1986

Domain Oriented Analysis of PDE Splitting Algorithms

Dan C. Marinescu

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
86-627

DOMAIN ORIENTED ANALYSIS OF PDE SPLITTING ALGORITHMS

Dan C. Marinescu
John R. Rice

Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-627
September 1986

DOMAIN ORIENTED ANALYSIS OF PDE SPLITTING ALGORITHMS

Dan C. Marinescu*
John R. Rice**

Computer Sciences
Purdue University
West Lafayette, Indiana 47906

CSD-TR 627
September 22, 1986

Abstract

We consider the performance of Schwarz splitting algorithms for PDE problems on the hypothetical Multi-FLEX machine. The particular Multi-FLEX considered consists of eight clusters of FLEX/32 multiprocessors. We introduce the concept of execution domains to model the three levels of memory on these machines: local, locally shared and global. We apply the Stochastic High Level Petri Nets (SHLPN) method to model the performance of these PDE splitting algorithms on a Multi-FLEX machine. For very large, but realistic, applications we project potential speedup of 100,000 for 2D problems and billions for 3D problems along with processor utilization of over 99 percent. Real computations on real machines can fall far short of this potential and still be very successful.

* Work supported in part by ARO grant DAAL03-86-K-0106
** Work supported in part by AFOSR grant 84-0385

DOMAIN ORIENTED ANALYSIS OF PDE SPLITTING ALGORITHMS

1. OVERVIEW

Domain oriented multiprocessor performance analysis is based upon a simple modeling technique which assumes that during the processing of an application, each processor of a multiprocessor system migrates among a set of *execution domains*. The processing speed in each domain is different and each domain is associated with a different level of contention for common resources. The transition from one domain to another can be subject to constraints determined by the maximum population size of the target domain and by synchronization conditions determined by the application.

The present paper uses the domain oriented performance analysis for partial differential equations (PDE) applications running on a Multi-FLEX system. The Multi-FLEX systems introduced in [1] are shared memory multiprocessor systems in which each processor has access to a hierarchy of memories with different access times. These machines are basically clusters of multiprocessors organized in a particular way. We consider a tree organization here, providing a machine with some of the characteristics of the Cedar machine being built at the University of Illinois. The application is partitioned in such a way that each processor needs to access both the local memory and different levels of shared memory in order to pass results of its own computations and use results produced by other processors.

In the followings, we consider a Multi-FLEX with two levels of shared memory: locally shared and global. The basically shared memory is accessible to all processors of one multiprocessor system, and the global memory is accessible to all processors of all systems. During an execution cycle, the computation on each processor P migrates between three *execution domains*, local (L), locally shared (LS), and global (G), as shown in Figure 1.

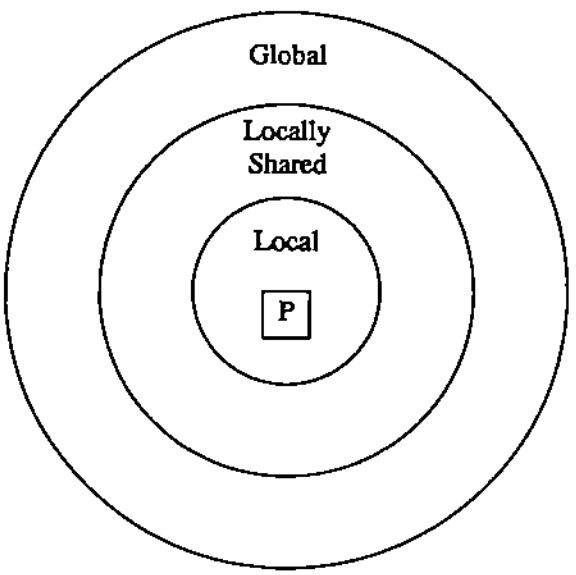

Figure 1. The execution domains as viewed by the processor P.

The purpose of our study is to determine the expected level of performance of the system depending upon:

(a) the hardware characteristics of the system,

(b) the characteristics of the application running on the system.

Our main concern is to determine the effect of contention for shared resources (busses and shared memory) upon the processor utilization. The method used is based upon the use of Stochastic High Level Petri Nets, as presented in [2], [3].

We say that a processor executes in a given domain when it accesses (reads or writes) storage located in that domain. The access time and consequently the execution speed differs in different domains. The access time increases as we move from the center of Figure 1, towards the periphery, and the execution speed decreases accordingly. The "slow-down" factor for the locally shared domain is defined as the ratio between the access time in locally shared memory and the access time in local memory. A similar "slow down" factor is defined for the global domain.

Two models will be considered. In the first model we assume that each processor's behavior is independent of the others, as soon as it has finished execution in a given domain, it migrates to the next domain. It should be pointed out that a low level of coordination exists since only one processor of each multiprocessor system is allowed to execute in the shared memory domain of the particular system at a given time. This condition is related to the assumption that when a processor is granted the common bus, it does not release the bus until it has finished execution in the locally shared memory domain. Moreover only one processor can be in the global domain at any given time.

In the second model, a certain level of synchronization is assumed. More precisely when a processor finishes execution in a certain domain it migrates to the next domain only when all processors in his "group" have completed the execution in the current domain. The "group" is determined by the semantics of computations performed by the application. It consists usually of processors which operates on data located in adjacent regions.

We have been able to carry out all the computations of the first model and to produce values of processor utilization as a function of other model parameters. These values show that the Schwarz splitting approach to PDE computation is very attractive, utilizations of over 99 percent are projected for broad classes of real applications. Further, the Schwarz splitting approach itself can be very powerful so that combining it with the Multi-FLEX machines show potential speed-ups of the order of 100,000 for 2D problems with a 500,000-1 million unknowns and of the order of billions for 3D problems with 10-50 million unknowns. A real machine and computation can fall far short of this potential and still be dramatically successful.

2. THE ARCHITECTURE OF A MULTI-FLEX SYSTEM

The system being analyzed consists of interconnected multi-processor systems as shown in Figure 2. This is one of several multi-FLEX configurations introduced in [1].

Though the multi-processor system can be fully connected we assume that we have only a star configuration in which the seven systems $MP_1 + MP_7$ are all connected to the global system MP_0. The configuration of each of the systems MP_1 to MP_7 is identical and it is presented in Figure 3.

Let us denote by $P_i \cdot MP_j$, $i \in [1,9]$, $j \in [1,7]$ any processor i of the j-th system which performs a user task. LM_i is the local memory of processor P_i, and it is accessible through a local bus. Each processor P_i of system MP_j can access its locally shared memory through a common bus.

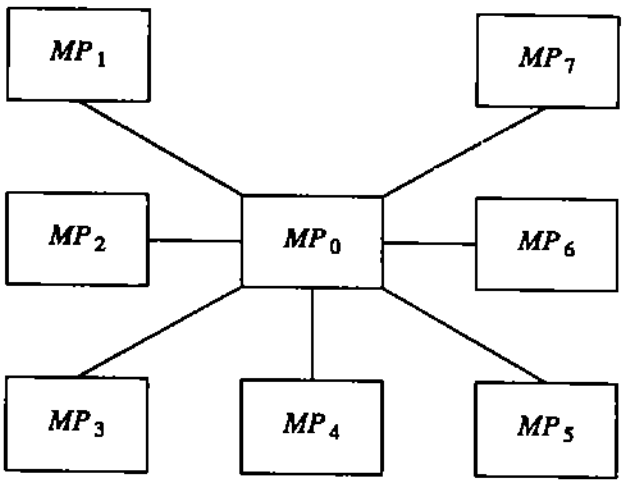

Figure 2. The Configuration of of Multi-FLEX System.

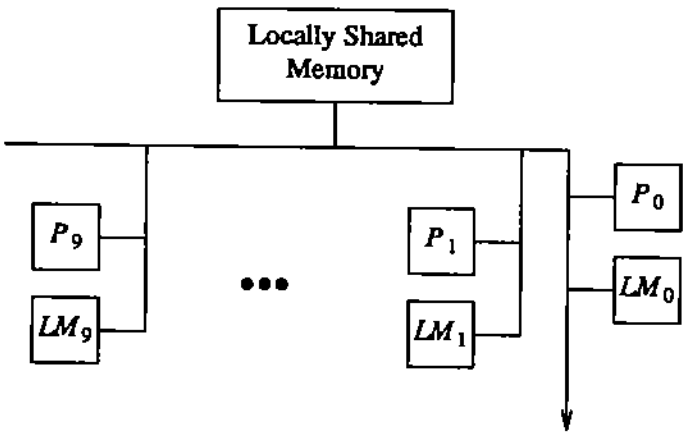

Figure 3. The configuration of MP_j, $j \in [1,7]$.

