Low Communication Parallel Multigrid
A Fine Level Approach

Marcus Mohr

System Simulation Group of the Computer Science Department,
Friedrich-Alexander-University Erlangen-Nuremberg
Marcus.Mohr@cs.fau.de

Abstract. The most common technique for the parallelization of multigrid methods is grid partitioning. For such methods Brandt and Diskin have suggested the use of a variant of segmental refinement in order to reduce the amount of inter–processor communication. A parallel multigrid method with this technique avoids all communication on the finest grid levels. This article will examine some features of this class of algorithms as compared to standard parallel multigrid methods. In particular, the communication pattern will be analyzed in detail.

Keywords: elliptic pde, parallel multigrid, domain decomposition, communication cost

1 Introduction

There exists a great variety of different parallel architectures, ranging from clusters of workstations to massively parallel machines. In this paper we will consider the case that the number of processors is significantly smaller than the number of grid points and that memory is distributed. With this background the most common approach to the parallelization of numerical algorithms is grid partitioning. In the special case of multigrid methods nested levels of grids are employed. This approach introduces the need to communicate values related to the points on or near the inner boundaries. The cost of communication naturally limits the possible speedup of a parallel method. This cost can be alleviated by sophisticated programming techniques, e.g. by overlapping communication with calculation.

Generally the communication cost is determined by the number of messages that must be exchanged and the number of bytes that have to be transmitted. In multigrid the number of messages is proportional to the number of domains and therefore independent of the grid level. The number of bytes on the other hand is strongly related to the coarseness of the respective level since it is coupled to the number of interface points.

This leads to another aspect, namely the parallel efficiency of the algorithm, determined by the ratio between communication and computation. This ratio is directly proportional to the ratio of volume and surface of the subgrids and therefore becomes worse on the coarser grid levels. This is the starting point for

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 806–814, 2000.

many approaches to improve parallel multigrid, like e.g. coarse grid agglomeration, multiple coarse grid and concurrent algorithms, see e.g. [6], or methods that employ different cycle schemes like e.g. the U–cycle [9].

A radical approach to reduce fine grid communication has been suggested by Brandt and Diskin in [3]. Here an algorithm that completely eliminates the need for inter–processor communication on several of the finest grids is presented. In the following we will examine a variant of this algorithm. We want to depict some of its characteristics and problems and compare its communication requirements to that of a "conventional" parallel multigrid algorithm. The algorithm will be introduced in Sect. 2. We will compare its communication requirements to that of a "conventional" parallel multigrid algorithm in Sect. 3 and analyze the communication pattern in Sect. 4.

2 Algorithm of Brandt & Diskin

In [3] Brandt and Diskin introduced a parallel multigrid algorithm completely without interprocessor communication on several of the finest levels. This was achieved by the use of segmental–refinement–type procedures, which were originally proposed as out–of–core techniques on sequential computers, cf. e.g. [1]. Since the bulk of communication takes place on the fine grids, it may be especially attractive to save this cost. The effectiveness of such a technique depends on specific machine and problem parameters. The potential benefits are most impressive when communication is slow and expensive with respect to computation.

Their algorithm can be described as follows. Starting from a hierarchy of grids, in a preliminary step all levels, except the coarsest one, are decomposed into as many overlapping subdomains as processors are available. We get sequences of nested subdomains, where a subdomain on a coarser grid occupies a larger area than the corresponding subdomain on the next finer grid. Each such sequence is then assigned to a processor, which starts on it a standard V–cycle, descending through its grid hierarchy. In this process it does not exchange data with its neighbours. When the second coarsest level is reached, the local (approximate) solutions from all processors are used to formulate a global coarse grid problem. This is then solved exactly by some unspecified algorithm, possibly of course a standard parallel multigrid method. The solution of the global coarsest grid problem is used by each processor to correct its local solution approximation on the second coarsest level. As in the following correction steps on the finer levels the values at the interfaces are included into the correction. After this step each processor finishes its V–cycle, again without communication with its neighbours. The algorithm uses the following basic principles to reduce communication:

– Clearly some exchange of information between the processors is inevitable to solve the problem. In the algorithm by Brandt and Diskin however, this data exchange is restricted to the coarse grid correction from the common coarsest grid. So there is no communication on the finer grid levels.

