KfK 4614
Oktober 1989

Concepts for Efficient Implementation of Multigrid Methods on SUPRENUM-Like Architectures

M. Alef
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Datenverarbeitung in der Technik

KfK 4614

Concepts for Efficient Implementation of Multigrid Methods on SUPRENUM-Like Architectures

Manfred Alef

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

The work desribed in this report was funded by the Federal Ministry for Research and Technology (BMFT), Fed. Rep. Germany, under the grant number ITR8502K/4.
The author assumes all responsibility for the contents of this report.

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Konzepte zur effizienten Implementierung von Mehrgittermethoden auf SUPRENUM-ähnlichen Architekturen

Zusammenfassung:

Bei der Implementierung von Algorithmen auf Parallelrechnern mit verteiltem Speicher ist es erforderlich, daß die parallelen Prozesse regelmäßig gewisse Zwischenergebnisse untereinander austauschen. Die Parallelisierung ist umso effizienter, je weniger Datenaustausch durchzuführen ist.

In diesem Bericht werden am Beispiel einer Implementierung für SUPRENUM verschiedene Konzepte vorgestellt, wie bei der Parallelisierung von Mehrgittermethoden der Kommunikationsaufwand gesenkt werden kann.

Concepts for Efficient Implementation of Multigrid Methods on SUPRENUM-Like Architectures

Summary:

The implementation of algorithms on distributed-memory multiprocessors requires regular exchange of certain provisional results between the parallel processes. The less data have to be moved the more efficient is the parallelization.

In this report some concepts are presented, how the communication overhead can be reduced when multigrid methods are parallelized on SUPRENUM.

Contents:

1. Introduction

The numerical simulation of many technical and scientific processes gains increasing importance. An example is the development of pulsed power ion diodes generating focused light-ion beams with high particle energy. The physical problem is to find a geometry in which the ion beams are optimally focused.

In order to obtain a better understanding of the underlying physical phenomena and to optimize the geometry of rotationally symmetric diodes, the two-dimensional quasistationary code BFCPIC - a particle-in-cell (PIC) code based on boundary-fitted coordinates (BFC) with a logical rectangular structure - has been developed [1,2].

Because the simulations require extremely large computation times, the most time consuming modules have been parallelized and prepared for the implementation on the distributed-memory multiprocessor SUPRENUM [3 - 5]. (See [6 - 9] for more detailed informations about the hardware of the SUPRENUM computer system, [8,10,11] for application software on SUPRENUM.)

One of the main components of this code is the computation of the potential of the electrostatic fields in the diode. For that purpose the Poisson equation is discretized in the boundary-fitted grid by an 9-point approximation and solved using multigrid methods. This module is performed thousands of times during a single run of the code [12].

In this report, different possibilities for implementation of this module on SUPRENUM are examined. In principle, these ideas are (with certain changes) valid also for other (2D or 3D) multigrid methods as well as for implementations on similar distributed-memory multiprocessors.

2. Problem

The potential of an electrostatic potential is determined by the Poisson equation

$$\Delta\Phi = -\rho/\varepsilon$$

(ρ: charge density, ε: dielectric constant).

Because of the rotationally symmetric description of the diodes (no θ-dependency) cylindrical coordinates are introduced, and Φ can be computed in the two dimensional (r,z)-space.

In order to obtain a proper description of the boundary conditions, the computations are performed in a boundary-fitted grid with a logical rectangular structure [12 - 16]. This results in the necessity of an operator with at least 9 points for the discretization of the Poisson equation. The resulting problem is solved using multigrid methods [17 - 19; 10 - 12].

In this report, only the concepts underlying the implementation on SUPRENUM are investigated. For a more detailed discussion of the numerical background see [12].

3. Performance Model

Every computation on a distributed-memory multiprocessor requires certain additional activities, compared with a corresponding scalar run: at least, the input data must be distributed among all parallel processes, and the results must be gathered. Further communication is necessary, if a process needs provisional results from another one. In this case, a message containing these data is exchanged between the two processes.

The efficiency of a computation using several computing nodes is determined by this communication as well as by the load-balancing of the parallel processes.

The performance model used for all estimations in this report is based on the one introduced by O. Kolp and H. Mierendorff [20, 21]. The idea is to look for the amount of work for the following system components:

1. Arithmetic operations (T_A)
2. Message transfer (communication) ($T_C = T_{CN} + T_{CC} + T_{CS}$)
   composed of the following parts:
   - Initialization of the message by the computing node (T_{CN})
   - Message transfer through the clusterbus (T_{CC})
   - If the message is sent to another cluster of nodes, the additional time needed by the SUPRENUMbus and the communication nodes of the source and destination cluster (T_{CS})

Each element T_i is summarized over a complete multigrid cycle. It consists of a time component $T_{i,1}$ needed for a single vector or message element, multiplied by the number and the lengths of the vectors or messages, and of a time amount $T_{i,2}$ needed for the initialization of the vector instructions or messages (start-up times). All T_i have been computed using a simplified model of the control section of the original program.

The following data have been used for the performance estimations:

- The number of vector instructions per grid line for the main multigrid components, relaxation, fine-to-coarse grid transfer (restriction), and coarse-to-fine grid transfer (interpolation) (table 3.1). Because the first relaxation sweep after each restriction will start with zeros as an initial guess, it requires less work.
- The number of messages during one of the above multigrid components (table 4.2.4, see section 4.2).
- The number of relaxation sweeps, restrictions or interpolations per grid level during one V-, W- or F-cycle (table 3.2) [12].
- The time needed for one vector or message element (independent of the length of the corresponding vector or message) and the start-up times (table 3.3 [21]).

	Relaxation		Restriction	Inter-polation
	after restr.	rest		
Vector instructions per grid line of the stripe assigned to the considered process *) treating just every second grid point	12.25 *)	22.00 *)	19.50	3.25

Table 3.1: Number of vector instructions per grid line during one relaxation sweep, restriction or interpolation; data for restriction and interpolation are per grid line of the finer grid

Number of ...	V-cycle	W-cycle	F-cycle
Relaxation sweeps immediately after restriction	1	2^{i-2}	$i-1$
Other relaxations (finest grid)	γ	$\gamma \cdot 2^{i-1}$	$\gamma \cdot i$
Other relaxations (coarse grids)	$\gamma-1$	$\gamma \cdot 2^{i-1}-2^{i-2}$	$\gamma \cdot i-(i-1)$
Restrictions and interpolations	1	2^{i-1}	i

Table 3.2: Number of relaxation sweeps, restrictions and interpolations per grid level i of a V-, W- or F-cycle of the multigrid method; γ is the sum of the number of relaxation sweeps before and after each coarse-grid treatment

Time per ...	T_A	T_{CN}	T_{CC}	T_{CS}
Vector or message element	0.1	0.6	0.031	0.8
Vector or message, independent of its length (start-up time)	2.25	600	1.25	60

Table 3.3: Times reqired for one unit of vector instructions or messages (in µs)

It will be assumed that all vector instructions and messages are performed sequentially within each parallel process. In this (worst) case the total time T needed for one V-, W- or F-cycle of the multigrid method is the sum

$$T = T_A + T_C = T_A + T_{CN} + T_{CC} + T_{CS}.$$

The amount of work necessary for I/O, creation and termination of the processes, and so on will not be investigated in this report. This work can be neglected for sufficiently large problems.

Let T(N) be the time needed by a certain algorithm using N computing nodes. Then S(N) = T(1) / T(N) is the **speedup** and E(N) = S(N) / N is the **efficiency** of the parallel computation.

For simplicity, the following assumptions are made:

- The number of grid points N_X, N_Y is equal in both directions, and $N_X = N_Y = 2^n + 1$ with an integer number n.
- The number of parallel processes equals the number N of nodes and is a power of 2, and the computational grid is mapped onto the processes by splitting it into subgrids with the number of grid points as equal as possible.
- All boundary conditions are of Dirichlet type.
- Triads are considered to be one vector instruction.
- The vector start-up time is assumed to be equal to the time for arithmetic operations of about 20 - 25 vector elements.

4. Parallelization Strategies

4.1 Domain Splitting

The parallelization is done by splitting the grid into subgrids which are assigned to the parallel processes, with the respective CPU-times as equal as possible.

Essentially, there are two alternatives:

- division into squares or square-like rectangles (fig. 4.1.1), which means that splitting is in both directions, or
- partitioning into stripes, i.e. in one direction only (fig. 4.1.2).

After each relaxation sweep and during the fine-to-course grid transfer (restriction using "full weighting"), the processes have to exchange certain results with the neighboring processes in the vicinity of their boundaries (see the textured areas in figures 4.1.1 and 4.1.2). The efficiency of the parallelization is essentially influenced by the communication overhead, compared with the arithmetic work.

There are considerable differences between the two cases in the amount of work required for this communication:

The advantages of the first strategy are that the message lengths due to the small boundary sizes are shorter, and that the load-balance of the processes especially during the treatment of the coarser grids is a little more favorable than in the other case. On the other hand, splitting into stripes minimizes the number of communications because each process has a maximum number of only two instead of eight neighbors. Furthermore, the vector lengths are a multiple of the vector lengths in the first case. Which of the two strategies is favorable depends upon the parameters of the system, especially whether the nodes of the machine are provided with vector processors, the start-up times, and the speed of the buses.

On the basis of the parameters of the SUPRENUM system and the performance model described in chapter 3 there is an obvious advantage for the second strategy, as long as the size of the problem is not extremely large - but in that case both, start-up as well as message transfer times, can be neglected compared with the amount of arithmetic work.