We assume that processor P_0 of each MP_j system, $j \in [1,7]$ as well as all P_i of MP_0 perform control functions and their behavior does not affect the overall load placed upon the shared resources (shared memory and common busses).

The communication speed, in particular the access time for the locally shared and global memory, has a significant influence upon the performance of the system. Since a Multi-FLEX is at this time a paper machine no actual measurements exists to determine the slow-down factor for the global memory. Even the establishment of a slow-down factor for the locally shared memory raises several problems. First of all the access times are different for different data types (integer, floating point, double precision floating point). Moreover there are different types of processors available, with different access times. For these reasons we have performed our modeling for a range of values for the slow-down factors rather than values measured on our system. The slow-down factor for locally shared domain is assumed to be in the range 1.4 to 4 while the slow-down factor for the global domain is in the 5 to 40 range. For the FLEX/32 we have measured the slow-down factor for the locally shared domain and found it to be about 1.5.

3. THE CHARACTERIZATION OF THE APPLICATION

Each processor running a user process manipulates data (reads and writes) located in three distinct domains:

- local memory
- locally shared memory
- global memory

The size of each domain is defined as the number of references executed by a processor in that domain. Clearly, the domain size depends upon the application. We consider as the class of applications the solution of partial differential equations (PDEs) using the Schwarz splitting method. We outline the structure of this method in a way intended to hide the technical details. See [4], [5], [6] and [7] for detailed and technically precise descriptions. This method is quite general, we first consider the case here of 2D problems.

A large domain is subdivided into *k_pieces*, pieces which overlap to cover the domain. Each piece is further subdivided into *k_regions*, regions R_{ij} which overlap to cover the piece and all its boundaries. Figure 4 shows one piece with *k_regions* = 9. All regions except 5 extend outside the piece unless they are on the boundary of the entire problem domain.

We now describe the key idea of Schwarz splitting in terms of processing region 5 seen in Figure 4. First, we guess at values of the solution on the boundary ∂R_{ij} of region 5 (this is the heavy black square). Second, we use these boundary values to solve the PDE on region 5. This procedure is carried out simultaneously on the 8 neighboring regions. We obtain new boundary values for region 5 by taking the solutions from neighboring regions evaluated along the boundary of region 5. Where two or more regions overlap a part of the boundary, then a particular averaging technique is used. This process is iterated and it can be shown to converge quite fast for large classes of model problems.

We now organize the method for the Multi-FLEX shown in Figure 2. We take *k_pieces* = 7 and the computational structure is:

```
ITERATE until Converged
    DO PARALLEL (i = 1 to 7)
        DO PARALLEL (j = 1 to k_region)
            Solve the PDE on region R_ij
        END PARALLEL
        DO PARALLEL (j = 1 to k_region)
            Obtain new boundary values for R_ij from values on
            ∂R_ij in neighboring regions of piece j.
        END PARALLEL
    END PARALLEL
    DO PARALLEL (i = 1 to 7)
        DO PARALLEL (j = 1 to k_regions)
            If R_ij overlaps the boundary of the i-th piece
            then obtain new boundary values on that part of
             ∂R_ij in neighboring regions of neighboring pieces
        END PARALLEL
    END PARALLEL
END ITERATION
```

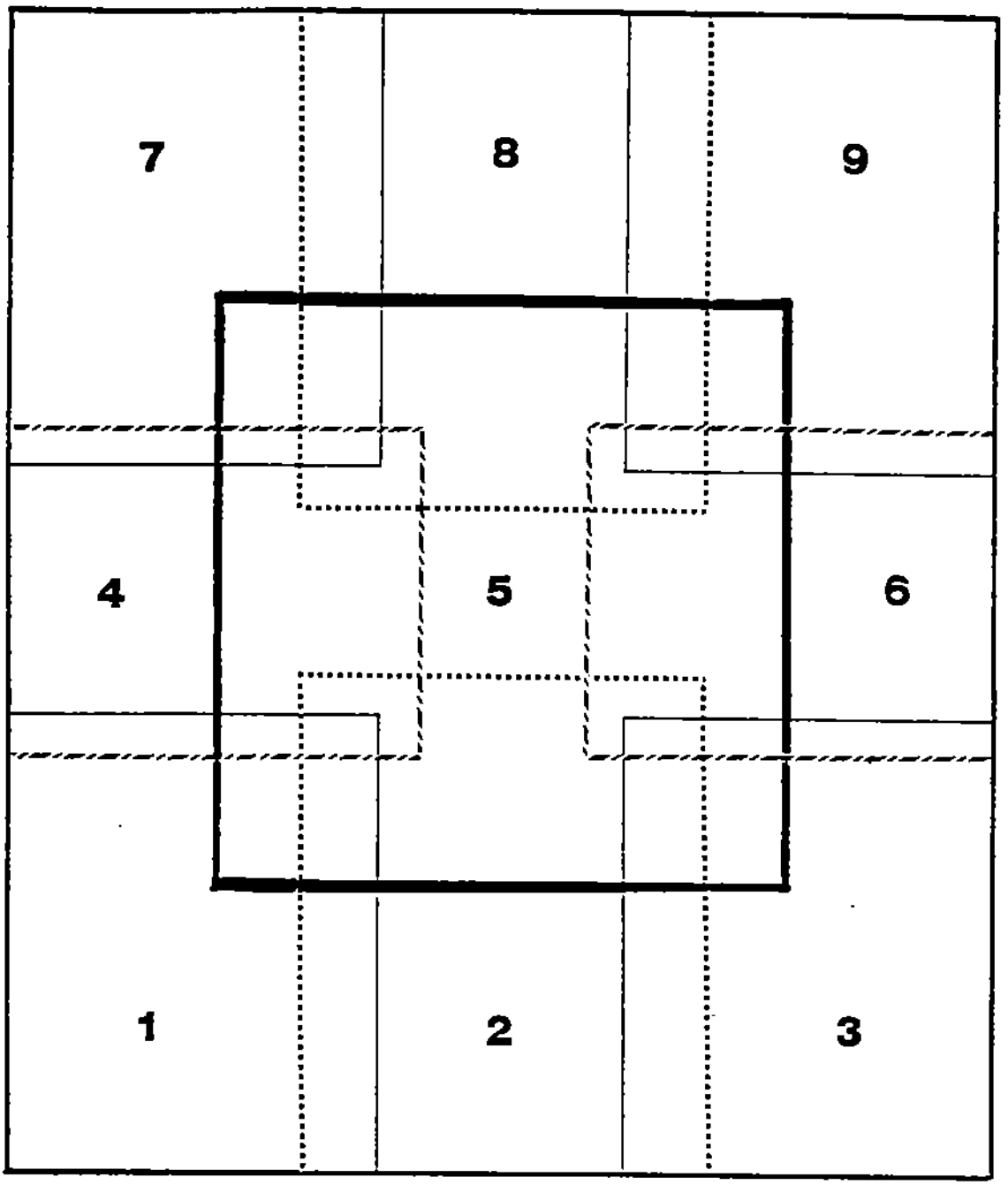

Figure 4. A piece showing nine overlapping regions.

A numerical method further subdivides each region R_{ij} into N^2 elements or mesh points (assuming the regions are square). The work to solve the PDE depends on the method used and is modeled by $C(N^2)^k$. Realistic values of k are from 1 (for FFT methods on "easy" problems) to 2 (for Gauss elimination type methods). The best and worst cases for Schwarz splitting are $k = 1$ and 2, respectively and we study these two cases which are called *pessimistic* and *optimistic* later. There are numerical methods that correspond to intermediate values of k. We take C = 1 since this just reflects the machines speed (we are not trying here to compare different numerical methods). We consider values of N from 10 to 100 which are typical for common problems.

The amount of information exchanged between iterations is clearly proportional to the perimeters of the regions (on the local piece level) or to the perimeters of the pieces (on the higher, inter-piece level). Different numerical methods and types of boundary values require different amounts of information to be exchanged for one piece, however a model of the form $4 \times N \times K$ is reasonable where K is a small integer, we consider the range 2 to 5 for K.

The problem data is organized so that locally shared memory contains all the boundary information for the regions in one piece and the global memory contains all the boundary information for regions which overlap piece boundaries. Since we have 7 pieces, one can reasonably estimate that the average amount of information in the global memory is about $N \times K$ for each piece. This is quite accurate if the pieces are long and thin, otherwise we have a uniform model of a non-uniform situation. Thus the size of the local domain is defined to be equal to $(N^2)^k$, the size of the locally shared domain is $4 \times N \times K$ and the size of the global domain is $N \times K$. Since the communication cost is higher as we migrate from the local to locally shared domain and to the global one, we expect that the overall system performance, in particular the processor utilization, increases when N increases and decreases when K is increased.