- To compensate errors introduced by the missing exchange of information, each subdomain has a buffer area around it, that fulfills two purposes. On the one hand, if an appropriate relaxation scheme, like e.g. red–black Gauß–Seidel is chosen, the buffer slows the propagation of errors due to wrong values at the interfaces. On the other hand, as in standard multigrid, the coarse grid correction introduces some high–frequency error components on the fine grid. Since the values at the interfaces cannot be smoothed, the algorithm cannot eliminate these components. But in elliptic problems high–frequency components decay quickly. So the buffer area keeps these errors from affecting the inner values too much.
- Nevertheless the algorithm will in general not be able to produce an exact solution of the discrete problem. While this may seem prohibitive at first glance, one should remember that, when solving a PDE, it is the continuous solution one is really interested in. Since the latter is represented by the discrete solution only up to the discretization error, the result of the algorithm is still valuable, as long as it can be guaranteed, that its algebraic error remains at least smaller than the discretization error.
- Since the overlap areas are included in the V–cycle, the algorithm trades communication for calculation. Thus savings in communication time may be partly compensated by the extra computation. This depends on several factors, such as size of the problem, number and arrangement of subgrids, extension of the overlap areas, and the MFLOP– and transfer–rates of the applied hard– and software. It will be further examined in the next section.

A crucial aspect of the above algorithm is of course the choice of an appropriate size for the overlap between subdomains. Until now there exists no general a priori error estimate that would allow to give a bound for the algebraic error depending on the overlap parameter J, but the experiments in [3,7,8] indicate that already small overlaps can produce reasonable results.

3 Efficiency Analysis

In this section we will explore how much overall computation time can be saved by the trade–off between communication and additional computation in the overlap areas. As mentioned above this depends on several architecture and problem parameters. To examine this question we compare a "standard" parallel V–cycle for the coarse grid correction (CGC) to an identical V–cycle that does not exchange data between processors, but instead employs overlapping subdomains. We model the times spent with computation and communication within the algorithm from the following simplifying assumptions:

- Data exchange between processors is performed by message passing.
- Communication and calculation are sequential and cannot be overlapped.
- A processor can send and receive messages simultaneously.

We now define a relative time saving T_{RS} per grid level in the following way

$$T_{\mathrm{RS}} := \frac{T(\mathrm{stand}) - T(\mathrm{nocom})}{T(\mathrm{stand})} \quad . \tag{1}$$

Here $T(\mathrm{nocom})$ is the time for the variant without communication and $T(\mathrm{stand})$ for the standard variant with communication. The times include all the work that has to be done for the specific grid level within one cycle of the CGC–scheme, that is smoothing, calculation of the defect, restriction of the defect, prolongation of the coarse grid solution and adding of the correction. They can be split in the following way

$$T(\mathrm{stand}) = T_{\mathrm{calc}}(\mathrm{stand}) + T_{\mathrm{comm}} \tag{2}$$

$$T(\mathrm{nocom}) = T_{\mathrm{calc}}(\mathrm{stand}) + T_{\mathrm{calc}}(\mathrm{buffer}) \quad . \tag{3}$$

To determine the times for communication and calculation we use the following two models

$$T_{\mathrm{calc}} = \frac{\gamma}{n_p}\Big(\nu F_{\mathrm{smooth}} N_1 + F_{\mathrm{multi}} N_2\Big) \tag{4}$$

$$T_{\mathrm{comm}} = \frac{1}{n_p}\Big(\alpha\, M + \beta\, W\Big) \tag{5}$$

with the parameters

α	latency	F_{smooth}	# of FLOPs per point for smoothing
β	bandwidth	F_{multi}	# of FLOPs per point for CGC
γ	time per FLOP	N_1	# of points that are smoothed
n_p	# of processors	N_2	# of points included in the CGC
ν	sum of smoothing steps	M	# of messages to be exchanged
W	# of words (double precision values) that must be exchanged		

