88

Scalability of Industrial Applications on Different Computer Architectures

M. Kahlert, M. Paffrath, and U. Wever

1 Introduction

In the field of partial differential equations, domain decomposition methods are among the most effective parallelization techniques. A direct parallelization leads to so-called data or grid partitioning strategies. The parallel version shows the same algorithmic behavior as the serial one but suffers from a sometimes considerable communication overhead. This disadvantage has led to indirect parallelization strategies which are characterized by an additional outer iteration. This paper presents real life examples from *fabrication of semiconductor devices* and *nuclear reactor analysis*, both for a workstation network and for a multiprocessor-shared memory architecture. In [Kah94] we have shown that for process simulation of semiconductor devices direct parallelization methods are superior to indirect ones, even for workstation clusters. In the present paper we restrict ourselves to direct methods for both application fields: parallel multigrid in connection with Newton's method and parallel nonlinear multigrid. In order to reduce communication overhead, the corresponding smoothing algorithms are changed.

The outline of the paper is as follows: Section 2 illuminates some aspects of multigrid parallelization, Section 3 gives an introduction to the applications, namely point defect and neutron kinetics simulation, and presents results for a workstation cluster and a shared memory system.

2 Aspects of Multigrid Parallelization

For both applications linear prolongation and restriction are used, the restriction operator being the transpose of the prolongation operator. On the finer grids, Gauß-Seidel iterations are used as smoother. On the coarsest grid, the system of equations is solved exactly: by serial Gaussian elimination in case of neutron kinetics simulation,

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

and by a serial or parallel conjugate gradient method in case of process simulation. The serial solution on the coarsest grid can either be computed by a separate host or in parallel by each host. The fastest variant may depend on the underlying hardware architecture. Because communication only takes place in a parallel coarse grid correction and the smoothing algorithm, these algorithms alone are described briefly. After preliminaries on matrix decomposition, parallel conjugate gradients and parallel Gauß-Seidel are described.

Matrix Decomposition

Let Ω, the domain of definition, be decomposed into overlapping or nonoverlapping subdomains $\Omega_i, i = 1, \ldots, N$. Let $Ax = b$ be the basic linear system of equations resulting from, for example, a finite element discretization of the PDE in Ω, A_i the matrix corresponding to the discretization in Ω_i, b_i the right-hand side, and R_i a matrix representing the algebraic restriction of a vector on Ω to the corresponding on Ω_i. Splitting up A_i, b_i and R_i into boundary and inner components, one obtains

$$A_i = \begin{pmatrix} A_i^{II} & A_i^{IB} \\ A_i^{BI} & A_i^{BB} \end{pmatrix}, \quad b_i = \begin{pmatrix} b_i^I \\ b_i^B \end{pmatrix}, \tag{2.1}$$

and

$$R_i = \begin{pmatrix} 0 & \ldots & 0 & I & 0 & \ldots & 0 & 0 \\ 0 & \ldots & & & & \ldots & 0 & R_i^B \end{pmatrix}. \tag{2.2}$$

Using these definitions, a decomposition of the basic system is given by

$$\sum_{i=1}^{N} R_i^T A_i R_i x = \sum_{i=1}^{N} R_i^T b_i. \tag{2.3}$$

Parallel Conjugate Gradient Methods

Conjugate gradient type methods can be parallelized as in [Mey90]. Assume the nonoverlapping domain decomposition with the representation of the basic system given by (2.3). Then two types of local vector representations have to be distinguished:

1. **Non-assembled:** At inner boundaries, a non-assembled vector only contains information on one subdomain and does not coincide with the global vector, e.g., b_i in (2.3).
2. **Assembled:** At inner boundaries an assembled vector coincides with the global vector, e.g., $x_s = R_s x$.