As similar analysis can be applied to the 3D case and Table 1 presents a summary of domain sizes for the two and three dimentional cases. The values of k for the optimistic and pessimistic cases in 3D are 7/3 and 1, respectively. Two approximations for the domain sizes are considered for each case, an optimistic and a pessimistic one. The term optimistic is used when the life time in the local domains is larger. This corresponds to a larger computation but one where this architecture is more suitable, hence the use of optimistic.

		Domain Size		
		Local	Locally Shared	Global
2D	Optimistic	N^4	4 N K	N K
	Pessimistic	N^2	4 N K	N K
3D	Optimistic	N^7	$6N^2K$	$2N^2K$
	Pessimistic	N^3	$6N^2K$	$2N^2K$

Table 1. Domain size functions of N and K.

In order to model the system we define the average "life.time" in a given domain as the product between the slow-down factor of the domain and the size of the domain. In this way the average life time provides a characterization of both the system and the application. Table 2 shows a summary of domain sizes, slow-down factors and average life-times for a two dimensional (2D), pessimistic case.

Domain	Domain Size	Slow-Down Factor	Average Life Time
Local	N^2	1	N^2
Locally Shared	$4NK$	μ_{ls}	$(4NK)\mu_{ls}$
Global	NK	μ_g	$(NK)\mu_g$

Table 2. Domain size - life time relationship for the pessimistic 2D case.

4. MODELING ASSUMPTIONS

We now explicitly state the assuming mode in our model of these computations on the multi-FLEX.

(A0) A processor which executes only in its local domain achieves full utilization.

(A1) We model the "computational behavior" of the system and consider that all the processors in the system have been loaded with the appropriate code prior to the beginning of our experiment. In an actual system we recognize that such a computational period is preceded by a loading phase in which, depending upon the application and the scheduling policy used, different user processes are assigned to the available processors and loaded for execution. Measurements performed recently indicate that for our FLEX system, in the concurrent C environment, the time between the execution of a "cobegin" statement and the actual activation of the code in a processor is of the order of one second. Nevertheless we can use our approximations to characterize the overall system behavior when the execution time of the code loaded in all processors is much larger than the time to actually load the code.

(A2) All processors exhibit identical behavior, they execute identical process code loaded into their local memory.

(A3) Each processor performs according to the following pattern:

(a) It executes in the *local domain*. The average lifetime in the local domain is denoted by τ_l and it is determined by the size of the domain σ_l (the number of references) and the slow-down factor denoted by μ_l. In case of local domain we have $\mu_l = 1$. The domain size, σ_l, is determined by the application, through two parameters N and K as shown in Table 1. We have

$$\tau_l = \sigma_l \cdot \mu_l \tag{1}$$

with

$$\mu_l = 1 \tag{2}$$

The transition rate from the local domain is defined as:

$$\lambda_l = \frac{1}{\tau_l} \tag{3}$$

(b) After completing execution in the local domain each processor moves to the locally shared domain. The average duration of this transition process is denoted by $\tau_{l,ls}$ and it is assumed to be very short as compared to τ_l. The corresponding rate is $\lambda_{l,ls} = \frac{1}{\tau_{l,ls}}$.

(c) The average life-time in the locally shared domain is:

$$\tau_{ls} = \sigma_{ls} * \mu_{ls} \tag{4}$$

The transition rate from the locally shared domain is defined as:

$$\lambda_{ls} = \frac{1}{\tau_{ls}} \tag{5}$$

(d) After completing execution in the locally shared domain, each processor moves into the global domain. The transition period from the locally shared to the global domain takes $\tau_{ls,g}$ units of time. We have assumed this transition period to be considerably shorter than the life-time in the locally shared domain. The corresponding rate is $\lambda_{ls,g} = \frac{1}{\tau_{ls,g}}$.

(e) The average lifetime in the global domain is

$$\tau_g = \sigma_g * \mu_g \tag{6}$$

To complete the cycle, the processor moves back to the local domain. The values of σ_l, σ_{ls} and σ_g are presented in Table 1.

(A4) To execute in the locally shared domain, a processor acquires the common bus and it holds the bus until it completes execution in the locally shared domain. Similarly, to execute in the global domain, a processor acquires the common bus and the global bus and releases them only after finishing execution in the global domain.

5. THE MODEL WITHOUT GROUP SYNCHRONIZATION

The model without group synchronization is presented in Figure 5. As mentioned earlier we use Stochastic High Level Petri Nets (SHLPN) for the modeling of the Multi-FLEX system. In the model without group synchronization it is assumed that only one processor $P_i \cdot MP_j$ $i \in [1,9]$ can be active in the locally shared domain for each $j \in [1,7]$ and only one processor is active in the global domain at a time. Clearly this conservative assumption makes our results lower bounds for the actual system performance.

The places and the transitions present in our SHLPN model as well as their significance is presented briefly in the followings:

P_1 is the "local domain" place. When this place holds tokens, the corresponding processors are active in the local memory.

P_2 is the "queuing for the locally shared domain" place. When this place holds tokens, the corresponding processors are queued for the locally shared domain.

P_3 is the "locally shared domain" place. When this place holds tokens, the corresponding processors are active in the locally shared domain.

P_4 is the "common bus" place. When this place holds a token, a common bus is free.

P_5 is the "queuing for the global domain" place. When this place holds tokens, the corresponding processors are queued for the global domain.

P_6 is the "global memory" place. When this place holds a token, the corresponding processor is active in the global domain.

P_7 is the "global bus" place. When this place holds a token, the global bus is free.

T_1 is the "end of local domain" transition. When it fires, a processor ends its activity in its local domain. The transition rate is $\lambda_1 = \lambda_l$.

T_2 is the "moving to the locally shared domain" transition. This transition is enabled when the common bus is free and there is at least one processor in the queue for the locally shared domain. The transition rate is $\lambda_2 = \lambda_{l,ls}$.

T_3 is the "end of locally shared domain" transition. When this transition fires, the processor ends its execution in the locally shared domain. The transition rate is $\lambda_3 = \lambda_{ls}$.

T_4 is the "moving to the global domain" transition. The transition is enabled when the common bus and the global bus are free and there is at least one processor in the global queue. The transition rate is $\lambda_4 = \lambda_{ls,g}$.

T_5 is the "end of global domain" transition. When it fires, a processor ends its activity in the global domain. The transition rate is: $\lambda_5 = 7\ \lambda_g$.

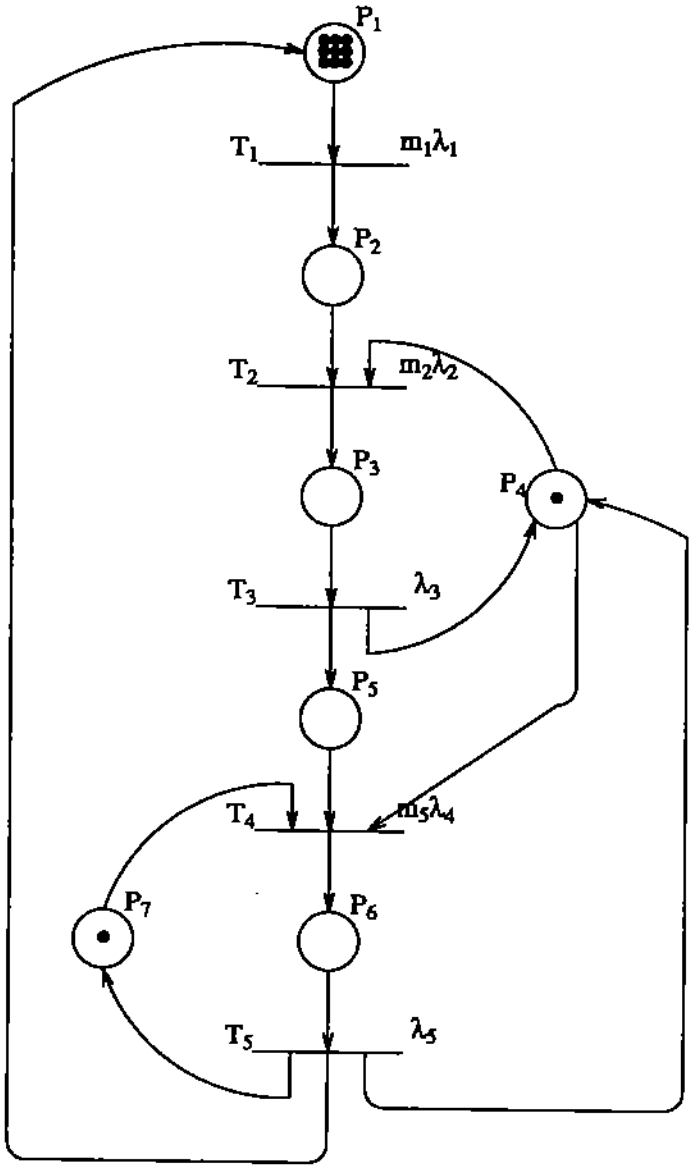

Figure 5. The SHLPN model of a Multi-FLEX system without group synchronization.