As model problem we consider a 2D elliptic boundary value problem discretized with a 5–point stencil on a logically rectangular grid. We assume the use of a 9–point stencil for restriction, bilinear interpolation as prolongation, and red–black Gauß–Seidel as smoother. For the 3D analogue, we use a 7–point discretization, a 27–point stencil for restriction, and trilinear interpolation. We assume that our global finest grid is quadratic/cubic with N^{dim} points ($N-1$ being a power of 2), and that we have a logical processor grid. Assuming furthermore that a non–overlapping domain decomposition approach is employed for the grid partitioning [5,8], we can, for given values of N, p_x, p_y and eventually p_z, and for a given overlap parameter J, derive the corresponding values of n_p, N_1, N_2, M, and W. We estimate the number of numerical operations per grid point as $F_{\mathrm{smooth}} \approx 9\,(2\mathrm{D})\,/\,11\,(3\mathrm{D})$ and $F_{\mathrm{multi}} \approx 7.5\,(2\mathrm{D})\,/\,9\,(3\mathrm{D})$.

Now we choose two different processor types, i.e. two different values for γ, one with 200 and the other one with 50 MFLOP. We compare the relative time saving T_{RS} per grid level for different grid sizes N and different communication

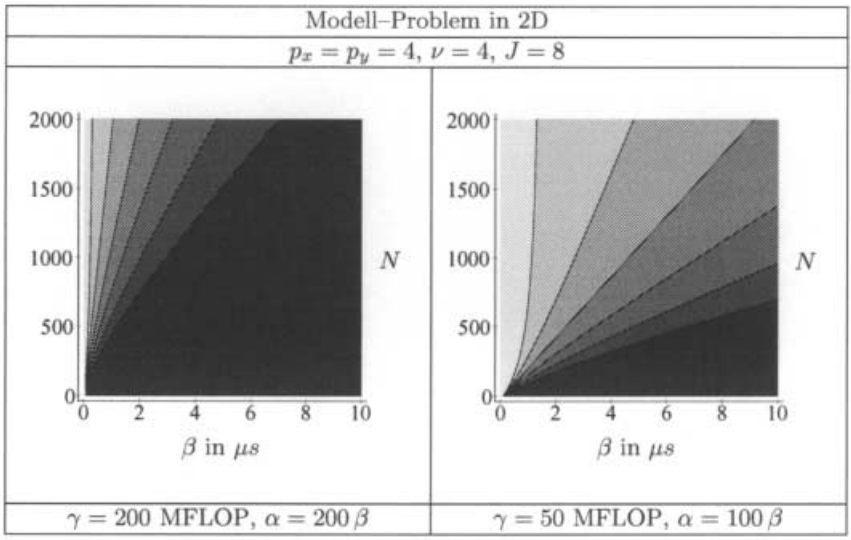

Fig. 1. Relative Time Saving per Grid Level for the 2D–Model Problem

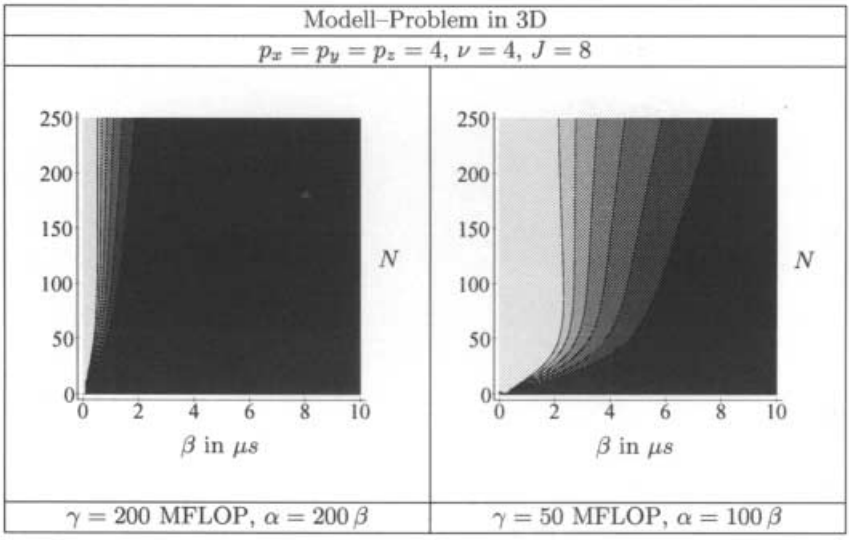

Fig. 2. Relative Time Saving per Grid Level for the 3D–Model Problem

speeds β. Latency effects are taken into account by assuming that $\alpha = \text{const} \cdot \beta$. The results are shown in Fig. 1 and 2.

Communication hardware may have widely varying performance characteristics. For orientation we present typical parameters for current implementations of the message passing interface (MPI). The following values have been taken from [4]:

	Myrinet	Fast Ethernet	Ethernet
α	$70\mu s$	$630\mu s$	$1150\mu s$
β	$0.36\mu s$	$2\mu s$	$11.4\mu s$
α/β	194	315	100

The results in Fig. 1 and 2 confirm the following properties of the approach. The value T_{RS} of the relative time saving grows with β. This was to be expected, since the more expensive communication, the greater the gain by replacing it with computation. However, as the grids become larger T_{RS} is reduced. Although the margin between communication time T_{comm} and additional calculation time $T_{\text{calc}}(\text{buffer})$ becomes more and more favorable as N grows, this is to some extent compensated