C.E.A.
Centre d'Etudes de SACLAY
Direction des Réacteurs Nucléaires
Département de Mécanique et de Technologie

Date : le

Réf. DM

DEMANDE D'AUTORISATION DE COMMUNICATION

95001054

PUBLICATION - Titre :
Auteur(s) :
Nature de la Publication :
Editeur(s) (CEA, AIEA, OCDE,...) :

COMMUNICATION -
Titre : MASSIVELY PARALLEL ALGORITHM FOR THE COLLISION PROBABILITIES CALCULATIONS IN APOLLO II CODE USING PVM LIBRARY

Auteur(s) : Z. STANKOVSKI

Titre de la Conférence : INTERNATIONAL CONFERENCE ON METHEMATICS AND COMPUTATIONS, REACTORS PHYSICS, AND ENVIRONMENTAL ANALYSES.

Lieu et Date :PORTLAND (Oregon) - 30 avril - 4 mai 1995

Organisateur(s) : ANS

Date de remise des textes : Septembre 1994
Commentaires :

Ce texte a-t-il déjà été publié ? NON
Si OUI : Référence de la publication antérieure :
Cette publication contient-elle, à votre avis, des informations brevetables OUI NON

Date de la demande : 28/3/95

Chef de Laboratoire : G. MAHONNIERE

Le Chef de Service : E. PROUST

DEMANDE D'AVIS (éventuellement) :

a) - C.P.I./Saclay OUI NON
b) - Chargé de Mission pour les affaires industrielles OUI NON
c) - SYFRA OUI NON
d) - Partenaires concernés (EDF, FRA, Dpts de la DRN) (joindre photocopie) **Date de la demande :**
e) - Autres unités opérationnelles **Date de la demande :**
f) - Autres avis demandés par le Chef du DMT :

Avis du : **(le cas échéant)**
Date :
les avis sont à envoyer au DMT/DIR
Date d'arrivée au DMT/DIR

Décision DMT n° Date :

Le Chef de Département : A. HOFFMANN

VOL 2 [illegible]

P.J. : Un texte complet
Copies : DMT/DOC
~~DMT/DIR~~
Note : Copie autorisation + résumé + Texte à INSTN/MIST/CIRST

We regret that some of the pages in this report may not be up to the proper legibility standards, even though the best possible copy was used for scanning

A MASSIVELY PARALLEL ALGORITHM FOR THE COLLISION PROBABILITY CALCULATIONS IN THE APOLLO-II CODE USING THE PVM LIBRARY

Zarko Stankovski

Commissariat à l'Energie Atomique
Service d'Etudes de Réacteurs et de Mathématiques Appliquées
Département de Mécanique et Technologie
Direction des Réacteurs Nucléaires
Centre d'Etudes de Saclay - 91191 Gif sur Yvette - France
tel. (33) 1 69 08 40 64 fax (33) 1 69 08 94 90
E-mail: stan@oldwoman.serma.cea.fr

ABSTRACT

The collision probability method in neutron transport, as applied to 2D geometries, consume a great amount of computer time, for a typical 2D assembly calculation about 90% of the computing time is consumed in the collision probability evaluations. Consequently RZ or 3D calculations became prohibitive. In this paper we present a simple but efficient parallel algorithm based on the message passing host/node programmation model. Parallelization was applied to the energy group treatment. Such approach permits parallelization of the existing code, requiring only limited modifications. Sequential/parallel computer portability is preserved, which is a necessary condition for a industrial code. Sequential performances are also preserved.

The algorithm is implemented on a CRAY 90 coupled to a 128 processor T3D computer, a 16 processor IBM SP1 and a network of workstations, using the Public Domain PVM library. The tests were executed for a 2D geometry with the standard 99-group library. All results were very satisfactory, the best ones with IBM SP1. Because of heterogeneity of the workstation network, we did not ask high performances for this architecture. The same source code was used for all computers.

A more impressive advantage of this algorithm will appear in the calculations of the SAPHYR project (with the future fine multigroup library of about 8000 groups) with a massively parallel computer, using several hundreds of processors.

I. INTRODUCTION

The collision probability (CP) method[1] in neutron transport, as applied to arbitrary 2D XY geometries, like with the TDT[2] module in APOLLO-II[3], consumes a great amount of computer time. Consequently RZ or 3D calculations became prohibitive. Fortunately, this method is very suitable for parallelization. Massively parallel computer architectures, especially MPMD machines, bring a new breath to this method.