The state space is moderate in size, there are 145 states. To simplify the model we have used the following observation: to model the contention for the global bus it is sufficient to consider that the total transition rate of transition T_4 of a Multi-FLEX system consisting of n identical systems is n times the transition rate of a Multi-FLEX system with a single system attached to it. Without this observation the state space would have been prohibitively large (probably one order of magnitude larger). Additionally, we have gained the flexibility of modeling Multi-FLEX systems with a variable number of systems using the same conceptual model and the same system of equations.

The overall processor utilization for difference values of μ_l, and μ_g are presented in Figures 6.1 to 6.8. A summary of the results is presented in Table 3. Q_l, and Q_g are the average queue length for the locally shared and the global domain respectively and η_p is the processor utilization in percent.

	μ_l,	μ_g	N	K	Q_l,	Q_g	η_p (percent)
2-D	1.5	15.0	10	2	0.2083	1.3386	82.81
			100	2	0.0009	0.0	99.99
optimistic			10	5	0.8204	1.5926	73.19
			100	5	0.0009	0.0	99.99
2-D			10	2	0.9179	6.8440	13.77
			100	2	2.2081	3.4054	37.62
pessimistic			0	5	0.6930	7.2118	12.17
			100	5	1.6580	5.3882	21.70
2-D	4.0	40.0	10	2	0.8928	1.5697	72.63
			100	2	0.0009	0.0	99.99
optimistic			10	5	2.1141	2.3660	50.22
			100	5	0.0009	0.0002	99.99
2-D			10	2	0.6823	7.2284	12.10
			100	2	1.6120	5.4941	21.40
pessimistic			10	5	0.5786	7.3857	11.50
			100	5	1.0762	6.5667	15.07
3-D	1.5	15.0	10	3	0.0009	0.0029	99.95
			100	3	0.0008	0.0	99.99
optimistic			10	7	0.0014	0.0154	99.81
			100	7	0.0008	0.0	99.99
3-D			10	3	0.6759	7.2386	12.05
			100	3	1.5533	5.5912	20.61
pessimistic			10	7	0.5811	7.3822	11.51
			100	7	1.0788	6.5549	15.18
3-D	4.0	40.0	10	3	0.0016	0.0201	99.75
			100	3	0.0008	0.0	99.99
optimistic			10	7	0.0009	0.001	99.98
			100	7	0.008	0.0	99.99
3-D			10	3	0.5714	7.3965	11.46
			100	3	1.0227	6.6568	14.67
pessimistic			10	7	0.5306	7.4557	11.26
			100	7	0.7645	7.0982	12.63

Table 3. Processor utilization and average queue lengths Q_l, and Q_g to enter the locally shared/global domains as function of hardware and application parameters.

Figures 6.1-6.4 show utilization for the optimistic and pessimistic 2D cases for two pair of slowdown factors: (1.5 locally shared, 15 global) and (4 locally shared, 40 global). We expect the (1.5, 15) case to be the more realistic, but the (4, 40) pair shows the effect of slower communication. It is clear from these figures that the utilization is very good for the optimistic case even

for small problems ($N \leq 25$) while it is poor for the pessimistic case even for $N = 100$. Recall that this Multi-FLEX has $7 \times 9 = 63$ processors so that $N = 25$ and 100 correspond to 2D problems with 40,000 and 630,000 elements, respectively. It is not surprising that the Schwarz splitting approach to parallelization of an FFT method is not very competitive for an 800×800 problem. FFT and similar fast methods are not applicable to most PDE problems and it is here that the power of Schwarz splitting and parallelism comes into play.

If we assume a 20 percent overlap in the splitting regions, then we can compute the operation counts for ordinary sequential Gauss elimination to be $(.8N \times 63)^4 = 6{,}400{,}000N^4$ and Schwarz splitting to be $63pN^4$ where p is the number of iteration. Thus, 100,000 iterations would be the break even point whereas we hope for 25 to 50 iterations in practice. We hope for a speedup of perhaps 2000 in this particular case just due to using Schwarz splitting. Then there is another speedup of 63 from using parallelisms for a total potential speedup of about 100,000 once N becomes of moderate size. While a real machine and computation might not reach this potential, it can fall far short and still be dramatically successful.

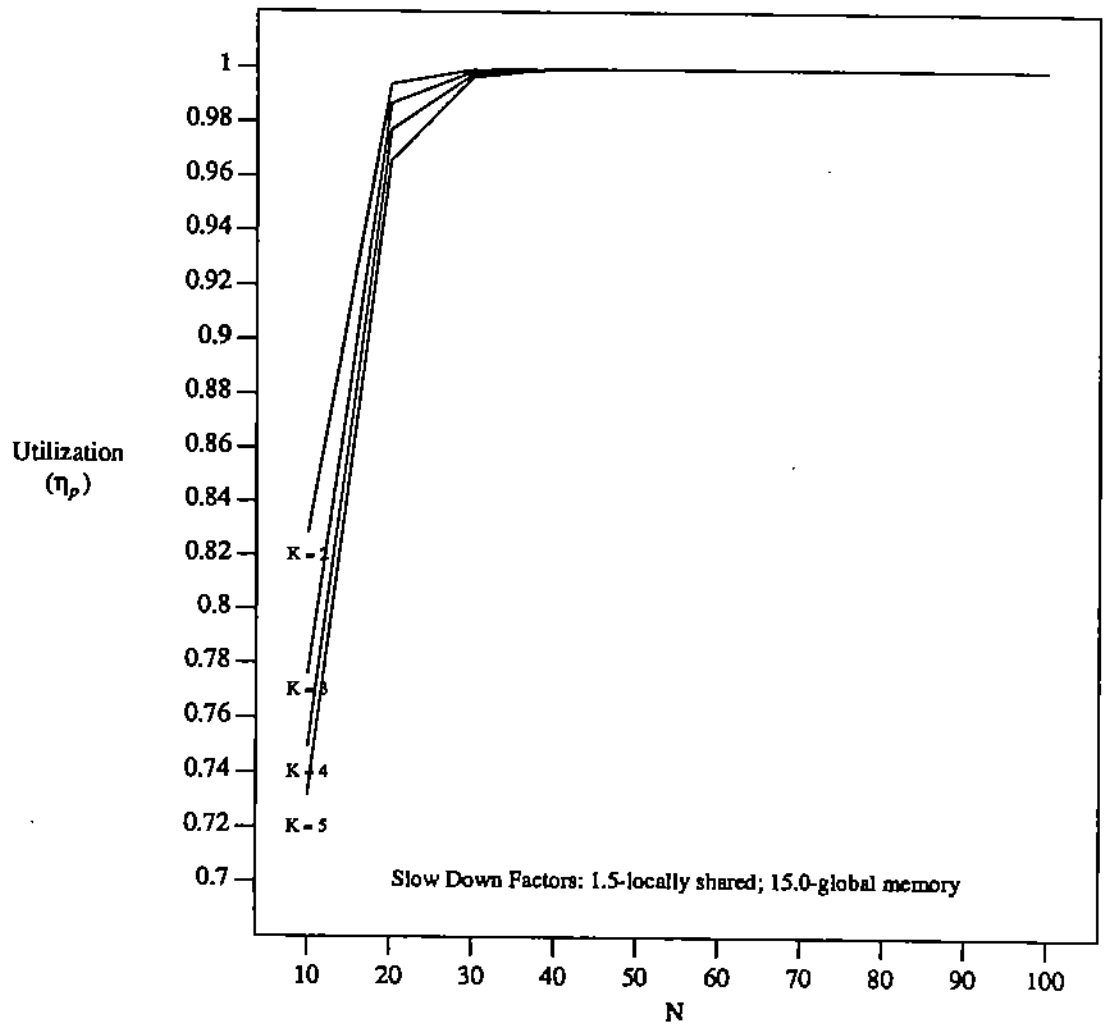

Figure 6.1. The utilization of a processor versus N and K for the optimistic 2D case of slowdown factors (1.5 locally shared, 15.0 global memory).