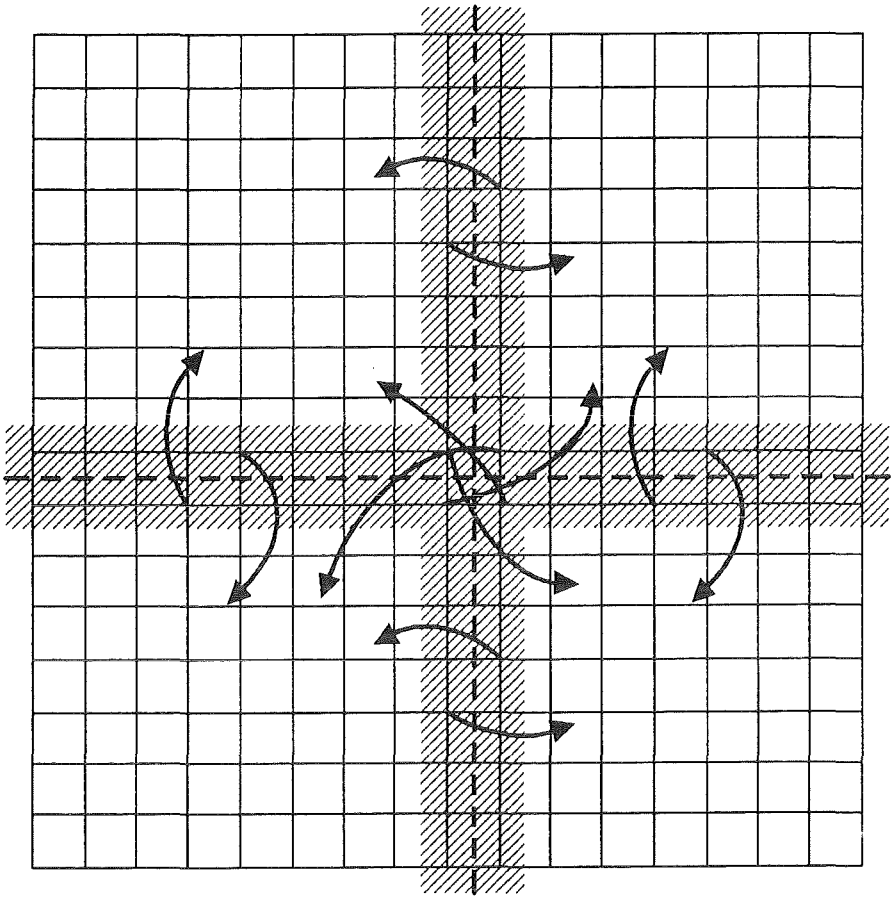

Fig. 4.1.1:

Domain splitting in both directions:

Every process has to exchange results with up to 8 neighbors for each update

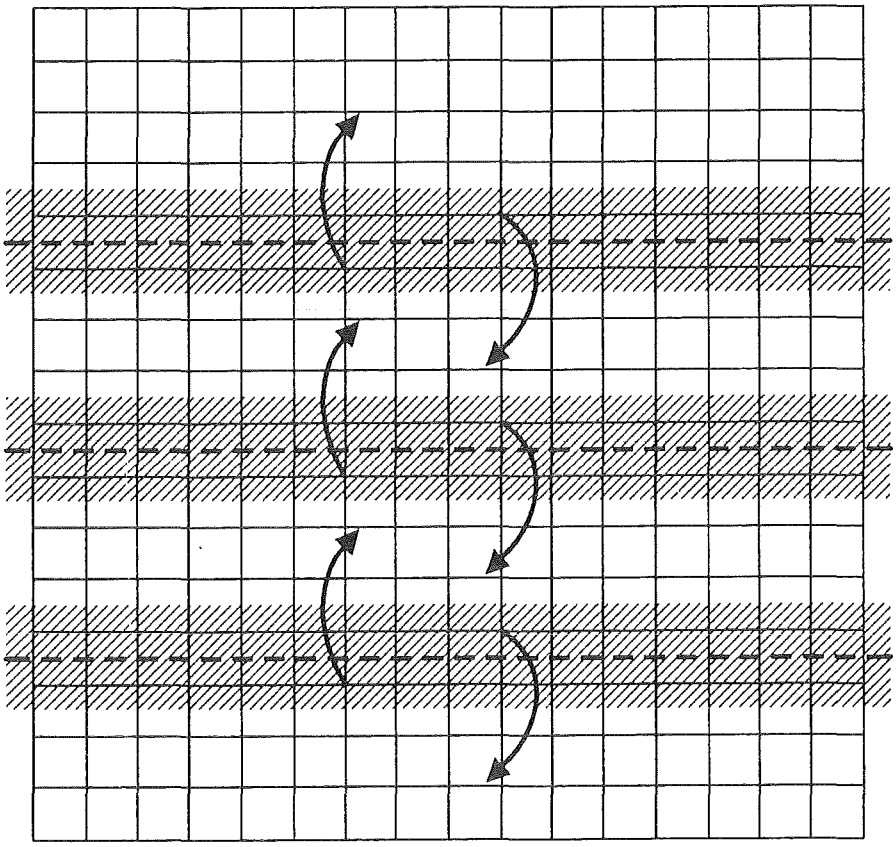

Fig. 4.1.2:

Domain splitting in one direction only:

There is communication with only up to 2 neighbors

(In both figures arrows show communication paths)

4.2 Multi-Color Relaxation Schemes

The standard relaxation method (also on scalar computers) for 9-point 2D-difference operators is the so-called "four-color" relaxation scheme. The grid is divided into four regular subgrids "A" - "D" (figure 4.2.1), and one full relaxation sweep consists of four partial steps, each of them treating exactly the points of one subgrid ("color"). The advantage of this scheme is that each "color" can be treated in parallel because the 9-point operator requires only data of the other 3 colors (see fig. 4.2.1).

The most interesting question is now: What is (are) the best sequence(s) to perform this four steps, out of the 4! = 24 possibilities?

One strategy is shown in figure 4.2.2: First, subgrid "A" is treated, then "D", "B", and finally "C". But this method has a serious disadvantage: if it is parallelized in the manner described in section 4.1, and there are two parallel processes named P_A (above) and P_B (below) treating the subdomain above or below the dashed line in figure 4.2.2, the following communication is necessary between these two processes: After the first partial relaxation step, P_B has to send a message of

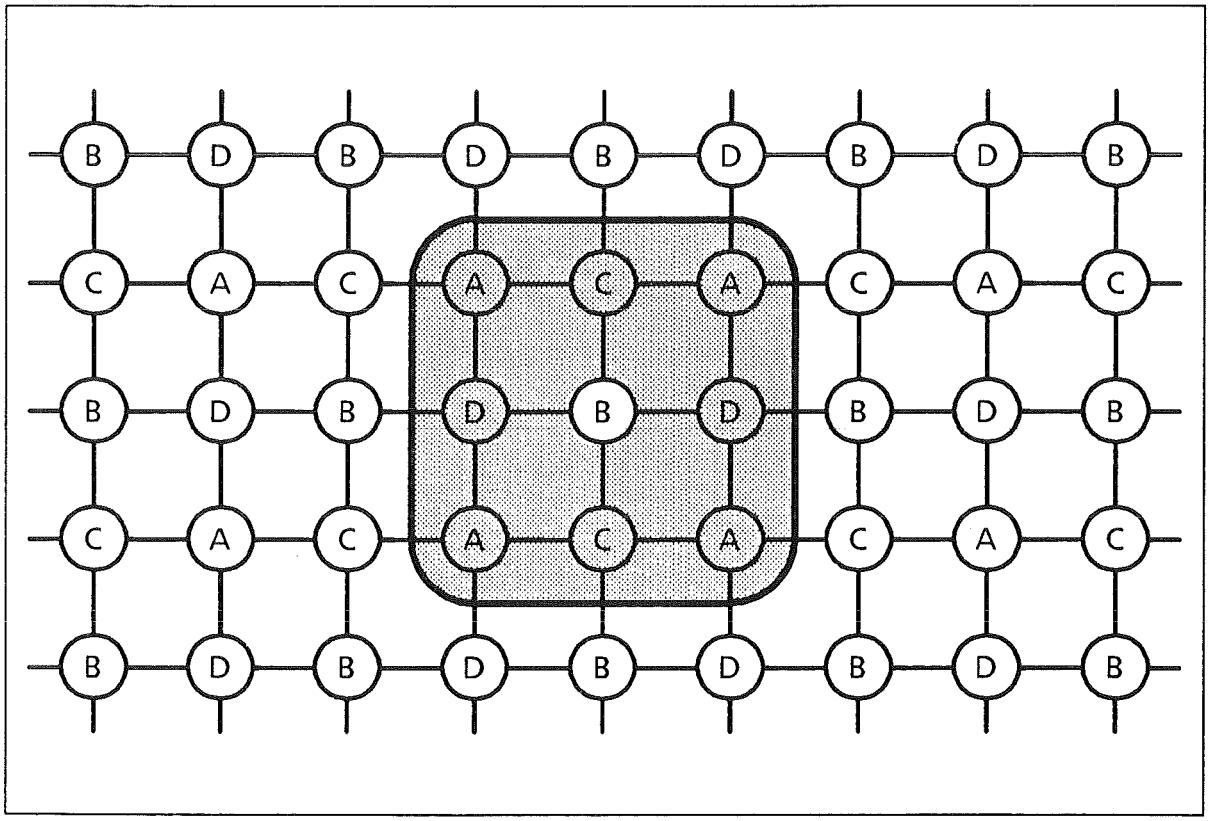

Fig. 4.2.1: Splitting of the grid into four "colors"

