54

Scalability and load imbalance for domain decomposition based transport

P. Wilders[1]

Introduction

Transport phenomena play an important role in reservoir engineering and surface or subsurface environmental studies. From the application viewpoint, there is a need to resolve large computational models with fine grids (for example, to study fingering). Scaled distributed computations are useful at this point and, as a consequence, scalability is of some concern in this field. We study a simple 2D tracer flow in a porous medium in order to reveal some basic properties at this point.

The domain decomposition method under consideration is a one-level Krylov-Schwarz method, formulated in terms of interface variables [BW97], [WB99]. This method is combined with ILU-preconditioned BiCGSTAB for the subdomain inversion. For the parallel implementation the focus is on distributed MIMD platforms with explicit parallel programming and a gather/scatter as the basic communication scheme. The parallel performance is studied, both analytically and experimentally on an IBM-SP2. It is shown that load balancing effects are of major importance and the impact of this observation on the scalability will be examined. Related studies on parallel performance modelling of domain decomposition methods have been undertaken in [GK89], [HZ93], [CS95], [BSK95]. In these studies load balancing effects are either absent or have assumed to be absent.

1 Delft University, Dept. of Math., P.O. Box 5031, 2600 GA Delft, NL, p.wilders@math.tudelft.nl

Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund ©1999 DDM.org

Following [HZ93], [CSSY94], our study deals with relative measures (relative efficiency, overhead) and concerns, speaking in terms of [Qui94], the parallelizability. This can be seen as a basic step in a full parallel performance analysis. The absolute efficiency may be written as a product of the numerical efficiency and the relative (or parallel) efficiency. This enables a separate study of the two. Of course, the application of one-level methods leads to a numerical efficiency that does not scale perfectly, e.g. [AN95], and in some cases degradation occurs. However, for linearly scaled transport problems the degradation at this point has been found to be moderate [Cai91], [WF98]. Moreover, it is known that in a two-level method the coarse grid correction leads to a significant degradation of the parallel efficiency, e.g. [HZ93].

Physical and numerical issues

We consider a scalar conservation law of the form

$$\varphi\frac{\partial c}{\partial t} + \nabla \cdot [\mathbf{v}f(c) - D\nabla c] = 0 \quad , \quad \mathbf{x} \in \Omega \in I\!R^2 \quad , \quad t > 0 \quad . \tag{1}$$

In this paper only tracer flows are considered. The coefficients $\varphi, \mathbf{v}$ and D are assumed to be time-independent and $f(c) = c$. Our interests are on the advection-dominated case, i.e. the diffusion tensor D depends on small parameters. For the spatial discretization we employ cell-centered triangular finite volumes, based upon a 10-point molecule [WF97]. This results in the semi-discrete system

$$L\frac{dc}{dt} = F(c, \mathbf{x}) \quad , \tag{2}$$

for the centroid values of the concentration. L is a diagonal matrix, containing the cell values of the coefficient φ multiplied with the area of the cell. Moreover, F is a nonlinear differentiable function of c. We set

$$J = \frac{\partial F}{\partial c} \quad . \tag{3}$$

The Jacobian J represents a sparse matrix with a maximum of 10 nonzero elements in each row.

The linearly implicit trapezoidal rule is used for the time integration:

$$(\frac{L}{\tau_n} - \frac{1}{2}J^n)c^{n+1} = (\frac{L}{\tau_n} - \frac{1}{2}J^n)c^n + F^n \quad . \tag{4}$$

Here, τ_n denotes the time step. The scheme is second-order accurate in time.

The linear system (4) is solved iteratively by means of a one-level Krylov-Schwarz domain decomposition method, in our case a straightforward nonoverlapping additive Schwarz preconditioner with GMRES as the Krylov subspace method. ILU-preconditioned BiCGSTAB is used for the approximate inversion of the subdomain problems.

Experimental studies have been conducted for tracer flows in reservoirs. The velocity $\mathbf{v}$

is obtained from the pressure equation, using strongly heterogeneous permeability data provided by Agip S.p.A (Italian oil company). In this paper we consider the quarter of five spots, a well-known test problem for porous medium applications. A fully balanced blockwise decomposition into Q subdomains is introduced. The total number of unknows is N. Extensive data with regard to timing and parallel performance (on a SP2) are available for linearly scaled problems, i.e. $N = N_0 Q$ (with $N_0 = 3200$) and $Q = 4, 9, 16$. As soon as N varies, either in the analysis or in experiments, it is important to take application parameters into account [SHG93]. From the theory of hyperbolic difference schemes it is known that the Courant number is the vital similarity parameter. Therefore, we fix the Courant number. This means that both the spatial grid size h ($N = O(1/h^2)$) and the time step τ vary with τ/h constant. For linearly scaled problems this results in $h, H, \tau = O(1/\sqrt{Q})$ with H measuring the subdomain size.