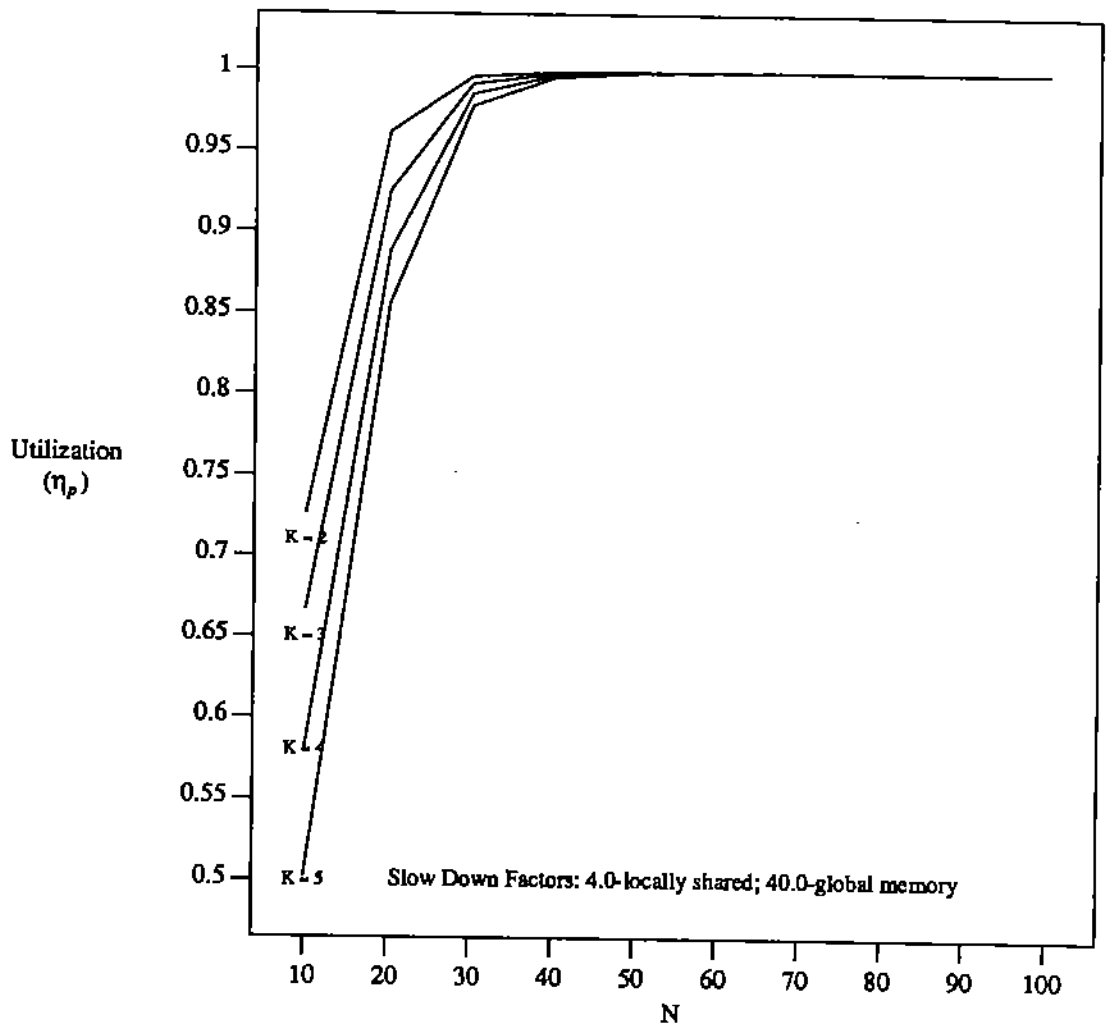

Figure 6.2. The utilization of a processor versus N and K for the optimistic 2D case of slowdown factors (4.0 locally shared, 40.0 global memory).

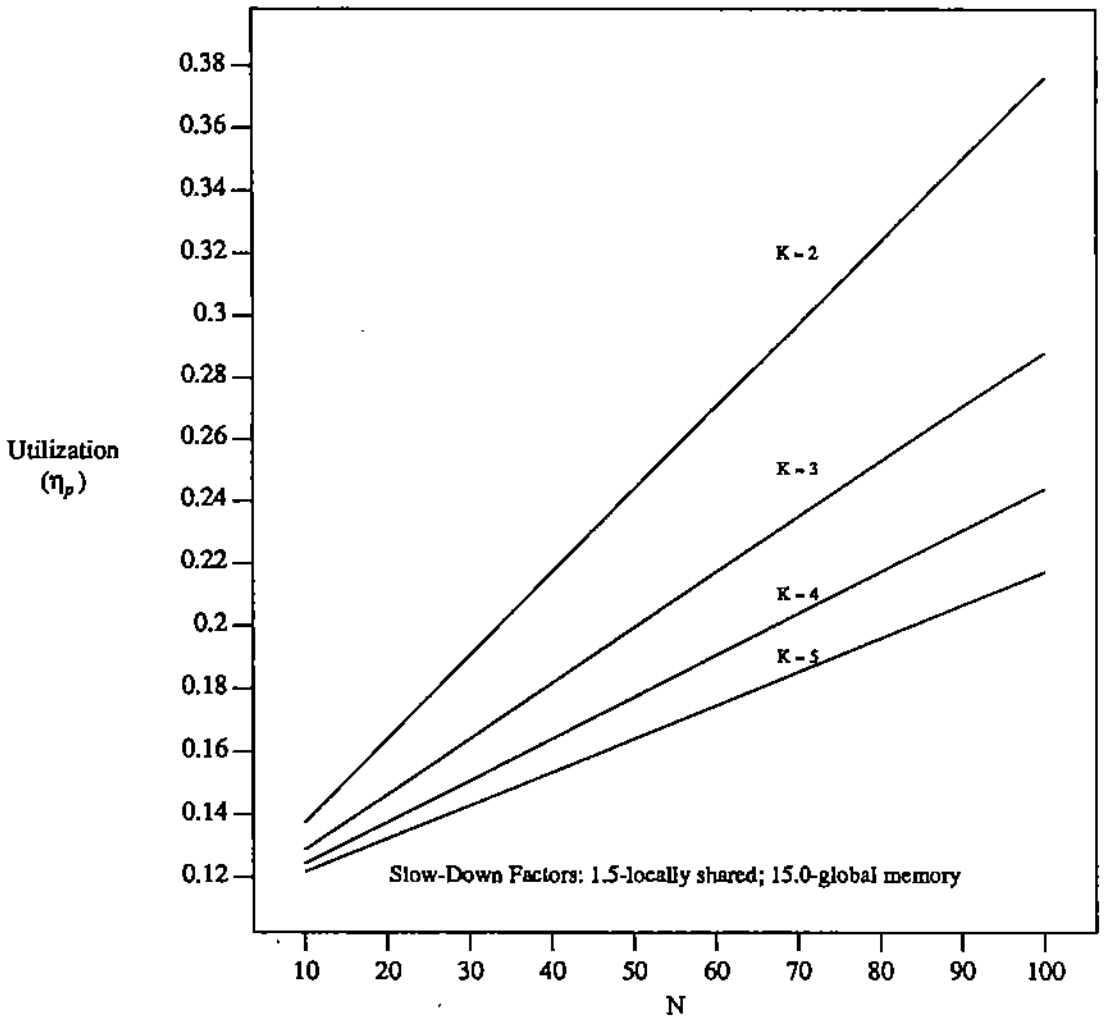

Figure 6.3. The utilization of a processor versus N and K for the pessimistic 2D case of slowdown factors (1.5 locally shared, 15.0 global memory).