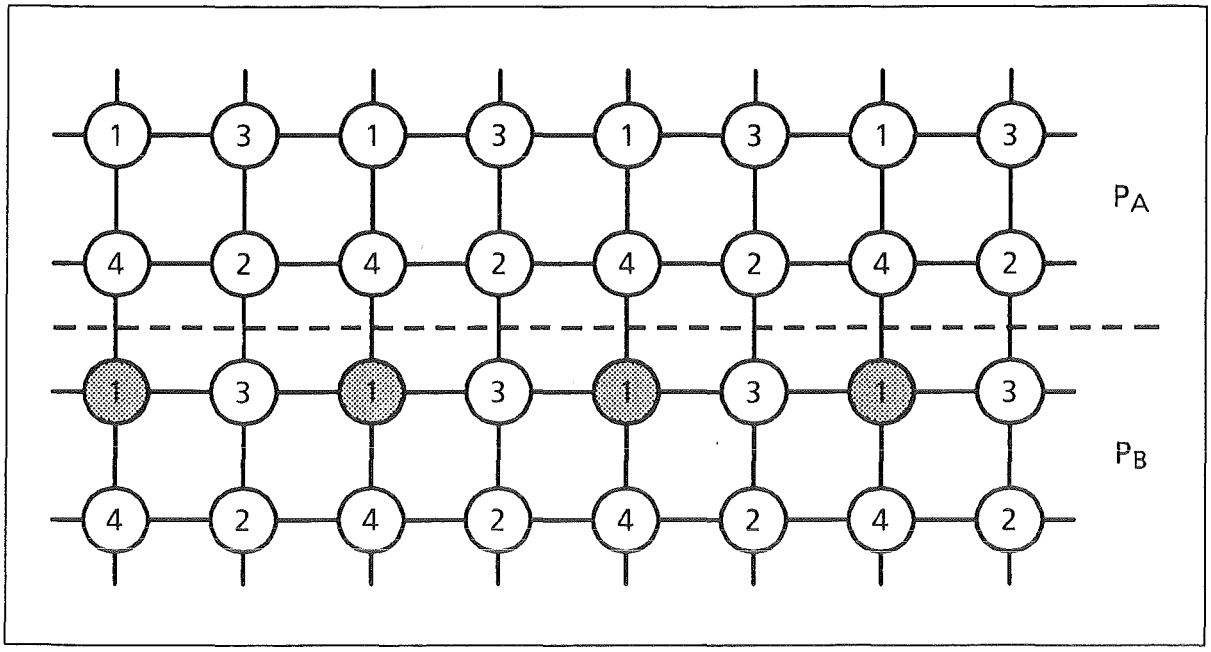

Fig. 4.2.2: Standard "four-color" relaxation scheme

length $N_X/2$ to P_A containing the recomputed results in the grid points "1" marked by darkened circles, where N_X is the number of grid points in this direction. P_A has to wait for this data required for the following computations in points "2". After the second step, P_A sends $N_X/2$ data to P_B, and so on. Until the full relaxation sweep has finished, there have been 2 single messages of length $N_X/2$ from P_A to P_B and another 2 of the same length in the opposite direction.

Another possibility is shown in figure 4.2.3, which differs only slightly from the first one. But now, there is only one message from P_B to P_A with N_X data elements after the second and another one back after the fourth relaxation step. (This is valid also for certain variations: either one of the "colors" 1 and 2 or 3 and 4 may be changed.)

Compared with the first strategy, the total number of data to be transferred is exactly the same - but this strategy economizes two start-up times. This is a big advantage especially if the message lengths are very short (when treating coarse grids of the multigrid method). In this case, the cost of a single message start-up may be a multiple of the cost for data transfer. As can be seen later, this start-up costs can outweigh the other components (computations, start-ups of the vector pipe, message transfer), particularly if W-cycles are performed.

The numerical properties like the convergence rates of the multigrid method only differ slightly. But one has to pay attention to the weighting operator used for the fine-to-coarse grid transfer of residuals (restriction): No problems will appear,

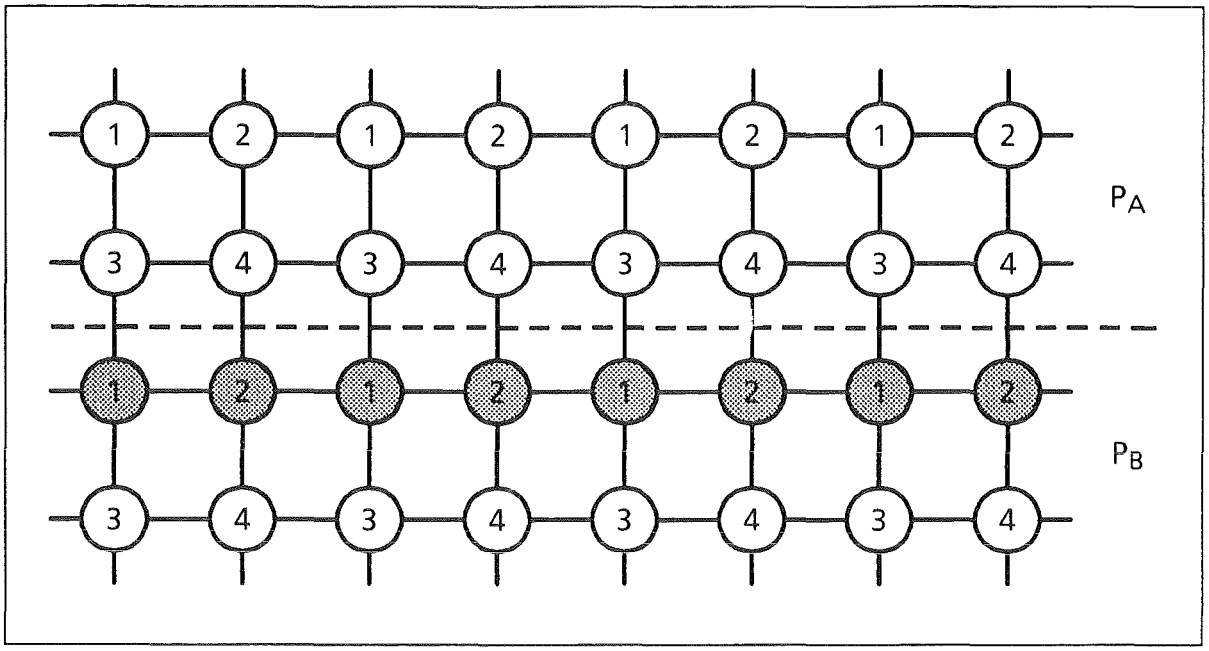

Fig. 4.2.3: Optimized "four-color" relaxation scheme

when using the "full weighting" operator because it takes the complete fine-grid residuals into consideration, and therefore the result will be an optimal average of them. However, the "half weighting" operator has to be modified according to the three colors included.

The data important for the performance estimations are listed in tables 3.1 - 3.3 (see chapter 3) and table 4.2.4.

The resulting efficiency rates are shown in tables 4.2.5 - 4.2.6: above all, there is a high increase of efficiency if W-cycles are performed. Also for V- and F-cycles an evident speed-up can be achieved if the problem size is not "too large".

	Relaxation		Restriction	Inter-polation
	after restr.	others		
Messages per boundary surface of the process (standard relax. scheme)	2.00	2.00	1.00	—
Messages per boundary surface of the process (optimized relaxation)	1.00	1.00	1.00	—

Table 4.2.4: Number of messages per parallel process (if the process contains just one grid line, there are additional messages, see section 4.3)

NX	V-cycle		W-cycle		F-cycle	
	standard relaxation	optimized relaxation	standard relaxation	optimized relaxation	standard relaxation	optimized relaxation
33	3.6	6.2	1.0	1.8	2.2	3.9
65	12.3	20.4	1.9	3.5	5.4	9.3
129	23.5	36.3	2.6	4.6	9.8	16.3
257	44.3	59.8	3.8	6.8	19.7	30.6
513	70.5	81.7	6.2	10.9	39.4	53.8
1025	88.7	93.4	10.7	17.9	65.9	77.4
2049	96.3	97.8	18.3	29.1	85.9	91.4
4097	98.8	99.3	3.02	44.2	95.2	97.1
8193	99.6	99.7	45.9	60.8	98.5	99.1
16385	99.9	99.9	62.7	75.4	99.5	99.7
32769	99.9	100.0	77.0	85.9	99.8	99.9
65537	100.0	100.0	86.9	92.4	99.9	100.0

Tab. 4.2.5: Efficiencies estimated for computations using one cluster (16 nodes); left column: standard, right: optimized (4-color) relaxation scheme

NX	V-cycle		W-cycle		F-cycle	
	standard relaxation	optimized relaxation	standard relaxation	optimized relaxation	standard relaxation	optimized relaxation
33	0.2	0.5	0.0	0.1	0.1	0.2
65	0.6	1.1	0.1	0.2	0.3	0.5
129	1.3	2.1	0.1	0.3	0.5	0.9
257	2.9	4.7	0.2	0.4	1.1	1.9
513	7.8	11.9	0.4	0.6	2.7	4.5
1025	21.2	29.5	0.6	1.1	7.5	11.8
2049	46.5	56.0	1.2	2.1	19.9	28.7
4097	72.8	78.4	2.3	4.0	44.0	55.1
8193	88.3	90.4	4.3	7.5	71.1	78.5
16385	95.1	95.7	8.2	13.9	88.1	91.2
32769	97.9	98.0	15.1	24.3	95.5	96.5
65537	99.0	99.1	26.2	38.9	98.2	98.5

Tab. 4.2.6: Efficiencies estimated for computations using 16 clusters (256 nodes)

4.3 Agglomeration of Processes

During the treatment of the coarsest levels of multigrid methods the communication cost may outweigh the arithmetic work because the number of messages necessary will not decrease as the number of grid points does. Therefore, it may be economical to accumulate the coarse subgrids of some or all processes by having only one remaining process treating the complete coarsest grid(s) (agglomeration). The advantage lies in the fact that in this level the communication overhead can be totally avoided. But on the other hand, the parallelization speed-up will decrease because of the poor load balancing.

The problem is to find an agglomeration scheme minimizing the communication cost with a minimum of additional work.

Basic Agglomeration Technique

The multi-color relaxation scheme discussed in section 4.2 has one characteristic property: if the stripes assigned to the parallel processes contain just one grid line, the following will happen (figure 4.3.1):

During the first two partial relaxation steps, only every second process will be busy. After they have finished, they must send their results simultaneously to both neighbors who are waiting for these data needed for the completion of the relaxation sweep. Also the restriction requires two messages instead of only one.

Combining the computations of all pairs of neighboring processes into only one remaining process (figure 4.3.2) will result in no more computation time because the remaining processes perform their additional computations in exactly the same time they would have to wait for results in the other case. But one of the two messages per relaxation sweep and per restriction can be economized!

On the other hand, there are two additional messages: first, the stopping process has to send its right hand side to the remaining one and finally receive the result. But because a multigrid cycle consists of at least two relaxation sweeps on each level, there will be a cut of at least one message.

Tables 4.3.3 - 4.3.4 show the estimated performance data in the case $\gamma = 2$; the efficiencies are improved for all cycle types.