by the growing influence of the regular computational work $T_{\text{calc}}(\text{stand})$ on the overall time. This can clearly be seen by comparing the figures for the 200 and the 50 MFLOP case. This compensation effect is significantly smaller in 3D than in 2D, since here the number of points and the amount of computation is of order $\mathcal{O}(N^3)$, whereas the number of interface points, responsible for communication, grows like N^2. In 2D, on the other hand, these values are N^2 and N, respectively.

As far as the remaining problem parameters are concerned, some properties can easily be derived. If we consider the overlap parameter J, it can be said that, as long as $N \gg J$, the additional computation grows almost linearly with J. So we have that $T_{\text{RS}} = C_1 - C_2 J$, with constants C_k depending on the other parameters. We found that the parameter ν had little influence on T_{RS}, but the number of processors n_p does. The relative time saving is more favorable, when there are more processors, because then there are fewer points on each subgrid and therefore the above mentioned compensation effect is reduced.

4 The Two Level Brandt–Diskin–Algorithm

In this section we will take a closer look at the two level version of the multilevel cycle in order to analyze in detail the communication pattern of the algorithm and show some of its properties. We do this by means of the model problem on the unit square Ω and restrict us to the case of two subdomains, which will suffice to explain the basic concept.

We discretize the problem on a fine grid $\Omega^h := \{x_{ij} \,|\, 0 \leqslant i, j \leqslant N\}$ and a coarse grid $\Omega^H := \{x_{2i2j} \,|\, 0 \leqslant i, j \leqslant N/2\}$. Here $x_{ij} = (ih, jh)$ denotes a grid point on the i^{th} vertical and the j^{th} horizontal line. We decompose the fine grid into two equally large parts $\Omega^{h,1}$ and $\Omega^{h,2}$ with the grid line $j = N/2$ as common

boundary. Now we augment each subgrid with a buffer zone of $J > 0$ grid lines and get the extended subdomains

$$\hat{\Omega}^{h,1} := \left\{ x_{ij} \;\middle|\; j \leqslant N/2 + J \right\} \quad \text{and} \quad \hat{\Omega}^{h,2} := \left\{ x_{ij} \;\middle|\; j \geqslant N/2 - J \right\} \; . \tag{6}$$

The quantity $J > 0$ describes the amount of overlap between the two subdomains. It is required that J is a multiple of 2 to ensure that there are coarse grid points that coincide with interface points on the fine grid.