Iteration steps containing summation of two vectors may trivially be parallelized. For a scalar product $< t, r >$ holds

$$< t, r > = < \sum_i R_i^T t_i, r > = \sum_i < t_i, R_i r > = \sum_i < t_i, r_i > \quad ,$$

with the assembled vector r_i and the non-assembled vector t_i in domain i. So $< t, r >$ can be calculated by summing up the local scalar products $< t_i, r_i >$. The norm $\|r\|$

may be calculated in a similar manner. But for stability reasons, $\|r\|$ is calculated by

$$\|r\| = \sqrt{\sum_{i=1}^{N} \|r_i\|_i^2} \quad ; \quad \|r_i\|_i^2 = \sum_{j=1}^{n_i} \alpha_i^j (r_i^j)^2 \quad ,$$

with n_i denoting the number of variables in subdomain i. α_i^j are weights equal to 1 for inner variables and equal to 0 or 1 for variables on subdomain boundaries. For an inner boundary node, α_i^j is 1 for just one adjacent subdomain i_0 and 0 for $i \neq i_0$.

The parallel algorithm is not as stable as the serial one. In practical applications there are sometimes problems in achieving the required accuracy which may have their roots in the assemblies. During the iteration process of the parallel algorithm vectors are assembled with nonzero entries, but the assembled vector should be zero for the converged solution. In the serial program these problems do not occur because the assembly is done before solving the linear system.

Parallel Gauß-Seidel

Parallelization of Gauß-Seidel depends on the type of application, e.g., whether we have a scalar equation or a system of equations. Of course, the numbering of the variables, in particular those at the inner boundaries, plays an important role, also the coupling between variables, and whether we choose standard Gauß-Seidel or Block-Gauß-Seidel. In this section, one parallel variant of standard Gauß-Seidel used in point defect simulation is described with the numbering of variables given by (2.1),(2.2),(2.3).

With a decomposition of matrix A into an upper, lower and a diagonal matrix ($A = U + L + D$) one iteration of Gauß-Seidel reads:

$$Dx^{k+1} = (b - Lx^{k+1} - Ux^k). \tag{2.4}$$

Applying partition (2.3) of matrix A it holds:

$$\sum_{i=1}^{N} R_i^T D_i R_i x^{k+1} = \sum_{i=1}^{N} R_i^T (b_i - L_i R_i x^{k+1} - U_i R_i x^k). \tag{2.5}$$

Splitting up U_i, L_i, D_i, R_i and b_i into inner and boundary components as in (2.1),(2.2) leads to

$$D_s^{II} x_s^{I,k+1} = b_s^I - L_s^{II} x_s^{I,k+1} - (U_s^{II} x_s^{I,k} + U_s^{IB} x_s^{B,k}) \tag{2.6}$$

for the inner variables of domain s and

$$\sum_{i=1}^{N} R_i^{B,T} D_i^{BB} x_i^{B,k+1} = \sum_{i=1}^{N} R_i^{B,T} (b_i^B - (L_i^{BI} x_i^{I,k+1} + L_i^{BB} x_i^{B,k+1}) - U_i^{BB} x_i^{B,k}) \tag{2.7}$$

for the boundary nodes. (2.7) carries a lot of communication. For some applications it may be therefore advantageous to replace (2.7) by the Jacobi iteration. Applying operator R_s^B one arrives at

$$\begin{aligned}(D_s^{BB} + R_s^B \sum_{i \neq s} R_i^{B,T} D_i^{BB} R_i^B R_s^{B,T}) x_s^{B,k+1} &= R_s^B \sum_{i=1}^{N} R_i^{B,T} (b_i^B \\ &- (L_i^{BI} x_i^{I,k+1} + L_i^{BB} x_i^{B,k}) - U_i^{BB} x_i^{B,k}).\end{aligned}$$

3 Hardware Environment

In order to make an easier interpretation of the performance tables in the next section, a short description of the used hardware environment is given. The two applications run on a workstation cluster and on a multiprocessor-shared memory architecture. The workstation cluster consists of four Hewlett Packard HP725/75MHz and of 16 SUN4 sparc2 workstations which are connected by Ethernet. The shared memory architecture is a Silicon Graphics Power Challenge with eight processors (R8000/90MHz). To give an idea of the floating point performance of the three different architectures, the theoretical peak performance and the LINPACK benchmark are listed in Table 1.

Table 1 The first column is the theoretical peak performance in Mflops and the second is the rate achieved for the decomposition of a matrix of dimension 1000.