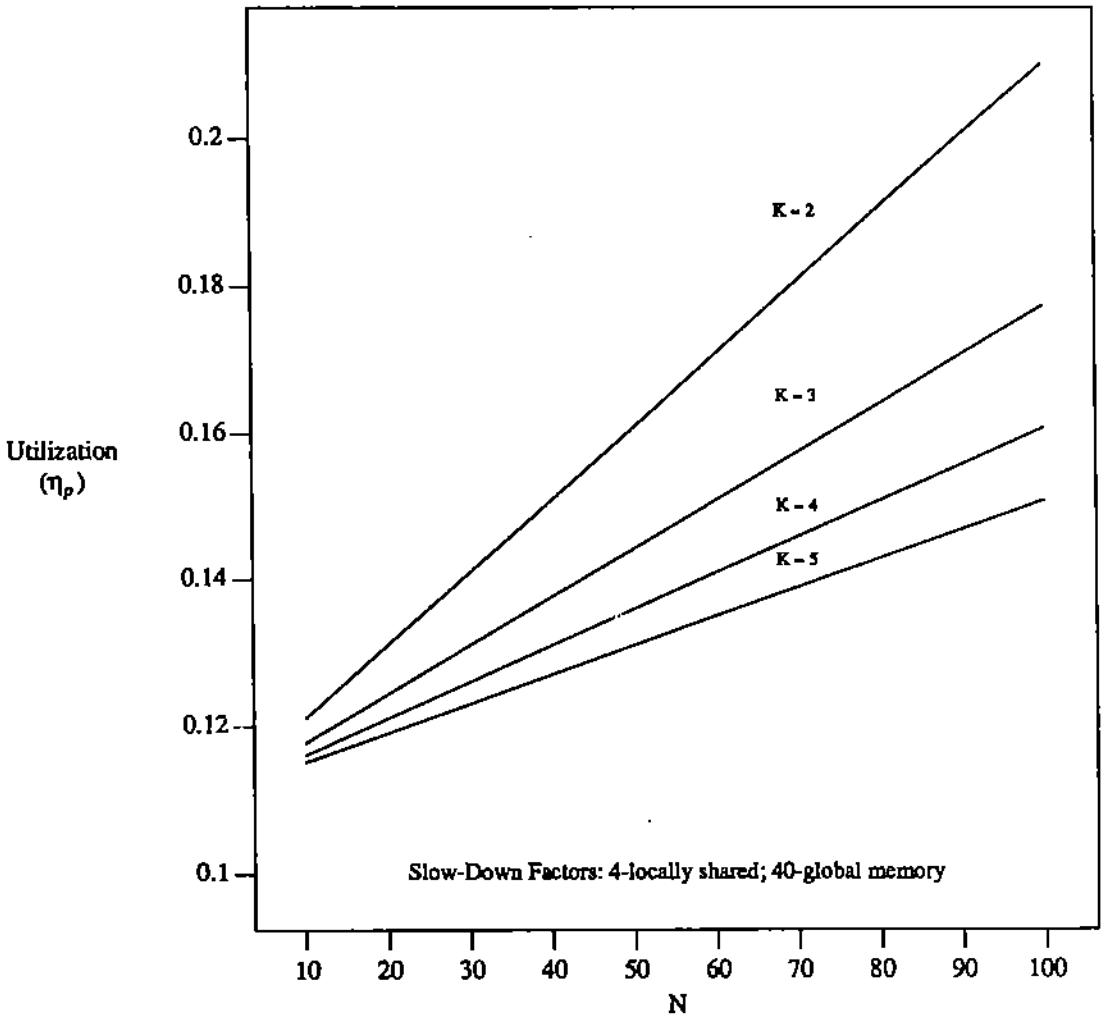

Figure 6.4. The utilization of a processor versus N and K for the pessimistic 3D case of slowdown factors (4 locally shared, 40 global memory).

Figures 6.5 to 6.8 show the optimistic and pessimistic 3D cases for the same two pairs of slowdown factors. We see extremely high levels of utilization (100 percent) in the optimistic case even for small values of $N (N \geq 10)$. The pessimistic case still shows poor utilization. However, the balance is strongly shifted toward high utilization levels for the intermediate values of k not shown here. These correspond to various iterative methods (SOR, SSOR or ADI) or sparse matrix techniques being used on the individual regions.

We repeat the operations count analysis in the 3D case and find the ordinary Gauss elimination count to be about 800 billion N^7 while the Schwarz splitting count is $63pN^7$. It is obvious that Schwarz splitting is even more advantageous, especially in view of the fact that the number p of iteration tends to be slightly less in 3D than in 2D. The sizes of the problems involved here are huge, for $N = 25$ and 100 we have, respectively, 1 million and 63 million unknowns.

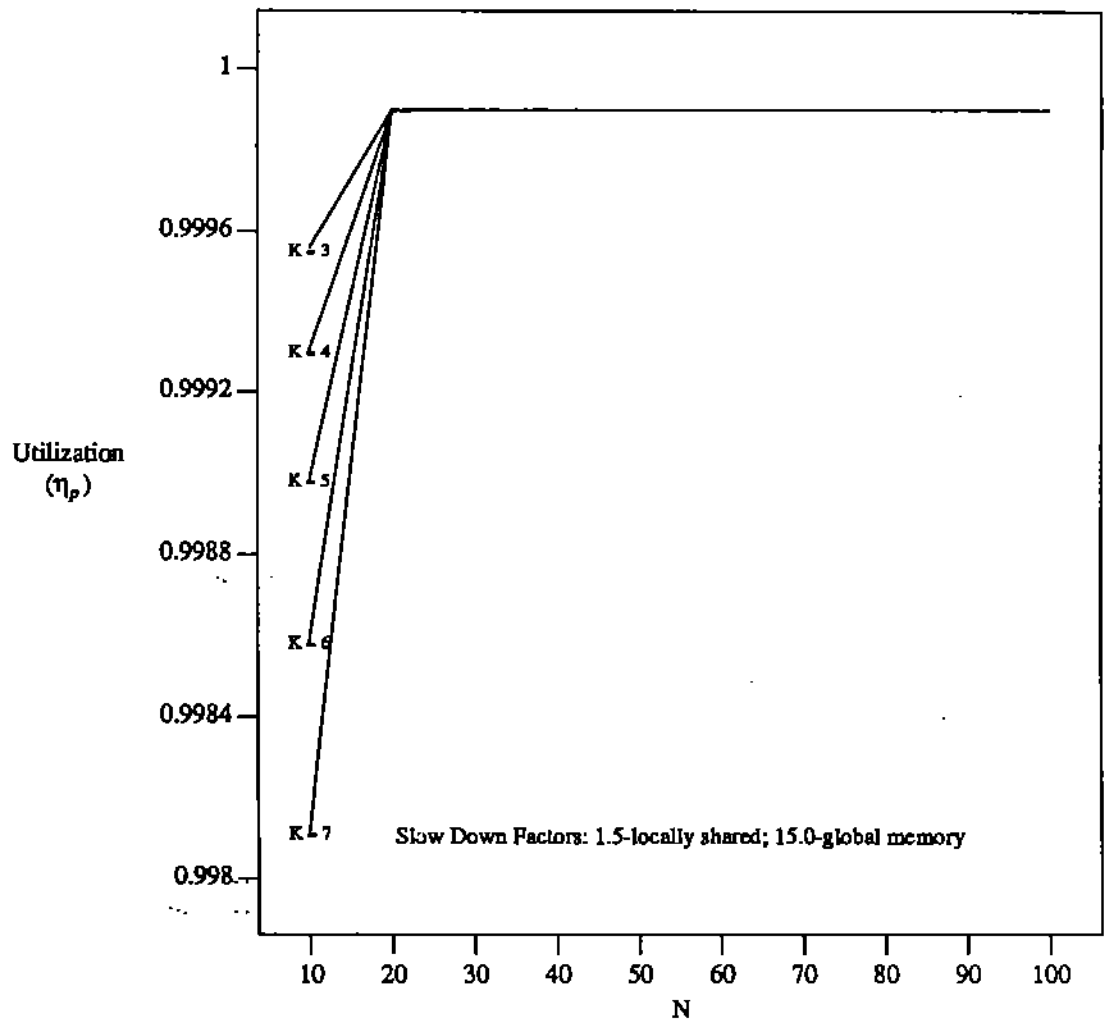

Figure 6.5. The utilization of a processor versus N and K for the optimistic 3D case of slowdown factors (1.5 locally shared, 15.0 global memory).