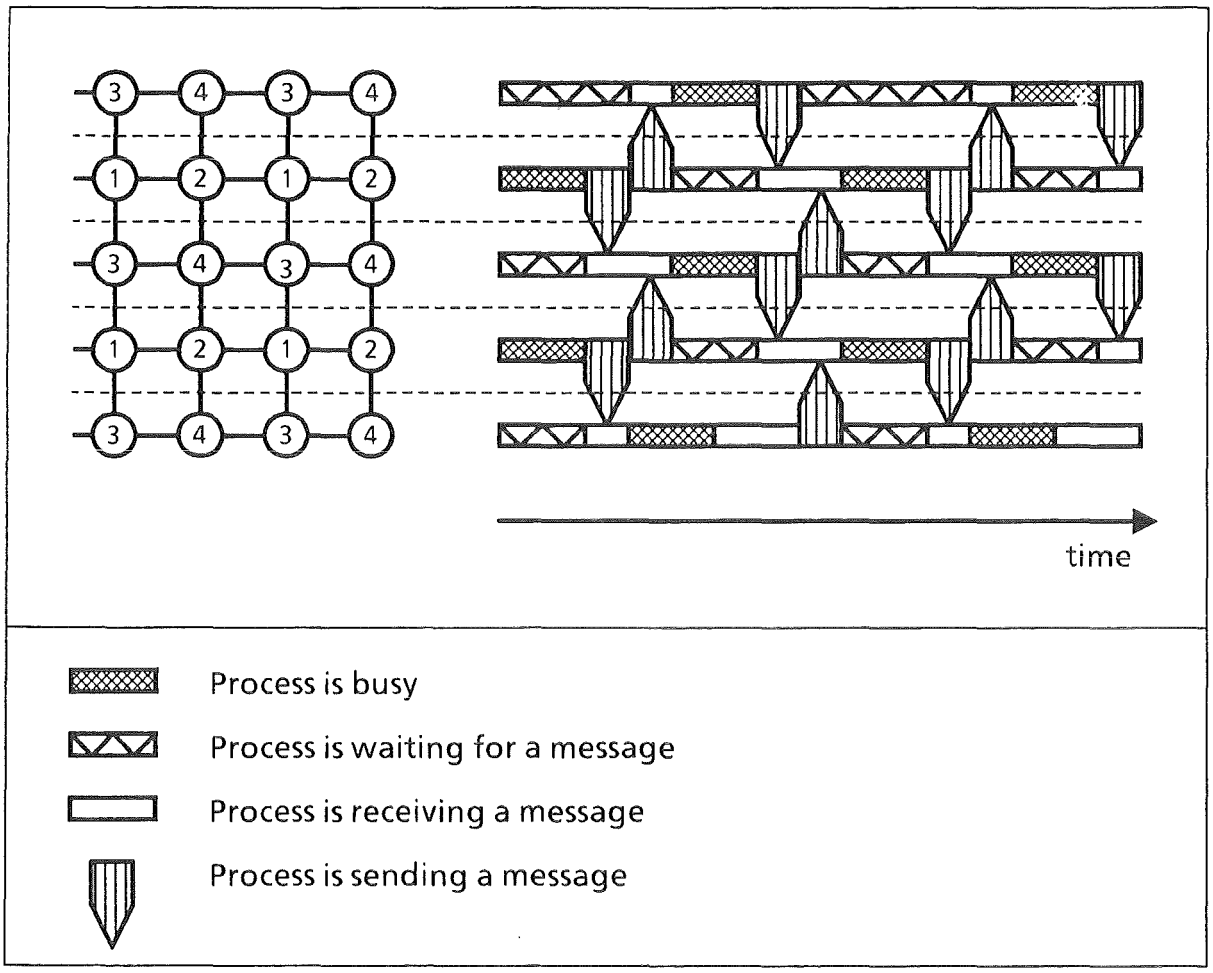

Fig. 4.3.1: Multiple communication and bad load-balancing, if the stripes contain only one grid line

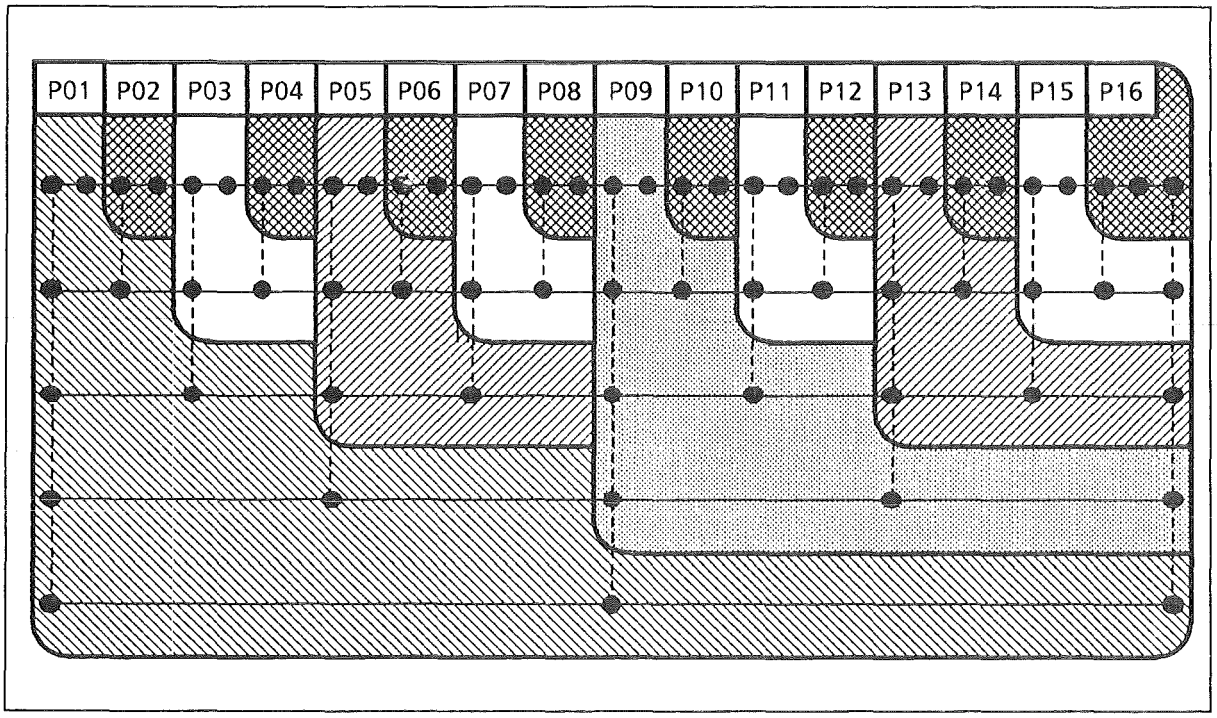

Fig. 4.3.2: Basic agglomeration technique - each black circle stands for a grid line

NX	V-cycle		W-cycle		F-cycle	
	without agglom.	basic agglom.	without agglom.	basic agglom.	without agglom.	basic agglom.
33	6.2	7.7	1.8	2.7	3.9	5.4
65	20.4	24.3	3.5	5.1	9.3	12.1
129	36.3	41.1	4.6	6.7	16.3	20.4
257	59.8	64.2	6.8	9.7	30.6	36.2
513	81.7	84.0	10.9	15.2	53.8	59.5
1025	93.4	94.3	17.9	24.3	77.4	80.9
2049	97.8	98.1	29.1	37.6	91.4	92.8
4097	99.3	99.3	44.2	53.7	97.1	97.6
8193	99.7	99.7	60.8	69.5	99.1	99.2
16385	99.9	99.9	75.4	81.8	99.7	99.7
32769	100.0	100.0	85.9	89.9	99.9	99.9
65537	100.0	100.0	92.4	94.7	100.0	100.0

Tab. 4.3.3: Efficiencies estimated for computations in one cluster (16 nodes); left column: without, right: with basic agglomeration technique

NX	V-cycle		W-cycle		F-cycle	
	without agglom.	basic agglom.	without agglom.	basic agglom.	without agglom.	basic agglom.
33	0.5	0.8	0.1	0.2	0.2	0.4
65	1.1	1.4	0.2	0.3	0.5	0.7
129	2.1	2.6	0.3	0.4	0.9	1.2
257	4.7	5.8	0.4	0.6	1.9	2.5
513	11.9	14.4	0.6	1.0	4.5	6.1
1025	29.5	33.5	1.1	1.7	11.8	15.2
2049	56.0	59.6	2.1	3.1	28.7	34.5
4097	78.4	80.1	4.0	5.9	55.1	60.8
8193	90.4	91.0	7.5	10.9	78.5	81.6
16385	95.7	95.9	13.9	19.4	91.2	92.3
32769	98.0	98.1	24.3	32.4	96.5	96.8
65537	99.1	99.1	38.9	48.8	98.5	98.6

Tab. 4.3.4: Efficiencies estimated for computations using 16 clusters (256 nodes)

Improved Agglomeration Scheme

The basic agglomeration technique desribed above only prevents that the communication overhead increases if there are processes treating a single grid line.

But of course, there remains a rest of communication. On certain coarse grid levels this communication can add up to a multiple of the arithmetic work. Therefore, an additional agglomeration may be suitable combining the computations - in one or several steps - in only one remaining node.

The performance estimations have shown that the best agglomeration strategy maps the two or three coarsest grids into one computing node and splits them into four processes treating the next finer grid, and so on, until a level containing just two grid lines is reached (figure 4.3.5). Then the agglomeration may continue using the basic technique.

The estimated performance data are listed in tables 4.3.6 - 4.3.7. As can be seen from these tables, a further improvement of the efficiencies can be achieved for all types of multigrid cycles.

