PARALLEL DOMAIN-ORIENTED MULTILEVEL METHODS

MICHAEL GRIEBEL

Institut für Informatik, Technische Universität München
Arcisstraße 21, D-80290 München
email: griebel@informatik.tu-muenchen.de

1. Results. We now report on the results of our numerical experiments. We consider the model problem

$$-\Delta u = f \text{ in } \Omega = (0,1)^2$$

with Dirichlet boundary conditions. Note that this simple Poisson problem is a hard test case when it comes to parallelization, since the ratio of arithmetic operations to communication costs is worse than for all other second order elliptic PDE's.

We implemented the point-oriented Gauss-Seidel iteration, i.e. the symmetric Gauss-Seidel iteration on the semidefinite system with the ordering as indicated in Figure **??**. We run it in parallel on a network of HP 9000/720 workstations that are equipped with *two* Ethernet cards. Thus, it is possible, to build tree-like configurations, see Figure 1. Communication can take place in parallel on the Ethernet connections of the different levels of the processor tree. The processors of each level of the processor tree can work in parallel. Our code is written in C and uses the PVM communication system.

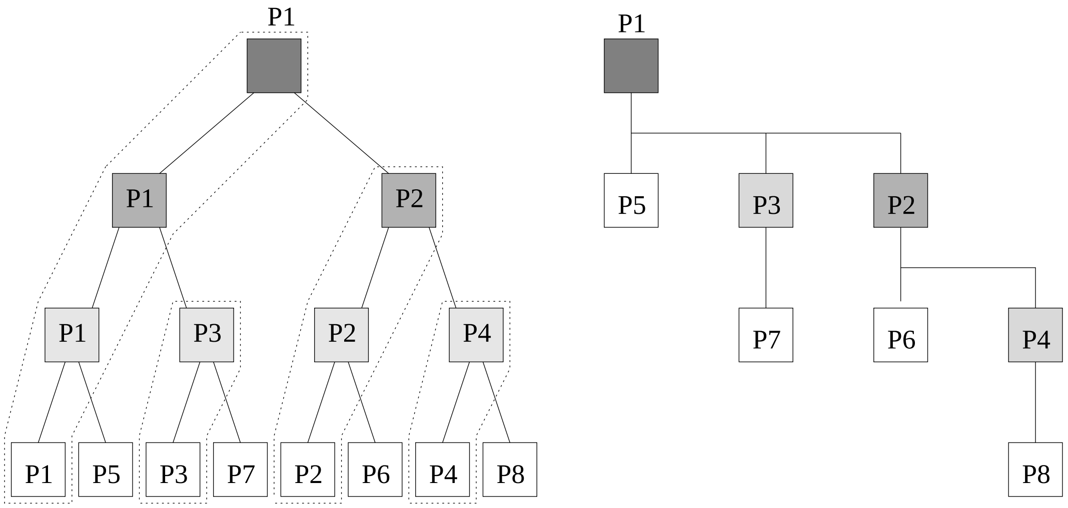

FIG. 1. *Process tree (left) and network configuration (right).*

In the following tables we show the time in seconds, used for computation, communication and setup in one iteration step of the symmetric Gauss-Seidel relaxation performed on the semidefinite system with the ordering as indicated in Figure **??**. We considered it for $k = 3, \ldots, 11$ levels and $P = 1, \ldots, 16$ processors.

First, we measured the communication setup time[1] for one symmetric Gauss-Seidel iteration step. The results are shown in Table 1. They are independent of the number k of levels but only depend on the number P of processors. We obtained $S \approx 0.023 \cdot \log_2 P$.

Table 1
Setup time S for $k = 3, \ldots, 11$.

P	1	2	4	8	16
S	0.000	0.022 - 0.024	0.044 - 0.047	0.067 - 0.075	0.090 - 0.097

Secondly, we measured the communication time C, the computation time W and the total time T for one symmetric Gauss-Seidel iteration of the overall algorithm, i.e. $T = S + C + W$. The results are shown in Table 2. Note that the missing values for $k = 11, P = 1, 2$ are not given since their computation involved paging and swapping effects.

Table 2
Communication time C, computation time W and total time T.