Timing and load imbalance

The computational domain has been divided into Q subdomains. Most of the computation within subdomain q can be done independently of the other subdomains; this part of the computation is referred to as *subtask* q. The subtasks may be executed in parallel. The computation is divided into $(p+1)$ tasks, numbered $\overline{j = 0}, ..., p$. Task 0 is the *sequential task* and the tasks $1, ..., p$ are the *distributed tasks*. $Q(j)$ of the subtasks are assigned to the distributed task j. We distinguish p *processes*. Task 0 and task 1 are taken together to form process 1. For $j > 1$, process j is identical to task j. Each process is executed on its own processor. The processes are connected via explicit parallel programming, using message passing.

Our concern is on the computational heart of the code, i.e. the time stepping loop. For the ease of presentation we consider a single time step. There are three phases in the program, i.e. communication and computation either as a part of the sequential task 0 or as a part of the distributed tasks $1, ..., p$. In this paper we consider the fully synchronized case only. The different processes are synchronized explicitly each time the program switches between two of these three phases. The elapsed time in a synchronized p processor run is denoted with T_p. It follows that

$$T_p = T^{(s)} + T_p^{(d)} + T_p^{(c)} \quad , \tag{5}$$

with $T^{(s)}$ the computational time spent in the sequential task 0, $T_p^{(d)}$ the maximal time spent in the distributed tasks and $T_p^{(c)}$ the communication time. Of course, $T_1^{(c)} = 0$.

Let us define

$$T_1(p) = T^{(s)} + pT_p^{(d)} \quad . \tag{6}$$

For $p = 1$, there holds $T_1(p) = T_1$. For $p > 1$, $T_1(p)$ is the elapsed time of a single processor shadow run, carrying out all tasks in serial, while forcing the distributed tasks to consume the same amount of time. It is clear that $T_1(p) - T_1$ presents a measure of idleness in the parallel run due to load balancing effects. It easily follows that

$$T_1(p) - T_1 = pT_p^{(d)} - T_1^{(d)} \quad . \tag{7}$$

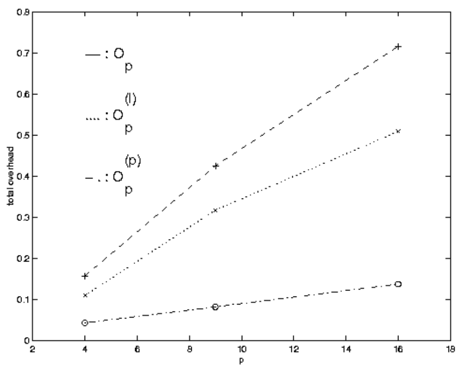

Figure 1 Overhead O_p and its components $O_p^{(l)}, O_p^{(p)}$.

Performance measures; factorization and approximation

The (relative) *efficiency* E_p, the (relative) *costs* C_p and the (relative) *overhead* O_p are defined by

$$E_p = \frac{T_1}{pT_p} \quad , \quad C_p = \frac{1}{E_p} = \frac{pT_p}{T_1} \quad , \quad O_p = C_p - 1 \quad . \tag{8}$$

Note that in (8) we compare the parallel execution of the algorithm (solving a multi-domain problem with Q subdomains) with the serial execution of the same algorithm (*relative*).

We introduce the following factorization of the costs C_p:

$$C_p = C_p^{(l)} C_p^{(p)} \quad , \quad C_p^{(l)} = 1 + O_p^{(l)} \quad , \quad C_p^{(p)} = 1 + O_p^{(p)} \quad , \tag{9}$$

where

$$O_p^{(l)} = \frac{T_1(p) - T_1}{T_1} \quad , \quad O_p^{(p)} = \frac{pT_p - T_1(p)}{T_1(p)} \quad . \tag{10}$$

It follows that

$$O_p = O_p^{(l)} + O_p^{(p)} + O_p^{(l)} O_p^{(p)} \quad . \tag{11}$$

The term $O_p^{(l)}$ is associated with load balancing effects. $O_p^{(p)}$ is referred to as the parallel overhead. $O_p^{(p)}$ combines sequential overhead and communication costs. Figure 1 presents the overhead O_p and its components $O_p^{(l)}, O_p^{(p)}$ such as measured in the linearly scaled experiments, mentioned in Section 54. The importance of load balancing effects can be observed. Let us introduce

$$\tilde{O}_p^{(l)} = \frac{pT_p^{(d)} - T_1^{(d)}}{T_1^{(d)}} \quad . \tag{12}$$

From (7), (10) it follows that

$$\frac{\widetilde{O}_p^{(l)} - O_p^{(l)}}{O_p^{(l)}} = \varepsilon \quad , \quad \varepsilon = \frac{T^{(s)}}{T_1^{(d)}} \quad . \tag{13}$$

Here, ε measures the fraction of the work done in the sequential task and this fraction is small ($< 1\%$ in our experiments). Therefore, we expect $\widetilde{O}_p^{(l)}$ to present a good approximation of $O_p^{(l)}$.