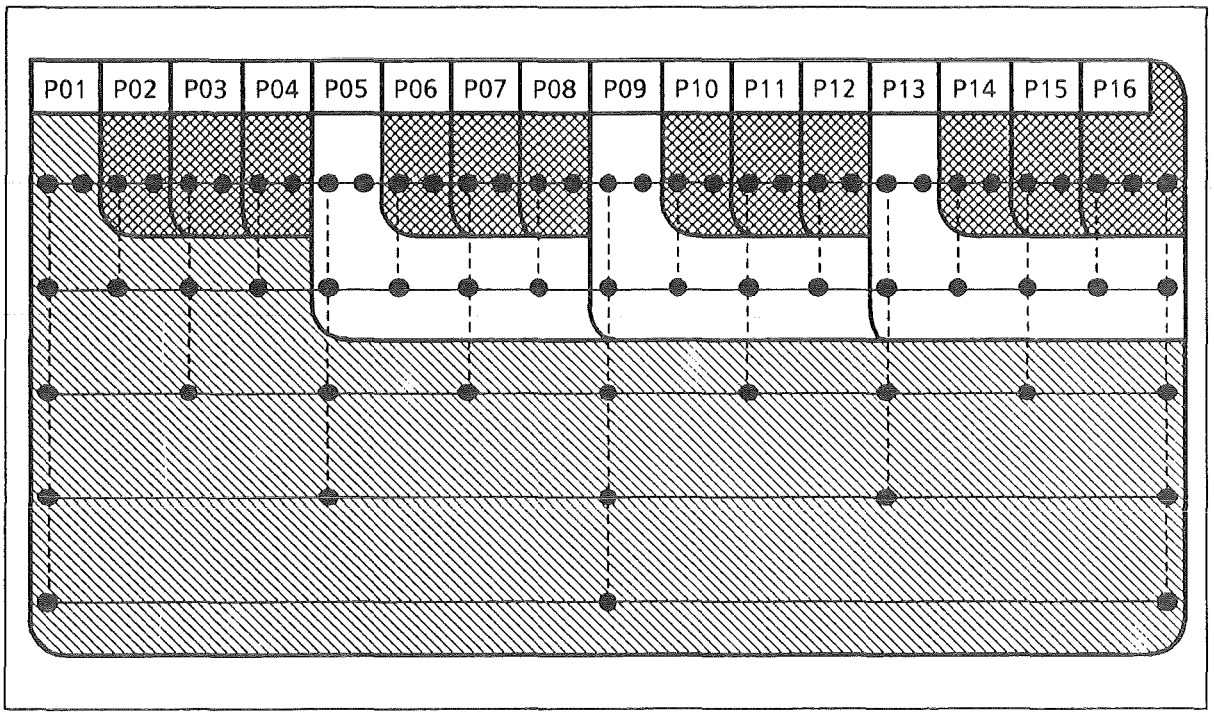

Fig. 4.3.5: Improved agglomeration scheme (principle)

NX	V-cycle		W-cycle		F-cycle	
	basic agglom.	optimized agglom.	basic agglom.	optimized agglom.	basic agglom.	optimized agglom.
33	7.7	10.3	2.7	6.5	5.4	9.0
65	24.3	30.5	5.1	11.9	12.1	18.3
129	41.1	47.9	6.7	15.1	20.4	28.1
257	64.2	69.6	9.7	20.8	36.2	45.2
513	84.0	86.7	15.2	30.3	59.5	67.2
1025	94.3	95.2	24.3	43.6	80.9	85.1
2049	98.1	98.3	37.6	59.1	92.8	94.4
4097	99.3	99.4	53.7	73.6	97.6	98.0
8193	99.7	99.8	69.5	84.5	99.2	99.3
16385	99.9	99.9	81.8	91.5	99.7	99.8
32769	100.0	100.0	89.9	95.5	99.9	99.9
65537	100.0	100.0	94.7	97.7	100.0	100.0

Tab. 4.3.6: Efficiencies estimated for computations in a single cluster (16 nodes); left column: basic, right: optimized agglomeration scheme

NX	V-cycle		W-cycle		F-cycle	
	basic agglom.	optimized agglom.	basic agglom.	optimized agglom.	basic agglom.	optimized agglom.
33	0.8	0.8	0.2	0.5	0.4	0.7
65	1.4	1.7	0.3	0.7	0.7	1.1
129	2.6	3.1	0.4	0.9	1.2	1.7
257	5.8	6.7	0.6	1.3	2.5	3.4
513	14.4	16.0	1.0	2.2	6.1	7.7
1025	33.5	35.7	1.7	3.8	15.2	18.2
2049	59.6	61.3	3.1	6.8	34.5	38.8
4097	80.1	80.9	5.9	12.4	60.8	64.4
8193	91.0	91.2	10.9	21.6	81.6	83.4
16385	95.9	96.0	19.4	35.2	92.3	92.9
32769	98.1	98.1	32.4	51.8	96.8	97.0
65537	99.1	99.1	48.8	68.0	98.6	98.7

Tab. 4.3.7: Efficiencies estimated for computations in 16 clusters (256 nodes)

General Remarks about Agglomeration Schemes

In both agglomeration techniques described above, there are two fundamental possibilities:

- Mapping the coarsest levels into just one remaining process, while the other processes are idle.
- Solving the coarsest-level problems by all processes at the same time, each of them treating the whole coarse grids.

There are no considerable differences between these two strategies with respect to the communication overhead: In the first case, the right hand sides must be collected by only one process, and by all processes in the other case. This will result in a disadvantage for the second strategy. But on the other hand, the first case requires the distribution of the result among all processes, so that altogether there will be nearly the same amount of work in both cases.

(It should be mentioned that the agglomeration technique is supported by the communication library available for SUPRENUM [22].)

4.4 Other Possibilities

Overlapping of Computations and Communications

The efficiency of the parallel computation might be improved a bit, if first of all the computations in the vicinity of the boundaries are performed. After these data have been sent to the neighboring processes, the computations continue in the interior region.

The increase of performance resulting from this procedure has not be investigated.

Using Larger Overlap Areas

It may be useful to assign a larger subdomain to each process than described in section 4.1. The idea is that more than one relaxation step can be performed before any exchange of data becomes necessary [10]. But the performance estimations showed that the multi-color relaxation scheme (see section 4.2) is the more favorable way to achieve this.

The disadvantages of the larger overlap areas are:

- Due to the larger number of grid points there are additional arithmetic operations, and the messages being exchanged are longer.
- The right-hand sides in the overlap areas also have to be exchanged after each restriction.

Furthermore, a combination of both techniques is not profitable.

4.5 Results of Performance Estimations

The estimations discussed in this chapter show that the efficiency can be improved by the optimized parallelization strategies. One has to distinguish between the different types of multigrid cycles:

The efficiency of V- and F-cycles (figures 4.5.1 and 4.5.3) can be improved by an increment of about 10 - 30 % until the peak rate of 100 % is reached.

When using W-cycles, a much higher increase can be obtained (figure 4.5.2). In this case, the rates of efficiency are up to 40 % better.

The efficiency of the optimized algorithm equals that of the standard one applied to larger problems, or to the same problem size using more computing nodes.

Figures 4.5.1 - 4.5.3 show the estimated differences between standard (standard relaxation as shown in figure 4.2.3, no agglomeration; dashed lines) and optimized algorithms.

The time components are shown in figures 4.5.4 (V-cycles), 4.5.5 (W-cycles), and 4.5.6 (F-cycles). The efficiency of the standard algorithm is determined by the portion of the nodes of the message start-up times ($T_{CN,2}$). In contrast, the amount of arithmetic operations ($T_{A,1}$, $T_{A,2}$) and of element-wise message transfer (especially $T_{CN,1}$ and $T_{CS,1}$) is increased in the optimized version, because more data have to be transferred due to the agglomeration.

In figure 4.5.7 the resulting MFLOPS-rates (million floating-point operations per second) are presented.

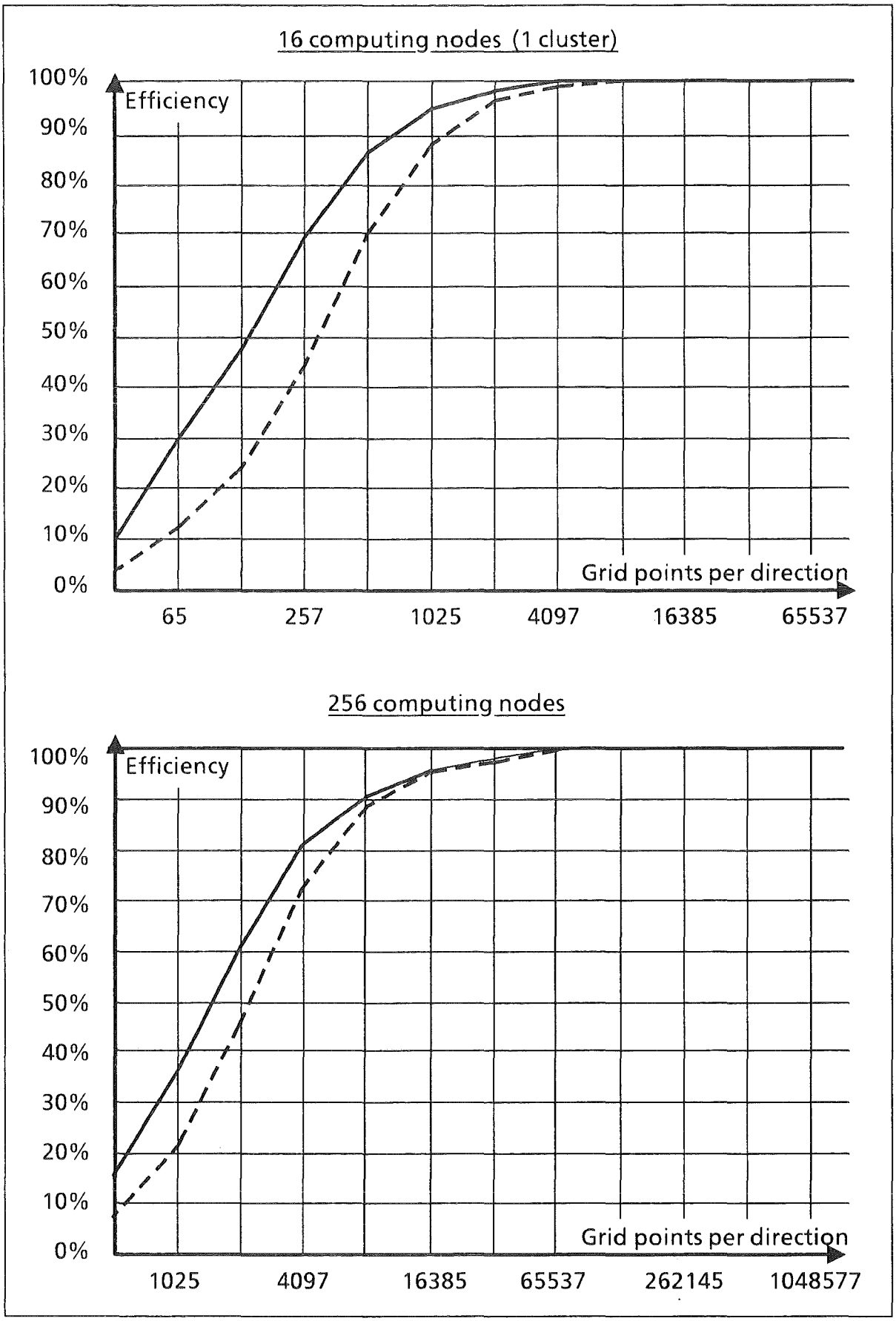

Fig. 4.5.1: Performance increase by optimized parallelization techniques (efficiencies for V-cycles using 16 or 256 computing nodes)

