgent. It seems wise to give priority to resolving the issues raised by the semi-Lagrangian scheme.

The first of those is its very implementation in HIRLAM, with a large number of communication steps during each time step. Hand coding for overlap is very cumbersome. Although the essentials of the semi-Lagrangian scheme can be generated by Ctadel (Van der Mark et al., 2004a) the complete scheme has never been generated. A full, generated, implementation will result in a very small number of communication steps, probably one per semi-Lagrangian iteration step, and from there on it would be straightforward to use the work we have done on the Eulerian scheme to experiment with overlapping in the semi-Lagrangian scheme.

A second issue of the semi-Lagrangian scheme is the width of the halo, usually in the order of 10 grid points. Because this is much wider than in the Eulerian scheme, the middle subdomain becomes smaller, and communications more costly. There will be less calculation time to hide the larger communications behind. This is partly compensated by the higher costs for semi-Lagrangian than for Eulerian time stepping. More serious is the fact that the wider halo will result in too large messages to fit into the MPI buffers. An interesting interaction will occur if we also implement the "halo on demand" technique (Van der Mark et al., 2004b). In this technique, advection wind speed and direction determine which part of the halo has to be communicated. It reduces communication. If combined with overlapping, the division into subdomains becomes flow dependent.

In the course of time, the number of grid points in a processor domain has the tendency to decrease (Cats, 2008). Within two decades from now, it will have shrunk to less than 400 grid points. In a semi-Lagrangian scheme, there will be nothing to overlap then if those 400 points are chosen in a 20 by 20 configuration. But in, e.g., a 10 by 40 configuration, N⇔S communication can still be overlapped with the calculations in the grey area of Fig. 5, even though the middle subdomain M has disappeared.

7 Conclusions

We have demonstrated that depending on the available hardware little to considerable savings on total execution times can be reached by overlapping communications with calculations, for the explicit Eulerian HIRLAM scheme. On our hardware, with our compiler, the savings are so substantial that there is no need for an expensive interconnecting network, and even running on a wide area Grid computer hardly incurs any delay. However, if a fast (hence expensive) interconnecting network is available, the attainable savings are negligible, and surely do not warrant the associated code complications.

We base our conclusions on experiments with the Ctadel generated code for the HIRLAM routine DYN. We chose this generated code mainly because initially we anticipated that we would need several optimizations, like loop unrolling and subroutine inlining, to reduce additional computational overhead to a level that the time savings by overlapping would exceed that overhead. However, we found that on our hardware, the overhead is small. Similar results would therefore have been obtained with the original HIRLAM hand code. In the configuration of 64 processors over 2 clusters, we found that pure calculation time of the generated code is not sufficient to fully cover the communication times. In that configuration the hand code, being computationally more expensive, would even have shown bigger profits of overlapping than the generated code.

We investigated several strategies for the overlap. We almost universally found that the strategy with 6 subdomains was the most efficient, in spite of it incurring the largest computational overhead; the reason being that on our hardware that overhead was small. We anticipate that on hardware that favors long inner loops, e.g., on vector machines, our strategy 3s, which was designed to avoid short inner loops, would be preferable.

It will take considerable efforts to implement our findings in a production code, in particular because that would require a fair amount of rewriting of the semi-Lagrangian scheme. We would also have to resolve the problem posed by buffer size limitations. Furthermore, the savings by the overlapping scheme would be reduced by the anticipated challenges and limitations to apply it to the global communications required for the implicit parts of the time stepping schemes. But perhaps the biggest challenge is how to cope with the required code complexities.

References

Gerard Cats, 2008: *24 More Years of Numerical Weather Prediction: A model performance model,* KNMI Scientific Report WR 2008-1, 31 pages. Available from KNMI.

Robert van Engelen, Lex Wolters, and Gerard Cats, 1997: *Tomorrow's Weather Forecast: Automatic Code Generation for Atmospheric Modeling,* IEEE Journal of Computational Science and Engineering, Vol. 4, No. 3, July/September 1997, pp. 22-31.

Ian Foster and Carl Kesselman (Editors), 2004: *The Grid 2: Blueprint for a New Computing Infrastructure,* Elsevier.

Paul van der Mark, Lex Wolters, and Gerard Cats, 2004a: *Semi-Lagrangian Formulations with Automatic Code Generation for Environmental Modeling,* in proceedings of the 19th ACM Symposium on Applied Computing (SAC04), March 2004, Nicosia, Cyprus, ACM Press, pp. 229-234.

Paul van der Mark, Gerard Cats, and Lex Wolters, 2004b: *A Dynamic Application-Driven Data Communication Strategy,* in proceedings of the 18th ACM International Conference on Supercomputing, June 2004, Saint-Malo, France, ACM Press, pp. 146-153.

Myricom, 2006: http://www.myri.com

Frank Seinstra and Kees Verstoep, 2007: DAS3 overview: http://www.cs.vu.nl/das3/overview.shtml. *DAS3 connectivity:* http://www.cs.vu.nl/das3/starplane.shtml

Wikipedia dataflow: http://en.wikipedia.org/wiki/Dataflowprogramming