Performance model for a square domain

As before, a fully balanced blockwise decomposition of the square domain into Q subdomains is employed; the total number of unknowns is N, N/Q per subdomain. The total number of edges on internal boundaries is B, on the average B/Q per subdomain. The number of interface variables depends linearly on B. B is given by

$$B = \sqrt{8N}(\sqrt{Q} - 1) \quad . \tag{14}$$

We assume that either $p = 1$, i.e. a serial execution of all subtasks, or $p = Q$, i.e. the subtasks coincide with the distributed tasks. In first approximation there holds

$$\frac{p}{Q} T_p^{(d)} = \alpha_1 \frac{N}{Q} + M(\alpha_2 \frac{B}{Q} + \alpha_3 \frac{N}{Q} I_p) \quad , \tag{15}$$

with

$$I_p = \begin{cases} \frac{1}{M} \sum_{m=1}^{M} \frac{1}{Q} \sum_{q=1}^{Q} I(m,q) & , \quad p = 1 \\ \frac{1}{M} \sum_{m=1}^{M} \max_{q=1,\dots,Q} I(m,q) & , \quad p = Q \quad . \end{cases} \tag{16}$$

Here, $m = 1, ..., M$ counts the number of matrix-vector multiplications in the domain decomposition iteration (outer iteration) and $I(m, q)$ denotes the number of matvec calls associated with the inner iteration in subdomain q for the m-th outer matvec call. The first term on the rhs of (15) corresponds with building processes (subdomain matrix, etc.). The third term on the rhs of (15) reflects the inner iteration and the second term is due to correcting the subdomain rhs.

A least-squares estimation of the coefficients in (15) has been carried out, using the linearly scaled experiments mentioned in Section 54 (six measurements available for fitting (15)). This leads to

$$\alpha_1 = .72 * 10^{-4} \quad , \quad \alpha_2 = .59 * 10^{-4} \quad , \quad \alpha_3 = .26 * 10^{-5} \quad . \tag{17}$$

Using (12), (15), it is now straightforward to obtain the analytical approximation $\widehat{O}_p^{(l)}$ of the overhead function $O_p^{(l)}$:

$$\widehat{O}_p^{(l)} = \frac{\alpha_3 N M (I_p - I_1)}{\alpha_1 N + M(\alpha_2 B + \alpha_3 I_1)} \quad . \tag{18}$$

This shows that the load imbalance is due to variations of the number of BiCGSTAB iterations over the subdomains. The load balancing effects seems to be that strong, because we are dealing with an advection-dominated transport problem in which a front, connecting two constants states, is moving through the domain. This means that there are subdomains in which the solution is constant and little happens, besides subdomains close to the front with a lot of activity.

We set

$$\text{Dev} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{O}_p^{(l)} - O_p^{(l)}}{O_p^{(l)}}\right| * 100\% \quad , \tag{19}$$

with n denoting the number of measurements for $O_p^{(l)}(n = 3;\ p = Q = 4, 9, 16)$. It is found that Dev $= 5.8$, showing that the analytical approximation is quite reliable.

Scalability

Using the isoefficiency metric a parallel algorithm is called scalable if it is possible to find curves in the (p, N)-plane on which the efficiency E_p is constant [GK93], [CSSY94]. A sufficient condition for the existence of isoefficiency curves is

$$\lim_{\substack{N\to\infty \\ p \text{ fixed}}} E_p > 0 \quad \text{or} \quad \lim_{\substack{N\to\infty \\ p \text{ fixed}}} O_p < \infty \quad . \tag{20}$$

The condition regarding the overhead function is equivalent with (see (11))

$$\lim_{\substack{N\to\infty \\ p \text{ fixed}}} O_p^{(l)} < \infty \quad , \quad \lim_{\substack{N\to\infty \\ p \text{ fixed}}} O_p^{(p)} < \infty \quad . \tag{21}$$

A discussion of the second limit in (21) is beyond the scope of the present paper. We only remark that numerical experiments indicate that $O_p^{(p)}$ approaches zero in the limit.

In Section 54 an analytical approximation of $O_p^{(l)}$ for square domain has been given together with arguments showing that this approximation is valuable, at least for linearly scaled problems. Here, we use this approximation to investigate the limiting process in (21). From (14), (18) it follows that

$$\widehat{O}_p^{(l)} \sim F_1 F_2 \quad , \quad N \to \infty \ , \ p \,\text{fixed} \tag{22}$$

with

$$F_1 = \frac{M I_1}{\frac{\alpha_1}{\alpha_3} + M I_1} \quad , \quad F_2 = \frac{I_p - I_1}{I_1} \quad . \tag{23}$$

Note that (17) leads to $\alpha_1/\alpha_3 \approx 28$.