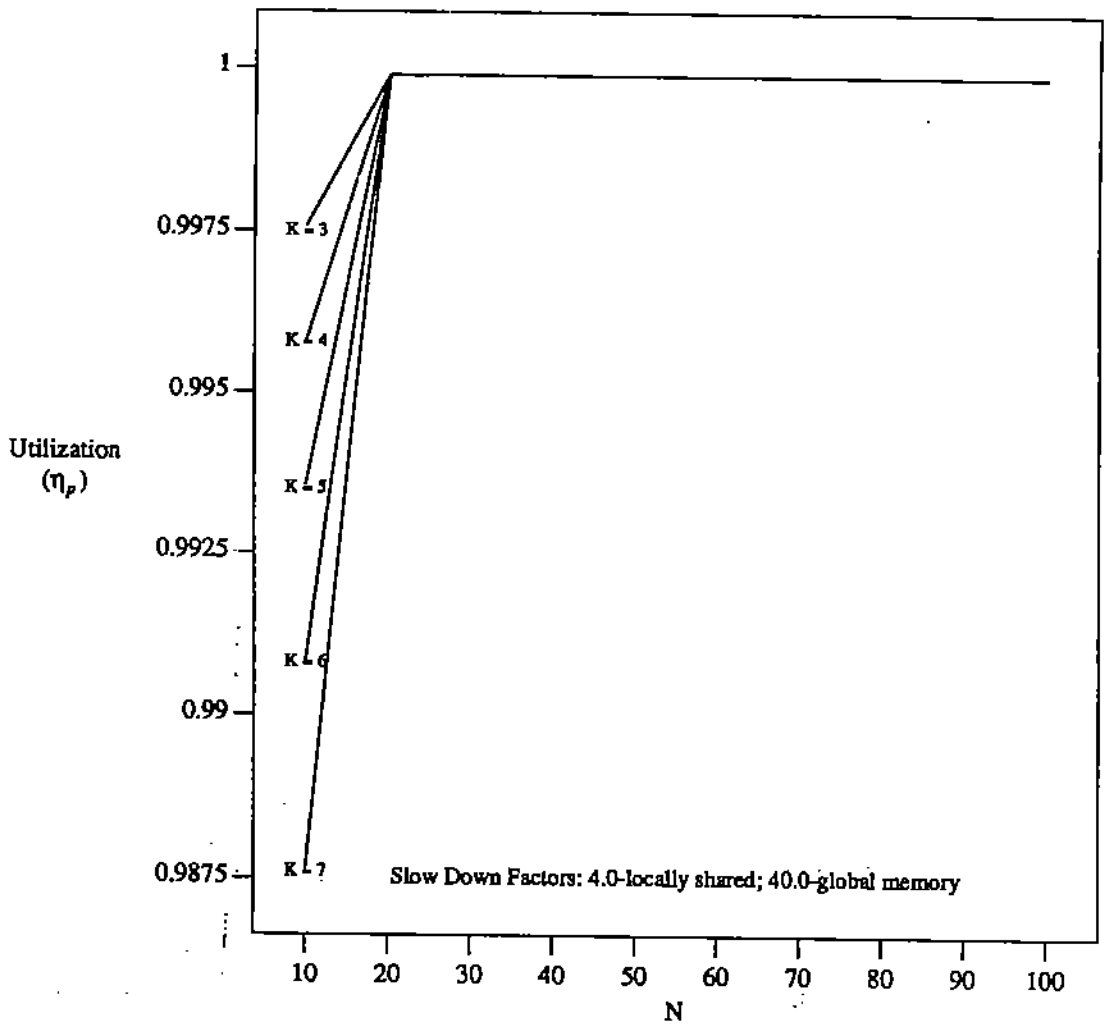

Figure 6.6. The utilization of a processor versus N and K for the optimistic 3D case of slowdown factors (4.0 locally shared, 40.0 global memory).

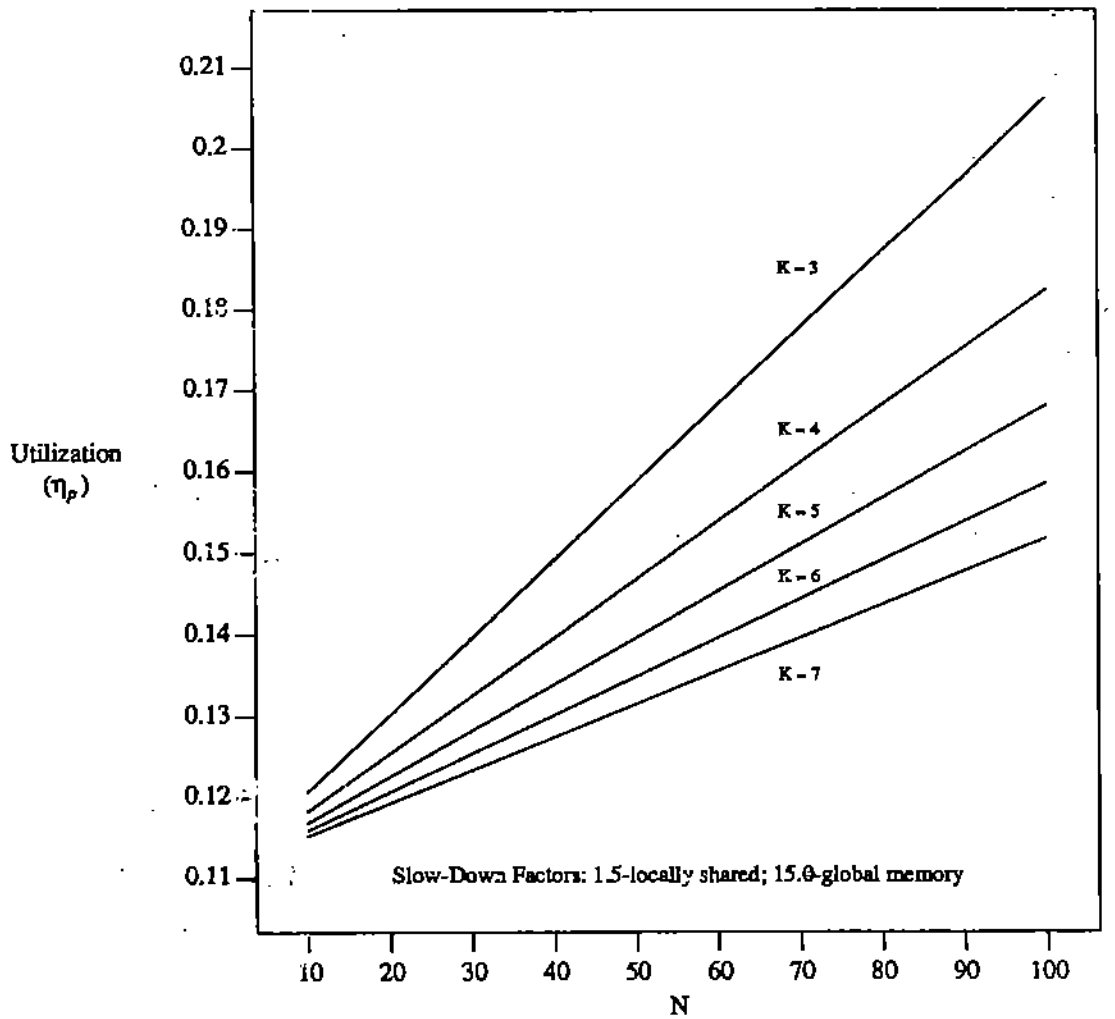

Figure 6.7. The utilization of a processor versus N and K for the pessimistic 3D case of slowdown factors (1.5 locally shared, 15.0 global memory).

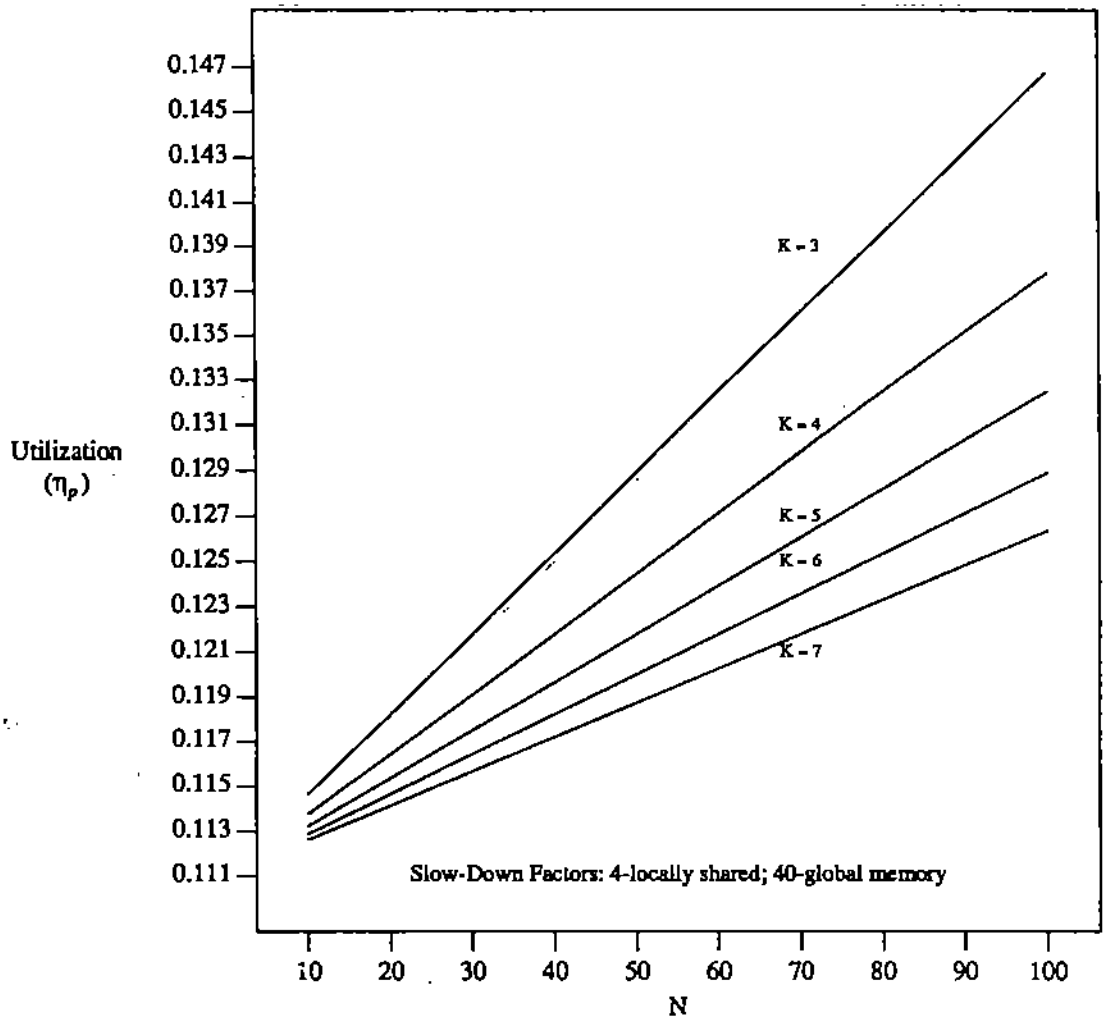

Figure 6.8. The utilization of a processor versus N and K for the pessimistic 2D case of slowdown factors (4 locally shared, 40 global memory).