Each subgrid is now assigned to one processor p_1 and p_2. Starting from an approximation u_S^h to the discrete solution of the fine grid problem the algorithm is defined by iteratively performing the following seven steps:

1. Both processors perform separate pre–smoothing steps.
2. A global approximate solution is composed from the local solutions.
3. A global defect function is composed.
4. With the global solution and defect the right hand side of the coarse grid equation is calculated according to the FAS–scheme.
5. The coarse grid equation is solved exactly (by whatever means).
6. Each processor uses the coarse grid solution to correct its local approximate.
7. Both processors perform separate post–smoothing steps.

In step 1 and 7 the iteration method used is red–black Gauß–Seidel and there is no communication between the processors to update values along the interfaces. As a consequence the local solutions $u^{2,1}$ and $u^{2,2}$ will start to differ.

Steps 2 and 3 are of virtual character in the sense that no processor knows the global functions completely. Formally the global solution u^h is computed by taking the values of the local solutions in the interior of the subdomains $\Omega^{2,p}$ and their average along their common boundary. The same is done for the global defect function, so that we get

$$r^h(x_{i,j}) := \begin{cases} r^{h,1}(x_{i,j}) & \text{if } j < N/2 \\ r^{h,1}(x_{i,j})/2 + r^{h,2}(x_{i,j})/2 & \text{if } j = N/2 \\ r^{h,2}(x_{i,j}) & \text{if } j > N/2 \end{cases} \tag{7}$$

Here $r^{h,p} = f^h - L^h u^{h,p}$ denotes the local defect functions and f^h and L^h are to be interpreted as restrictions on the extended subdomains $\hat{\Omega}^{h,p}$. We point out that in general the so defined global defect function r^h is different from the defect $f^h - L^h u^h$ of the global solution function at points near the interface. We will return to this fact later on.

In a normal FAS–scheme the correction in step 6 would be performed as

$$u^h_{\text{new}} = u^h_{\text{old}} + I_H^h \left[u^H - I_h^H u^h_{\text{old}} \right] \tag{8}$$

where u^H is the solution of the coarse grid problem obtained in step 5, and I_H^h and I_h^H are a prolongation and a restriction operator. In the two level cycle the same is done, but since the global solution is not locally available, the local

solution is corrected instead. In order to understand how this correction leads to an update of information between the processors, we re-write the correction. We define a prolongation operator

$$J_H^{h,p} : \Omega^H \to \hat{\Omega}^{h,p} \tag{9}$$

calculating the values in $\hat{\Omega}^{h,p}$ from the respective values in $\Omega^H \cap \hat{\Omega}^{h,p}$ by bilinear interpolation and a restriction operator

$$J_{h,p}^H : \hat{\Omega}^{h,p} \to \Omega^H \tag{10}$$

which computes the values in $\Omega^H \cap \hat{\Omega}^{h,p}$ through injection and sets the values in $\Omega^H \backslash \hat{\Omega}^{h,p}$ to zero. Denoting by E^p the identity operator on $\hat{\Omega}^{h,p}$, we get

$$u_{\text{new}}^{h,p}(P) = \left(E^p - J_H^{h,p} J_{h,p}^H\right) u_{\text{old}}^{h,p}(P) + \left(J_H^{h,p} u^H\right)(P) \,. \tag{11}$$

Consider now a point $P = x_{ij}$ that lies in the overlap area $\hat{\Omega}^{h,1} \cap \hat{\Omega}^{h,2}$. For such a point both processors store a local solution value. Due to lack of communication during relaxation these values will in general differ prior to the correction step. The second term in the correction formula will lead to the same value on both processors, while the first term in (11 will vanish for points that can be found on the coarse grid. So in this case the values of the two processors at the point will coincide after the correction. For points not on the coarse grid, the first term will in general not vanish. This is due to interpolation errors. So the values of the two processors will be composed of a common part, namely $(J_H^{h,p} u^H)(x_{ij})$, and a disturbance, which is a remainder of the old function value at each processor. In this way an information update between the processors is embedded into the correction.