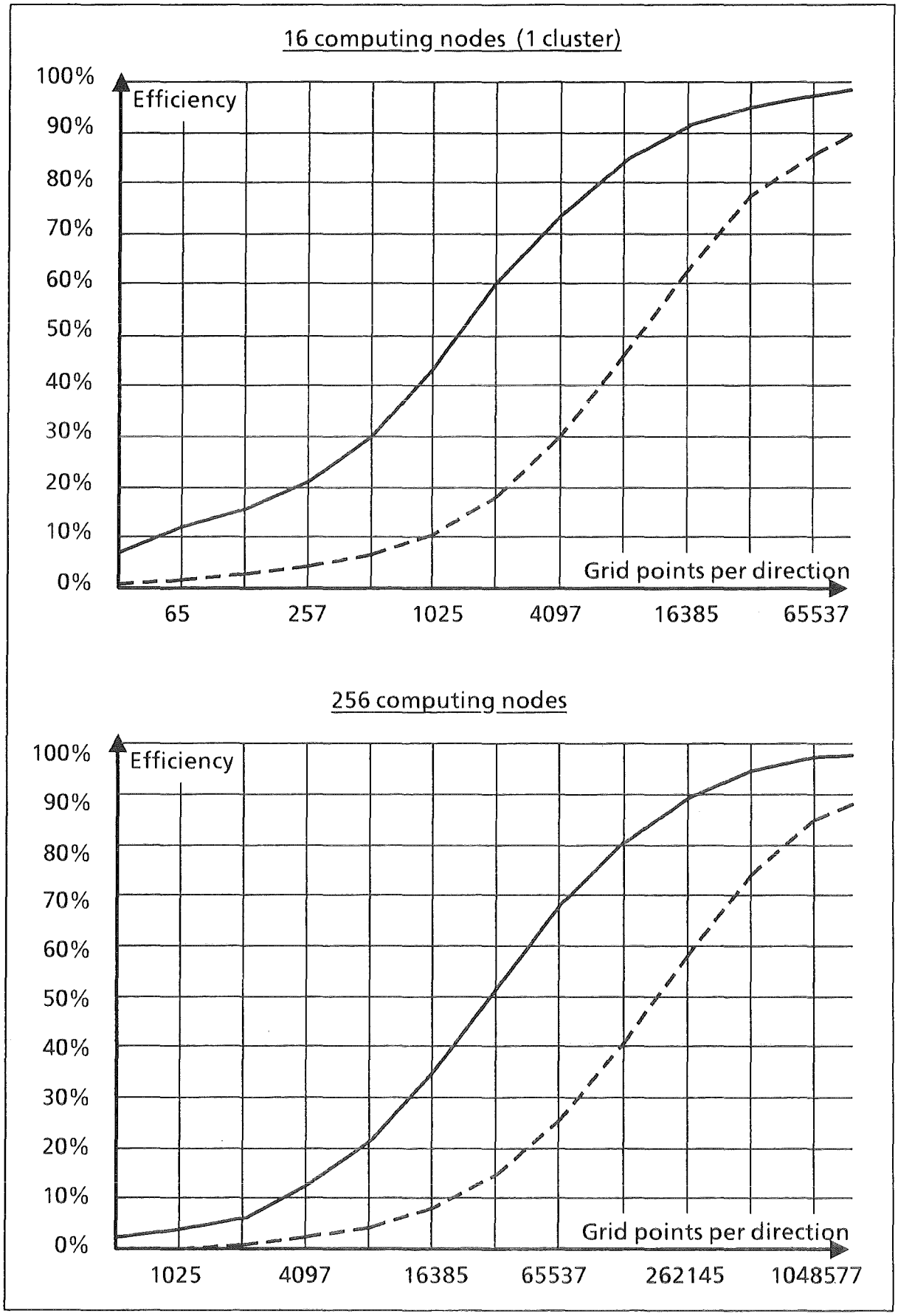

Fig. 4.5.2: Performance increase by optimized parallelization techniques (efficiencies for W-cycles using 16 or 256 computing nodes)

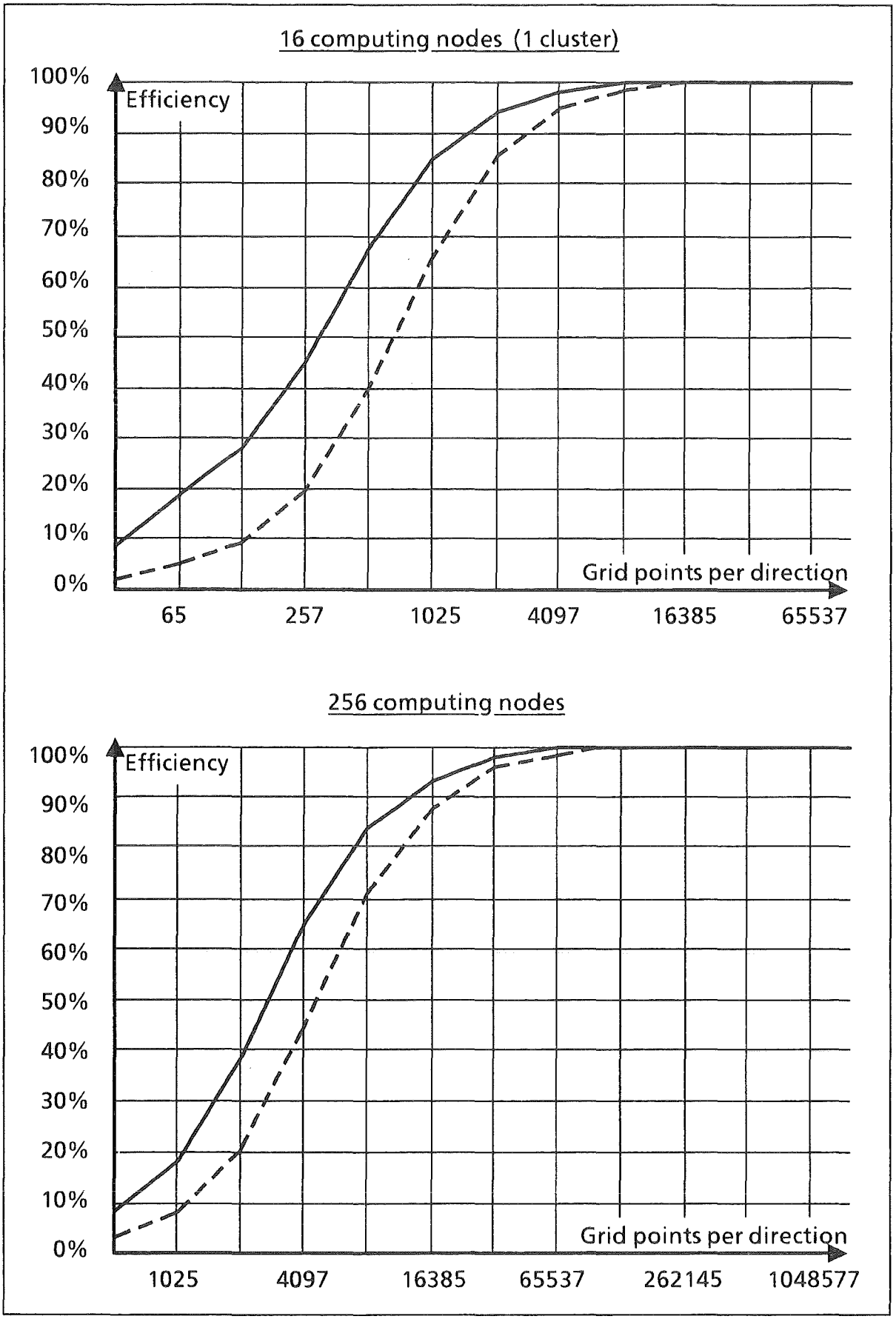

Fig. 4.5.3: Performance increase by optimized parallelization techniques (efficiencies for F-cycles using 16 or 256 computing nodes)