The numbers F_1 and F_2 have been computed, as an average over multiple time steps, for $p = Q = 4$ (4 subdomains) and $N = 3200, 12800, 51200$. Table 1 presents all relevant information (note that these data can be obtained from a serial run). The

Table 1 Iterative properties for 4 subdomains ($p = Q = 4$) and F_1F_2, the approximation of load imbalance.

N	M	I_1	I_p	F_1	F_2	F_1F_2
3200	11.2	5.8	7.0	.70	.21	.15
12800	11.5	6.2	8.0	.72	.29	.21
51200	12.0	6.8	8.6	.74	.26	.19

data in Table 1 indicate that the parallel algorithm meets the first condition of (21). However, the overhead due to load balancing effects does not approach zero in the limit and some degradation of the efficiency has to be accepted. The values found for F_1F_2 in Table 1 are in agreement with some early experiments done on a SP2. In fact, these experiments gave $O_p = .18, .19, .20$ for, respectively, $N = 3200, 12800, 51200$.

Concluding remarks

We have presented a parallel performance model for domain decomposition based transport. The model was used to investigate load balancing effects. Load imbalance due to variable iterative properties of the inner iteration over the subdomains presents a major factor. The analysis shows that load imbalance excludes efficiencies close to one. However, it was observed that load balancing effects do not prevent scalability in the isoefficiency metric. Experimental results indicate that isoefficiency curves with an efficiency around 0.8 exist.

The values of M, I_1 and I_p found in Table 1 indicate that the iterative procedures depend only weakly on the mesh size h for a fixed number of subdomains. With regards to the domain decomposition iteration a similar observation was done in [BW97]. A further study of the iterative procedures is necessary in order to present theoretical arguments for this behaviour.

Acknowledgment

The research was supported by the Dutch Ministery of Economic Affairs as part of the HPCN-TASC program.

REFERENCES

[AN95] Axelsson O. and Neytcheva M. (1995) Scalable parallel algorithms in CFD computations. In Hafez M. and Oshima K. (eds) *Computational Fluid Dynamics Review 1995*, pages 837–857. Wiley.

[BSK95] Brakkee E., Segal A., and Kassels C. (1995) A parallel domain decomposition algorithm for the incompressible Navier-Stokes equations. *Simulation Practice and Theory* 3: 185–205.

[BW97] Brakkee E. and Wilders P. (1997) The influence of interface conditions on convergence of Krylov-Schwarz domain decomposition for the advection-diffusion equation. *J. Scientific Computing* 12: 11–30.

[Cai91] Cai X. (1991) Additive Schwarz algorithms for parabolic convection-diffusion equations. *Numer. Math* 60: 41–61.

[CS95] Chan T. and Shao J. (1995) Parallel complexity of domain decomposition methods and optimal coarse grid size. *Parallel Computing* 21: 1033–1049.

[CSSY94] Carey G., Schmidt J., Singh V., and Yelton D. (1994) A prototype scalable, object-oriented finite element solver on multicomputers. *J. Par. Distr. Comp.* 20: 357–379.

[GK89] Gropp W. and Keyes D. (1989) Domain decompositions on parallel computers. *Impact of Computing in Science and Eng.* 1: 421–439.

[GK93] Gupta A. and Kumar V. (1993) Performance properties of large scale parallel systems. *J. Par. Distr. Comp.* 19: 234–244.

[HZ93] Hoffmann K. and Zou J. (1993) Parallel efficiency of domain decomposition methods. *Parallel Computing* 19: 1375–1391.

[Qui94] Quinn M. (1994) *Parallel computing: theory and practice*. Mc Graw-Hill. second edition.

[SHG93] Singh J., Hennessy J., and Gupta A. (1993) Scaling parallel programs for multiprocessors: methodology and examples. *IEEE Computer* July: 42–50.

[WB99] Wilders P. and Brakkee E. (1999) Schwarz and Schur: a algebraical note on equivalence properties. *SIAM J. Sci. Comp.* to be published.

[WF97] Wilders P. and Fotia G. (1997) Implicit time stepping with unstructured finite volumes for 2D transport. *J. Comp. Appl. Math.* 82: 433–446.

[WF98] Wilders P. and Fotia G. (1998) One-level Krylov-Schwarz domain decomposition for finite volume advection-diffusion. In Bjørstad P., Espedal M., and Keyes D. (eds) *Domain Decomposition Methods 9*, pages 558–565. Domain Decomposition Press, Bergen.