Computer	Peak	Matrix
SUN4 Sparc10/51 (50MHz)	50	27
Hewlett Packard 725 (75MHz)	150	92
SGI Power Chall. (90MHz) 1 Proz.	360	308

The communication could be performed by interface software packages such as MPI or PVM; see [MPI94, GBD+93, pvm]. On the workstation cluster the Send/Receive is realized by using UNIX sockets. The shared memory variant uses a simple copy mechanism. One of the major advantages of these interfaces is that the software can be transferred to the new platform without any modifications. Further acceleration on the multiprocessor architecture is achieved by using the machine-specific interfaces of SGI. The machine-specific interfaces directly exploit the shared memory structure and need no copy at all. For the current implementations PVM was used.

4 Applications

Point Defects

It is state-of-the-art to simulate point defects in order to obtain a consistent model for dopant diffusion; see [PJK+93]. Diffusion of dopant ions (boron, phosphorus arsenic and antimony) in silicon only takes place by way of interaction with either a silicon interstitial or a silicon vacancy. In the case of inert diffusion an equilibrium concentration of point defects can be assumed. Process steps like oxidation of silicon disturb this equilibrium concentration, thus the transient evolution of point defects has to be simulated. Following Hu [Hu74], point defects are modelled by two linear diffusion equations which are coupled by a nonlinear recombination term. The generation of interstitials depends on the interface velocity between silicon and silicon oxide:

$$\frac{\partial C_I}{\partial t} = \nabla \cdot (D_I \nabla C_I) - k(C_I C_V - C_I^* C_V^*), \quad C_I(0) = C_I^*, \tag{4.8}$$

$$\frac{\partial C_V}{\partial t} = \nabla \cdot (D_V \nabla C_V) - k(C_I C_V - C_I^* C_V^*), \quad C_V(0) = C_V^*, \tag{4.9}$$

and the boundary conditions

$$D_I \frac{\partial C_I}{\partial n} = \begin{cases} -K_I(C_I - C_I^*) + F(v_{int}) & \text{on } \Gamma \\ 0 & \text{otherwise,} \end{cases} \tag{4.10}$$

$$D_V \frac{\partial C_V}{\partial n} = \begin{cases} -K_V(C_V - C_V^*) & \text{on } \Gamma \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

C_I and C_V denote the concentrations of interstitials and vacancies, C_I^* and C_V^* the thermal equilibrium concentrations, D_I and D_V the diffusion coefficients, and K_I and K_V the surface recombination velocities of interstitials and vacancies. v_{int} is the velocity of the interface Γ between oxide and silicon. The computational domain is 2D.

For space discretization linear finite elements are used, time discretization is performed by the trapezoidal rule, backward difference method (TRBDF) [BCF+85]. In each time step two nonlinear systems have to be solved by Newton's method. Both linear systems are solved with parallel multigrid.

The multigrid cycles are optimized with regard to the serial algorithm (V-cycles start on coarse grid); see [Kah94]. On the coarsest grid, the *serial* CGS [Son89] is applied (The *parallel* CGS leads to a communication overhead). This means that all relevant data has to be sent to a specified host before starting serial CGS there. Table 2 shows the h-independence of the multigrid algorithm. The number of multigrid cycles is nearly a constant, while the number of variables increases.

With the use of PVM, the software was transferred to the SGI Power Challenge without any changes. The difference in performance with respect to the SUN4 processor is dramatic. Thus, time measurements make only sense for a large number of variables. Table 2 shows results of the parallel multigrid algorithm for the SUN4 workstation cluster and the SGI Power Challenge.

Table 2 Speed-up of the multigrid algorithm on a SUN4 workstation cluster (first row) and on the SGI Power Challenge (second row). The point defect equations are computed with 162 to 132098 variables. The first number in the Table gives the real time for one timestep (8 Newton iterations) in seconds; the second denotes the number of multigrid cycles.

#Variables	162	578	2178	8450	33282	132098
1 domain	4/16	9/20	31/20	111/19	428/18	
					25/18	119/17
4 domains	7/16	14/20	26/20	57/20	146/18	
					7/18	30/17
16 domains	22/21	21/21	30/21	49/20	101/18	

We have also considered a parallel CGS accelerated by an additive nonoverlapping Schwarz; see also [KPW96]. The local problems associated with the additive Schwarz method are solved by the serial multigrid algorithm. For the both architecture platforms considered, the parallel multigrid works faster.