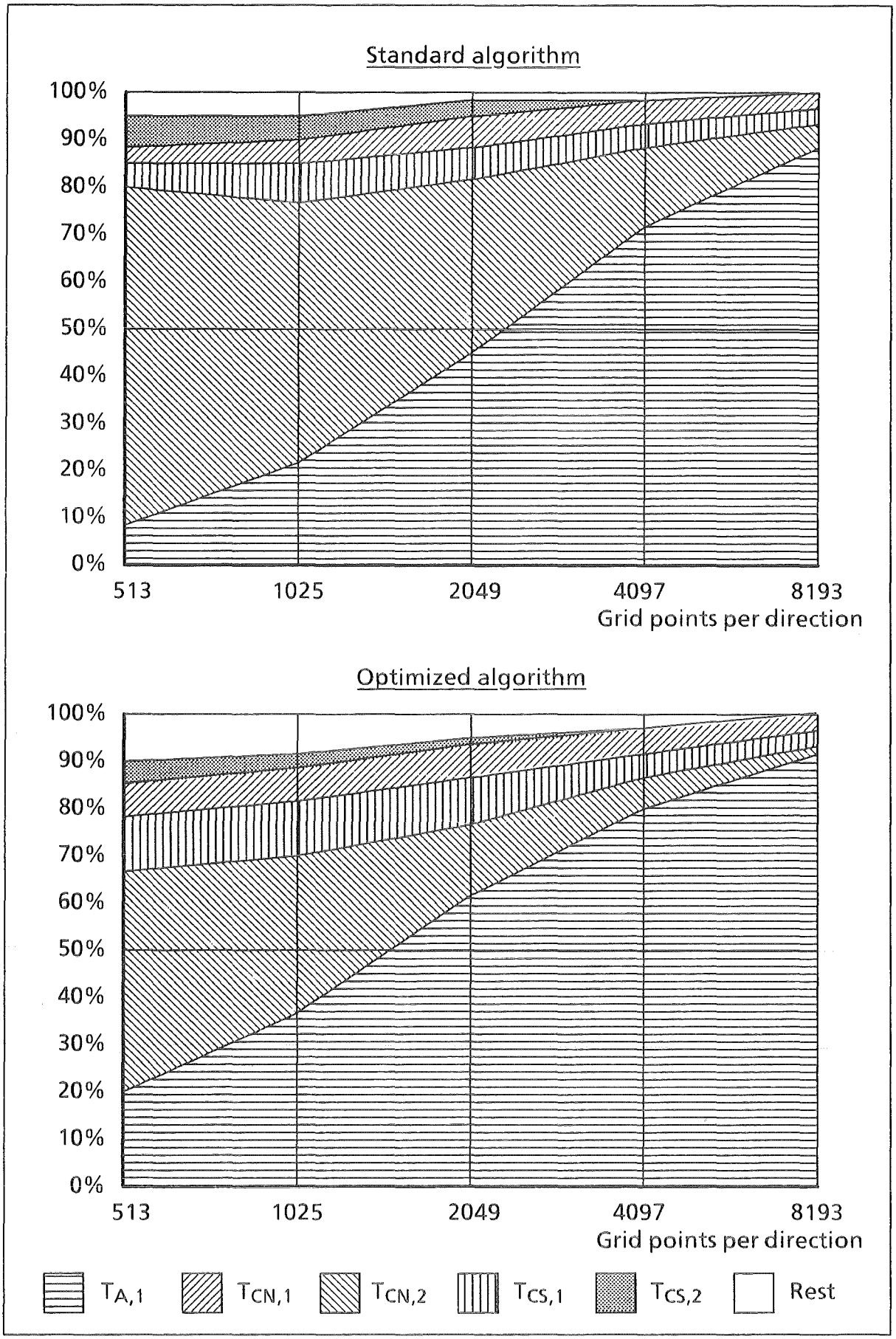

Fig. 4.5.4: Time components estimated for V-cycles (256 nodes)

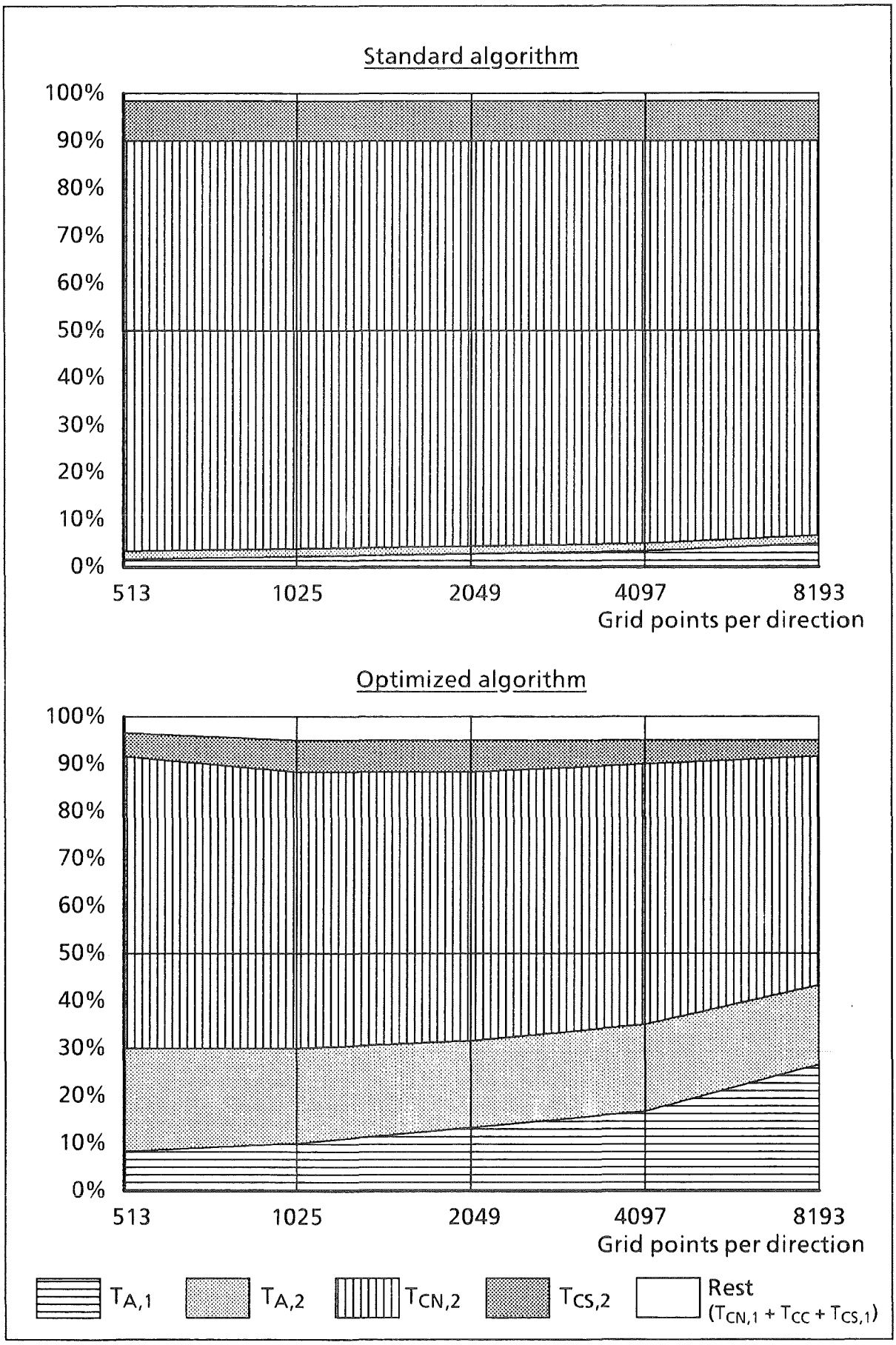

Fig. 4.5.5: Time components estimated for W-cycles (256 nodes)

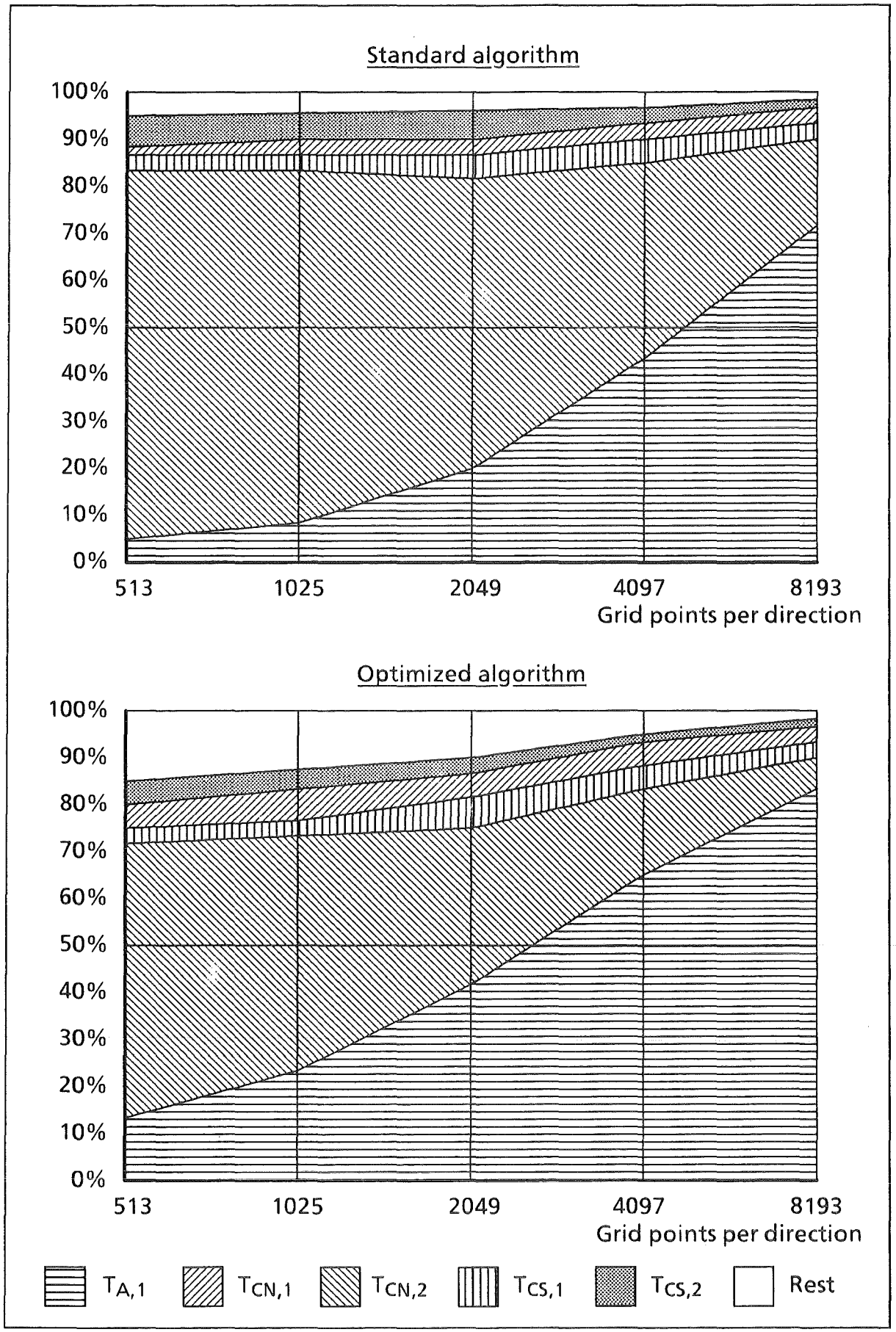

Fig. 4.5.6: Time components estimated for F-cycles (256 nodes)