In the following, we want to point out some of the features of the two level cycle. First of all, since the algorithm represents an iteration on the pair of local solution functions $(u_i^{h,1}, u_i^{h,2})$, we have to analyze its convergence.

The experiments in [3] as well as the analysis for the special case of exact solvers as smoothers in [7] indicate that the algorithm will converge for every choice of initial values. But contrary to standard multigrid the result depends on the initial values. These can be grouped into equivalence classes according to their hierarchical offset, with every class converging to the same limit function. The reason for this is that the first term in (11) will always reproduce the hierarchical offset of the solution, while the second term represents a bilinear interpolation and therefore has no hierarchical offset at all. As a consequence the hierarchical offset of the initial values along the inner boundaries is never changed by the algorithm.

As another property we require that for a proper choice of the overlap parameter J the error with respect to the true discrete solution remains at least of the same order of magnitude as the discretization error. In [3,7] it was shown by experiment that this can already be achieved with an overlap of 2 to 4 grid lines. The accuracy can be improved, by a different way to compose a global

defect function r^h. As already mentioned, the global defect function when set up according to (7) does not match the defect $\bar{r}^h := f^h - L^h u^h$ of the global solution approximation. If we use $\bar{r}^h$ as the global defect, the algebraic error will be smaller than with the use of r^h and it will strongly be restricted to the vicinity of the common boundary of the subdomains. In this case also smaller overlaps are sufficient to keep the algebraic error smaller than the discretization error and this will often be achieved in a smaller number of cycles, [7,8].

This approach does not cause extra cost. Since the calculation of the defect $f^h - L^h u^h$ as well as the prolongation by full weighting and the application of the coarse grid operator L^H are linear operations the amount of communication needed to set up the coarse grid equation is the same independent of whether we use $\bar{r}^h$ or r^h.

5 Conclusions

In this paper we have shown that the approach by Brandt and Diskin to parallelize multigrid methods can be used to reduce overall computation time. This remains valid even for high–speed communication infrastructures as long as the processor speeds are fast enough. We have presented an analysis of the communication pattern of the two–level–version of their algorithm and have shown a way to improve its performance. Questions that remain open are a priori estimates for the algebraic error of the algorithm as well as convergence rates.

References

1. A. Brandt, Multi–level adaptive solutions to boundary value problems, Mathematics of Computation 31 (1977) 333 - 390.
2. A. Brandt, B. Diskin, Multigrid Solvers on Decomposed Domains, Contemporary Mathematics 157 (1994) 135 - 155.
3. B. Diskin, Multigrid Solvers on Decomposed Domains, M. Sc. Thesis, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, 1993.
4. M. Griebel and G. Zumbusch, Parnass: Porting gigabit–LAN components to a workstation cluster, in: W. Rehm, ed., Proceedings des 1. Workshop Cluster-Computing, 6.-7. November 1997, in Chemnitz, (Chemnitzer Informatik Berichte, CSR-97-05) 101-124.
5. M. Jung, On the parallelization of multi–grid methods using a non–overlapping domain decomposition data structure, Appl. numer. Math. 1 (1997) 119-138.
6. L. Matheson, R. Tarjan, Parallelism in Multigrid Methods: How much is too much?, Int. J. Parallel Programming 5 (1996) 397 - 432.
7. M. Mohr, Kommunikationsarme parallele Mehrgitteralgorithmen, Diplomarbeit, Institut für Mathematik, TU München, 1997.
8. M. Mohr, U. Rüde, Communication Reduced Parallel Multigrid: Analysis and Experiments, Technical Report No. 394, University of Augsburg, 1998.
9. D. Xie, L. Scott, The Parallel U–Cycle Multigrid Method, Virtual Proceedings of the 8$^{\text{th}}$ Copper Mountain Conference on Multigrid Methods, MGNET, 1997, (http://casper.cs.yale.edu/mgnet/www/mgnet.html).