Neutron Kinetics

One major task of reactor core simulation is the calculation of neutron fluxes given by the transient multigroup neutron diffusion equations [FG81]:

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t}(\mathbf{r},t) + \nabla \mathbf{j}_g(\mathbf{r},t) + (\Sigma_{ag} + \sum_{g'>g} \Sigma_{g'g})\phi_g(\mathbf{r},t) \tag{4.12}$$

$$= \sum_{g'<g} \Sigma_{gg'}\phi_{g'}(\mathbf{r},t) + \frac{1}{\lambda}\sum_{g'=1}^{G} \Sigma_{pgg'}\phi_{g'}(\mathbf{r},t)$$

$$+ \sum_{i=1}^{I} \chi_{dg}^{i}\lambda_i C_i(\mathbf{r},t) + S_g^{ext}(\mathbf{r},t) \ ,$$

$$\mathbf{j}_g(\mathbf{r},t) + D_g \nabla \phi_g(\mathbf{r},t) = 0 \tag{4.13}$$

$$\frac{\partial C_i}{\partial t}(\mathbf{r},t) + \lambda_i C_i(\mathbf{r},t) = \frac{1}{\lambda}\sum_{g'=1}^{G} \Sigma_{fg'}^{i}\phi_{g'} \tag{4.14}$$

in a 3D computational domain. ϕ_g and $\mathbf{j}_g$ are the flux and current in energy group g, v_g the neutron velocity, D_g the diffusion constant. χ_{dg}^i are fission spectra of delayed neutrons, S_g^{ext} is the external source, λ the eigenvalue of the reactor. $\Sigma_{ag}, \Sigma_{gg'}, \Sigma_{pgg'}$, $\Sigma_{fg'}^i$ are cross sections, C_i the precursor concentration of type i, λ_i the decay constant. The unknowns to be computed are $\phi_g, \mathbf{j}_g$ and C_i. At outer boundaries mixed boundary conditions (vacuum or albedo boundary conditions for a reflective medium) hold. Space discretization is done by a *nodal expansion* method (NEM) [FBW77, FG81]. The domain is subdivided into boxes, (4.12),(4.14) are integrated over the volume and (4.13) over the surfaces of each box. For time discretization implicit Euler is applied in combination with an exponential transformation technique [FG81]. The equations for precursor concentrations can be substituted into the balance equations for the fluxes. This results in the system of finest grid equations. On coarser grids a multiplicative variant of multigrid, CMR ("Coarse Mesh Rebalancing") [FBMK91], is applied. On grid levels 1 to $L-1$, the coarse mesh equations are solved by red-black Gauß-Seidel, whereas on coarsest grid level L Gaussian elimination is applied.

The basic principle of parallelization is axial domain decomposition. The axial layers of the core are distributed among the processes which solve the NEM equations on the finest grid level and the CMR equations on grid levels 2 to $L-1$.

The parallel neutron kinetics code is part of a coupled thermal hydraulics / neutron kinetics code system [KM94], the thermal hydraulics part running sequentially.

As test example an emergency power case has been chosen. The simulation is performed for three different problem sizes: the first problem with approximately 4000 boxes, the second one with 8000 and the third one with 16000 boxes on the finest grid. Table 3 shows results for a cluster of HP 725 workstations. The execution times are overall execution times including the sequential parts of the program.

In this simulation only the two finest grid levels were computed in parallel, with the number of parallel processes given in the table. The coarser grid levels were computed by a single master process.

The parallelization of the multilevel cycle leads to considerable communication between the processes. Thus much better speed-up factors are expected on a shared

Table 3 Results for the parallel neutron kinetics code on HP 725 workstations with PVM: execution time in seconds.