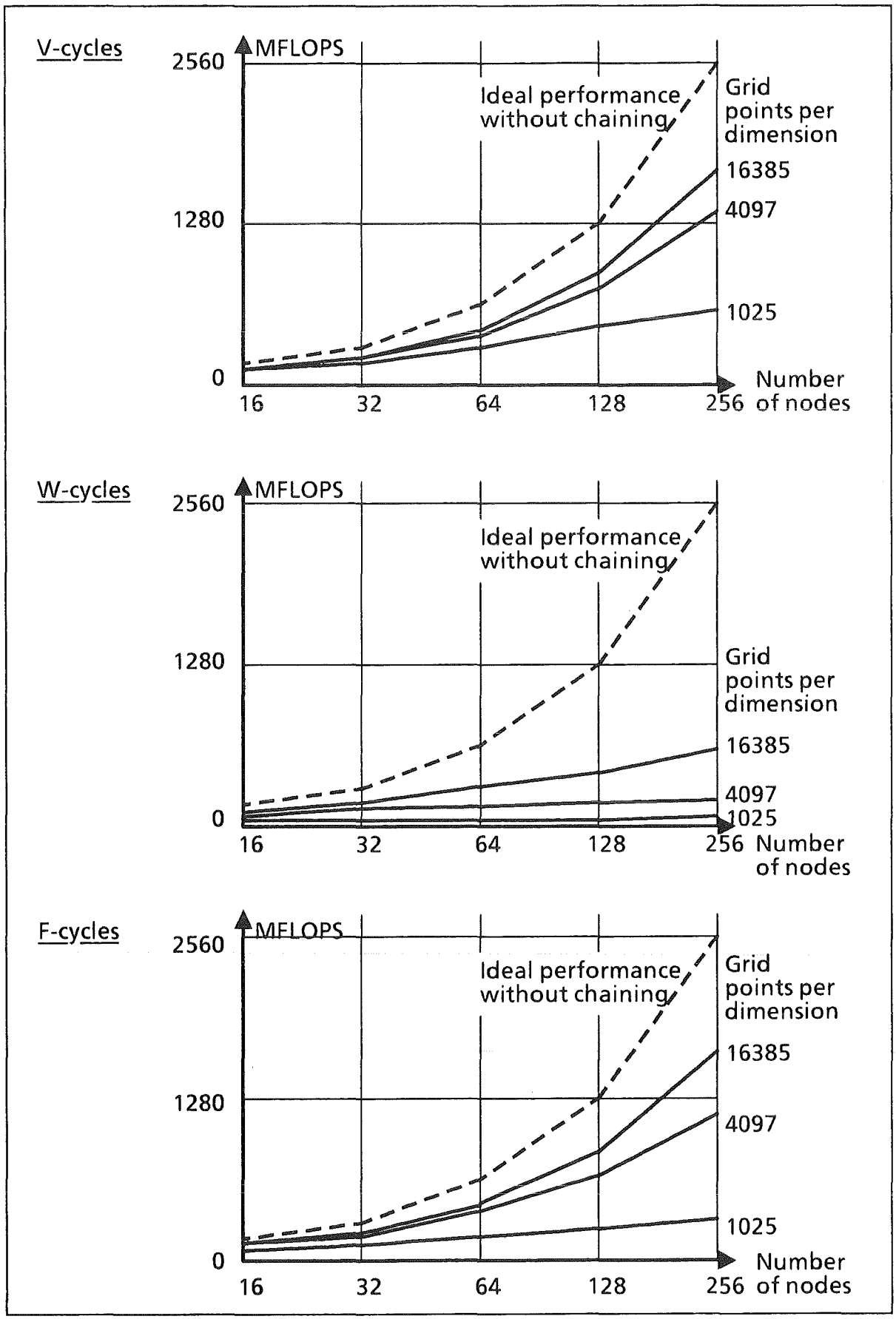

Fig. 4.5.7: Performance (MFLOPS-rates) estimated for V-, W-, and F-cycles of the optimized program, dependent of the number of nodes

5. Application to General Multigrid Methods

In principle, the concepts described above are valid also for other (standard or improved [23, 24]) multigrid implementations on distributed-memory multiprocessors like SUPRENUM. The expected speed-up will be the better the smaller the amount of arithmetic work between two communications is. Especially, the application of these ideas to 2D and 3D multigrid methods for solving the Poisson equation (or similar problems) in standard grids is very promising because there are much less arithmetic operations between two communications than in the problem described in this report.

But on the other hand, there will be additional arithmetic work: In many multigrid algorithms, "red-black" relaxation is used as the smoothing operator, combined with "half weighting" of residuals - which is simplified to "half injection" because of the special relaxation scheme. In combination with the multi-color relaxation scheme, the full "half weighting" operator requiring additional operations and communication must be used. (There is another, numerical, problem: under certain circumstances, the weighting operator has to be slightly modified in order to obtain proper results. This depends upon the sequence in which the four relaxation "colors" are performed. Indeed, best convergence rates can be obtained by "full weighting".)

These additional arithmetic operations increase proportionally, whereas the communication overhead only grows slowly and can be neglected for larger problems.

For this reason, the modified relaxation scheme is even less efficient, if the problem size is so large that the standard algorithm already reaches the peak rate of efficiency.

6. Conclusions

The efficiency of the implementation of multigrid methods on distributed-memory multiprocessors like SUPRENUM is determined by the amount of work necessary for communication activities required by the relaxation sweeps and by the fine-to-coarse transfers of the residuals (restrictions).

This communication overhead obviously can be reduced by a few changes in the parallelized algorithm, namely by simple modifications of the relaxation method and by optimized agglomeration schemes.

Especially if W-cycles are required for numerical reasons, these algorithmical changes result in an improved performance of the parallel computations. But also if V- or F-cycles are sufficient, there is an increase of efficiency.

With small limitations, the ideas described above apply also to general multigrid methods, as well as to other distributed-memory multiprocessors.

7. Literature

[1] T. Westermann:
A Particle-in-Cell Method as a Tool for Diode Simulations.
Nucl. Instr. Meth. A263 (1988), pp. 271-279

[2] T. Westermann:
Numerische Simulationen von technisch relevanten Ionen-Dioden mit der Particle-in-Cell Methode.
Nuclear Research Center Karlsruhe, report no. KfK 4510, January 1989

[3] M. Alef, D. Seldner, T. Westermann:
Numerische Algorithmen für elektrodynamische Modelle und ihre Implementierung auf Supercomputern.
Informatik-Fachberichte 150 (J. Halin, ed.), Springer 1987, pp. 298-305

[4] D. Seldner, M. Alef, T. Westermann, E. Halter:
Parallel Particle Simulation in High Voltage Diodes
(Algorithms and Concepts for Implementation on SUPRENUM).
In [8]

[5] D. Seldner:
Modelle zur Parallelisierung der Teilchenbehandlung in Particle-in-Cell Codes auf MIMD-Rechnern mit lokalem Speicher am Beispiel SUPRENUM.
Nuclear Research Center Karlsruhe, report no. KfK 4495, January 1989

[6] U. Trottenberg:
On the SUPRENUM Conception (version 2).
SUPRENUM report 1, SUPRENUM GmbH, Bonn, Jan. 1987

[7] U. Trottenberg:
SUPRENUM - a MIMD System for Multilevel Scientific Supercomputing.
SUPRENUM report 2, SUPRENUM GmbH, Bonn, Feb. 1987

[8] U. Trottenberg (ed.):
Proceedings of the 2nd International SUPRENUM Colloquium "Supercomputing Based on Parallel Computer Architectures".
Special issue, Parallel Computing 7, 1988

[9] W.K. Giloi:
SUPRENUM: A Trendsetter in Modern Supercomputer Development.
In [8]

[10] K. Solchenbach:
Grid Applications on Distributed Memory Architectures:
Implementation and Evaluation.
In [8]

[11] K. Solchenbach, C.A. Thole, U. Trottenberg:
Parallel Multigrid Methods:
Implementation on SUPRENUM-like Architectures and Applications.
SUPRENUM report 4, SUPRENUM GmbH, Bonn, Aug. 1987

[12] M. Alef:
Effiziente Berechnung elektrostatischer Potentiale mit Mehrgittermethoden in technischen Geometrien.
Nuclear Research Center Karlsruhe, report no. KfK 4613, 1989

[13] J.F. Thompson, Z.U.A. Warsi, C.W. Mastin:
Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review.
J. Comp. Phys. 47, 1982, pp. 1-108

[14] S. Ohring:
Application of the Multigrid Method to Poisson's Equation in Boundary-Fitted Coordinates.
J. Comp. Phys. 50, 1983, pp. 307-315

[15] E. Halter:
Die Berechnung elektrostatischer Felder in Pulsleistungsanlagen.
Nuclear Research Center Karlsruhe, report no. KfK 4072, April 1986

[16] M. Alef, T. Westermann:
Unpublished report, KfK, Juni 1989

[17] A. Brandt:
Multi-Level Adaptive Solutions to Boundary-Value Problems.
Mathematics of Computation Vol. 31 No. 138, 1977, pp. 333-390

[18] W. Hackbusch, U. Trottenberg (Hrsg.):
Multi-Grid Methods.
Lecture Notes in Mathematics 960, Springer-Verlag, 1981

[19] W. Hackbusch:
Multi-Grid Methods and Applications.
Springer Series in Computational Mathematics Vol. 4, Springer-Verlag, 1985

[20] O. Kolp, H. Mierendorff:
Performance Estimations for SUPRENUM Systems.
In [8]

[21] H. Mierendorff, U. Trottenberg:
Performance Estimations for SUPRENUM Systems.
Proc. UNICOM Seminar on Evaluating Supercomputers, 1-3 June 1988, London

[22] R. Hempel, A. Schüller:
Vereinheitlichung und Portabilität paralleler Anwendersoftware durch Verwendung einer Kommunikationsbibliothek.
Arbeitspapiere der GMD 234, St. Augustin, November 1986

[23] P. O. Frederickson, O. A. Mc Bryan:
Parallel Superconvergent Multigrid.
Cornell Theory Center, Ithaca (NY), Technical Report, July 1987

[24] W. Hackbusch:
Robust Multi-Grid Methods.
Institut für Informatik und Praktische Mathematik der Christian-Albrechts-University of Kiel, report no. 8708, July 1987

Acknowledgement

I would like to mention the helpful discussions and support by the members of the SUPRENUM project, especially U. Trottenberg, U. Brass, R. Hempel, H. Mierendorff, C.A. Thole, R. Vogelsang, and K. Witsch, and by my colleagues D. Seldner and T. Westermann.