6. THE MODEL WITH GROUP SYNCHRONIZATION

A model of a Multi-FLEX system with group synchronization is presented in Figure 7. The group synchronization is necessary in order to model the semantics of cooperation among different processors which operate upon different segments of a large array of data and have to exchange partial results. Two synchronization conditions are imposed. The first one requires that a processor migrating to the locally shared domain must wait for all other processors in its group to complete execution in the local domain. A group consists of three processors. The second synchronization condition is related to the transition to the global domain. In this case all processors of a system must complete their execution in the locally shared domain before any of them is allowed to enter the global domain.

The introduction of these two synchronization conditions has increased the complexity of the model. Two new places as well as two new transitions appear in the SHLPN graph in Figure 7. The state space has increased from 145 to 1505 states. The methodology developed for SHLPN modeling has proved its usefulness and we were able to construct the state transition table of this system with a reasonable effort.

The places and the transitions present in the SHLPN graph in Figure 7 have the following significance:

P_1 is the "local domain" place. When this place holds tokens, the corresponding processors are active in their local domain.

P_2 is the "group synchronization" place. When this place holds tokens, the corresponding processors are waiting for synchronization with their neighbors in the same multiprocessor system.

P_3 is the "locally shared memory queuing" place. When this place holds tokens, the corresponding processors are queued for the locally shared memory.

P_4 is the "locally shared memory" place. When this place holds tokens, the corresponding processors are accessing the locally shared memory.

P_5 is the "common bus" place. When this place holds a token, the common bus is free.

P_6 is the "system synchronization" place. When this place holds tokens, the corresponding processors are waiting for synchronization with all others in the same system.

P_7 is the "global queuing" place. When this place holds tokens, the corresponding processors are queued for global memory.

P_8 is the "global domain" place. When this place holds a token, the corresponding processors are executing in the global domain.

P_9 is the "global bus" place. When this place holds a token, the global bus is free.

T_1 is the "end of local domain activity" transition. When it fires, a processor ends its activity in the local memory.

T_2 is the "group synchronization" transition. When all processors belonging to a group finish their execution in the local domain, the transition can fire.

T_3 is the "getting common memory" transition. The transition is enabled when the common bus is free and there is at least one processor waiting in the common queue.

T_4 is the "end of execution in the shared domain" transition. When it fires, a processor ends its activity in the shared domain.

T_5 is the "system synchronization" transition. When all processors of a system finish their execution in the locally shared domain, this transition can fire.

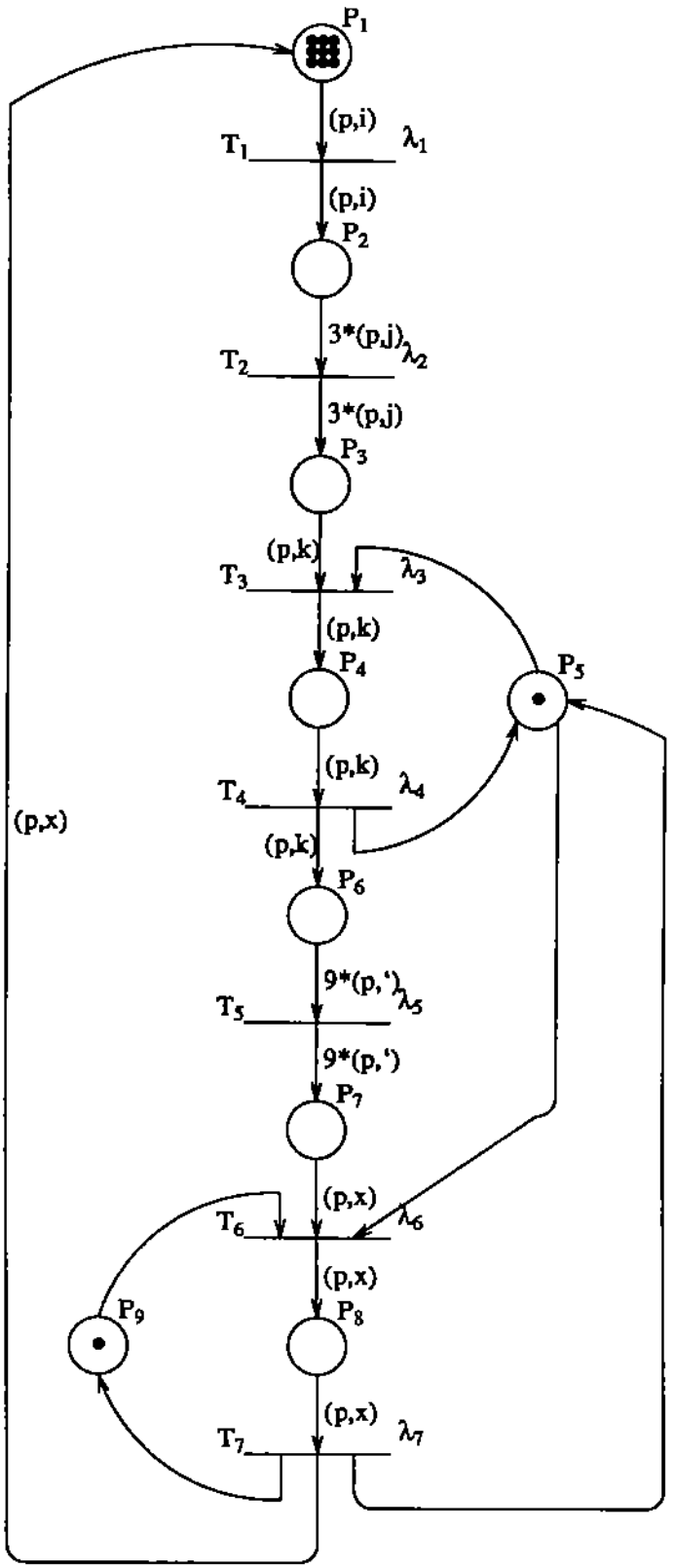

Figure 7. The SHLPN model with group synchronization.

T_6 is the "getting global memory" transition. The transition is enabled when the common bus and the global bus are free and there is at least one processor waiting in the global queue.

T_7 is the "end of global domain" transition. When it fires, a processor ends its activity in the global domain.

We are in the process of solving the equations of the model with group synchronization.

7. REFERENCES

[1] John R. Rice, "Multi-FLEX machines: Preliminary report", CSD-TR-612, Computer Science, Purdue Uiversity, June 1986

[2] Chuang Lin and Dan Cristian Marinescu, "Stochastic high level Petri nets and applications", CSD-TR-613, Computer Science, Purdue University, July 1986.

[3] Dan C. Marinescu and Chuang Lin, "Preliminary results on multiprocessor modeling and analysis using stochastig, high level Petri nets", Twenty-Fourth Allerton Conference on Communication, Control and Computing, October 1986.

[4] G. Rodrique and J. Simon, ''A generalization of the numerical Schwarz algorithm'', In *Computing Methods in Applied Science and Engineering* (Glowinski and Lions, eds.) North-Holland (1984), 273–283.

[5] K. Miller, ''Numerical analogs to the Schwarz alternating procedure'' *Numer. Math.* 7 (1965), 91–103.

[6] L.V. Kantorovich and V.I. Krylov, *Approximate Methos in Higher Analysis.* Noordhoff, 4th edition (1958).

[4] W.P. Tang, *Schwarz Splitting, A model for parallel computations,* Ph.D. thesis, Stanford University (1986).