#Processes	Problem 1	Problem 2	Problem 3
1P	5185	8795	14533
2P	4916	6915	10568
4P	4736	6638	9476

memory computer. Table 4 (each of the first numbers) shows results for the parallel code system using again PVM as parallel platform, the neutron kinetics running on the SGI Power Challenge and the thermal hydraulics on an HP 725 workstation. The

Table 4 Results for the parallel neutron kinetics code on the SGI Power Challenge. The first number gives the execution time in seconds with PVM as parallel platform, the second number the execution time using machine-specific interfaces.

#Processes	Problem 1	Problem 2	Problem 3
1P	1529	2788	4624
2P	909/919	1402/1370	2314/2259
4P	651/626	844/ 851	1271/1277
6P	562/506	672/ 644	945/ 946
8P	776/511	851/ 629	964/ 791

basic overhead of parallelization is the same as in the case of workstation clusters, but a Send/Receive of PVM is realized by a simple copy mechanism and the more costly communication through UNIX sockets is avoided. The speed-up factors scale only up to 6 processors; for 8 processors the communication overhead becomes dominant. Further acceleration is achieved by using machine-specific interfaces of SGI instead of PVM which need no copy at all. The results are also shown in Table 4 (the second of each set of numbers).

5 Conclusion

Our goal was to study the scalability of parallel domain decomposition methods in an industrial environment. Three different computer architectures were used: cluster of HP and SUN workstations, and an SGI Power Challenge, a shared memory computer. For point defect analysis, the parallel multigrid with minor modifications turned out to be a very efficient method. It is somewhat surprising, that even on a workstation cluster with its low communication performance, the method works quite satisfactorily. In neutron kinetics simulation, future work concentrates on variants of multigrid with similar convergence rates, but with higher parallel potential for a large number of processors.

REFERENCES

[BCF+85] Bank R., Coughran W. M., Fichtner W., Grosse E. H., Rose D. J., and Smith R. K. (1985) Transient simulation of silicon devices and circuits. *IEEE Trans. Computer Aided Des.* 4(4): 436+.

[FBMK91] Finnemann H., Böer R., Müller R., and Kim Y. (1991) Multi-level techniques for the acceleration of nodal reactor calculations. In *Proc. Int. Top. Meeting Advances in Math. Computations and Reactor Physics*. Pittsburgh.

[FBW77] Finnemann H., Bennewitz F., and Wagner M. (1977) Interface current techniques for multidimensional reactor calculations. *Atomkernenergie* 33: 1123+.

[FG81] Finnemann H. and Grundlach W. (1981) Space-time kinetics code IQSBOX for PWR and BWR. Part I: Description of physical and thermo-hydraulic models. *Atomkernenergie-Kerntechnik* 37(3): 176+.

[GBD+93] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., and Sunderam V. (1993) *PVM 3.0 User's Guide and Reference Manual.* Oak Ridge National Laboratory, Tennessee.

[Hu74] Hu S. M. (1974) Formation of stacking faults and enhanced diffusion in the oxidation of silicon. *J.Appl.Phys.* 45: 1567+.

[Kah94] Kahlert M. (1994) Prozeßsimulation auf verteilten Rechnersystemen. Master's thesis, Mathematisches Institut der Technischen Universität München.

[KM94] Knoll A. and Müller R. (1994) Coupling RELAP5 and PANBOX2: A three-dimensional space-time kinetics application with RELAP5. In *Proceedings of Intern. Conference on New Trends in Nuclear System Thermohydraulics*. Pisa.

[KPW96] Kahlert M., Paffrath M., and Wever U. (1996) Grid partitioning versus domain decomposition: A comparison of some industrial problems on workstation clusters. *Surv. Math. Ind.* 6: 133+.

[Mey90] Meyer A. (1990) A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. *Computing* 45(217+).

[MPI94] (1994) *Message Passing Interface Forum. MPI: A message-passing interface standard.*

[PJK+93] Paffrath M., Jacobs W., Klein W., Rank E., Steger K., Weinert U., and Wever U. (1993) Concepts and algorithms in process simulation. *Surv. Math. Ind.* 3: 149+.

[pvm] PVM home page in the World Wide Web. World Wide Web, , http://www.epm.oml.gov/pvm/pvm_home.html.

[Son89] Sonneveld P. (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* 10(36+).