	P	1	2	4	8	16
$k = 3$	C	0.000	0.013	0.026	0.042	0.059
	W	0.000	0.000	0.000	0.000	0.000
	T	0.000	0.036	0.070	0.109	0.149
$k = 4$	C	0.000	0.012	0.025	0.044	0.061
	W	0.003	0.002	0.002	0.001	0.001
	T	0.003	0.036	0.072	0.112	0.152
$k = 5$	C	0.000	0.012	0.026	0.046	0.064
	W	0.013	0.008	0.005	0.002	0.001
	T	0.013	0.042	0.077	0.116	0.155
$k = 6$	C	0.000	0.013	0.027	0.050	0.068
	W	0.062	0.030	0.016	0.012	0.011
	T	0.062	0.066	0.091	0.131	0.170
$k = 7$	C	0.000	0.016	0.034	0.057	0.073
	W	0.356	0.160	0.077	0.038	0.035
	T	0.356	0.200	0.157	0.166	0.192
$k = 8$	C	0.000	0.022	0.042	0.065	0.104
	W	1.599	0.780	0.376	0.180	0.095
	T	1.599	0.825	0.466	0.321	0.293
$k = 9$	C	0.000	0.027	0.067	0.102	0.127
	W	6.478	3.230	1.620	0.812	0.404
	T	6.478	3.291	1.734	0.987	0.627
$k = 10$	C	0.000	0.068	0.118	0.173	0.217
	W	26.111	13.064	6.538	3.304	1.696
	T	26.111	13.156	6.703	3.551	2.001
$k = 11$	C					0.392
	W					6.657
	T					7.147

[1] The initialization setup time, i.e. the time necessary to distribute the code and the processes on the network, to allocate necessary data at the beginning of the computation, and so on, is not contained.

Regarding the communication time it can be seen that C behaves for fixed k like $O(\log_2 P)$. Furthermore, for sufficiently large values of k, the communication time necessary to provide the final, i.e. largest separator with its data, c.f. Figure **??**, tends to dominate C. Thus, (without the second, aforementioned parallelization step on the separating lines) we must obtain $C = O(2^k)$ for fixed P, which corresponds to the size of the separator anyway. A short analysis of our data shows that

$$C \approx \log_2(P) \cdot (0.013 + 4 \cdot 10^{-5} \cdot 2^k)$$

at last for $P \leq 32$. (From theory [**?**], we know that the amount of data to be communicated sequentially does not depend on P for sufficiently large values of P since some sort of geometric series develops. We then would get $C \approx c_1 \cdot \log_2(P) + c_2 2^k \cdot 6.75$ with appropriate constants c_1, c_2 of size 10^{-5} but this can not be seen from our table.)

For the computation time, we obtain for sufficiently large values of k

$$W \approx 2.5 \cdot 10^{-5} \cdot 2^{2k}/P.$$

For small values of k an additional factor depending on 2^k seems to come in that might stem from the work spent on the largest separator. It possesses a constant in the same range (10^{-5}). Furthermore, we see that W nicely scales with $W^{k,P} = W^{k+1,4P}$ for $k \geq 7$, whereas for $k < 6$ the influence of the separator still dominates.

Altogether, we see that the part of the execution time which only depends on the number of communication steps is quite dominating. For example, even in the case $P = 16, k = 8$ the setup time S is still as large as the computation time W. We know that this effect is much more crucial for parallelized multigrid methods since then communication has to take place on every level and is therefore dependent on the number k of levels. Thus, we infer that our approach is superior to most parallelized multigrid methods especially on a cluster of workstations (with a binary-tree like configuration).

In Table 3 the efficiencies are given for $k = 7, .., 11$. Here, the efficiency E is defined by the ratio

$$E(P) = \frac{T(1)}{T(P) \cdot P}$$

where $T(P)$ denotes the time needed for the calculation on P processors. For $k = 11$, $T(1)$ was obtained by multiplying $T(1)$ for $k = 10$ by four.

TABLE 3
Efficiencies E.

P	2	4	8	16
$k = 7$	89.1	56.7	26.8	11.7
$k = 8$	97.0	85.7	62.3	34.1
$k = 9$	98.4	93.4	82.1	64.6
$k = 10$	99.2	97.4	92.0	81.2
$k = 11$				91.3

We see that we obtained good efficiencies (above 81 – 89 percent) for the case $k - 6 \geq \log_2(P)$ even if we use a network of workstations as a parallel computing system.