In a recent communication[4], the interface current method was parallelized by simultaneous calculations of the sub-assemblies. Also a CM5 implementation of the CP method, using a hostless programming model and a specific message passing library, was presented[5]. These algorithms are quite efficient, but not well suited for implementation in a production code.

The CP method for 2D geometries consists, for each energy group, on a double integration on R and φ over the neutron trajectories. For each trajectory one must calculate the contribution to the Pij matrix of every traversed region i in the basic domain with every traversed region j within the optical distance $\tau \leq \tau max$.

The quadrature formula consisting of the number of trajectories and their geometrical length, depends on the geometrical complexity and the cross sections of the problem treated. A complex geometry with many regions needs thousands or tens of thousands of trajectories. The transparency of the fast energy groups requires very long trajectories (in length, and consequently in number of intersections), to the order of several hundreds intersections. The interface current approximation can significantly reduce the length of the trajectories, but a finer quadrature formula is necessary in order to achieve sufficient accuracy for the escape and transmission probabilities.

Massively parallel architectures allow simultaneous calculations on many processors. One can imagine several parallelization techniques, each one with a different degree of complexity. However, it is necessary to find the equilibrium between different computational parameters. This is done by minimizing the following parameters: the quantity of communications, repeated calculations, repeated memory storage and unoccupied processors.

It is also important to develop a parallel algorithm independent (as much as possible today) of the computer architecture, and especially independent of the number of processors. But our first criterion was to develop a parallel algorithm applicable to a production code. This means:

a. The modifications should be limited and should not need reorganisation of the entire code,

b. The entire code should work on the parallel computer,

c. Sequential/parallel computer portability should be preserved,

d. Sequential performances should be preserved.

II. PARALLELIZATION ALGORITHM

Our reflection is based on three facts: a) APOLLO-II already works on CRAY 90, IBM RISC and workstations, b) for a typical 2D assembly calculation about 90% of the computing time is consumed in the CP evaluations and c) the CP module represents about 10% of the source code.

The easier approach is the parallelization of the energy group treatment. In the classical, sequential, approach the task is split into two steps: trajectory tracking and CP evaluation. Tracking is done only once, and the CP calculation is repeated for each energy group, with the same trajectories, and changed cross-sections. The optical length of the trajectories depends on the cross-sections. Thus one has to track very long trajectories, in order to obtain the necessary optical length for the fast groups. Because of the amount of data, the trajectories should be stored on disk and reread for each group.

The parallel algorithm distributes several energy groups to each processor. Trajectory tracking is repeated by each processor for each group, and is thus not excessively penalizing. If this is not done one must either execute tracking on one processor, while all others are unoccupied, or parallelize the tracking, which is difficult. Another advantage is a dramatic decrease of communications. The same geometrical data are broadcast to all processors, and the corresponding total cross-sections are sent to every processor; as a result there is no tracking data communications between processors nor with the disk. One more advantage is that there is no overtracking because only the necessary optical length of the trajectories is calculated for each group.

The Pij matrix calculation time is different for each group: the high energy groups are more transparent, the neutron trajectories are longer, and so the computation time is greater. In order to equilibrate the work of the processors and to achieve the best occupation ratio, we distribute the energy groups over different processors in order of their transparency. We define the transparency of the assembly for a given energy group g as:

$$T_g = \frac{\sum_{i=1}^{N} V_i}{\sum_{i=1}^{N} V_i \, \Sigma_{gi}}$$

where V(i) is the volume of the region i, $\Sigma(i,g)$ is the total cross section of the region i (and group g), and N is the total number of regions.

The most adapted programmation model for the necessary MPMD treatment of our problem is message passing. We use the PVM library, which is a standard in fact, close to the future MPI standard.

The flow-chart diagram is shown the Figure 1. Two main programs exist. The first one, **mainH**, runs on the host. It is the users choice whether it will run in sequential or in parallel mode. The sequential run is the same as before parallelization. The parallel run is also the same, before and after the CP evaluation, the only difference is that the CP evaluation is replaced by communications with the nodes. The second one, **mainN**, runs on NPROC processors and computes the CP's.

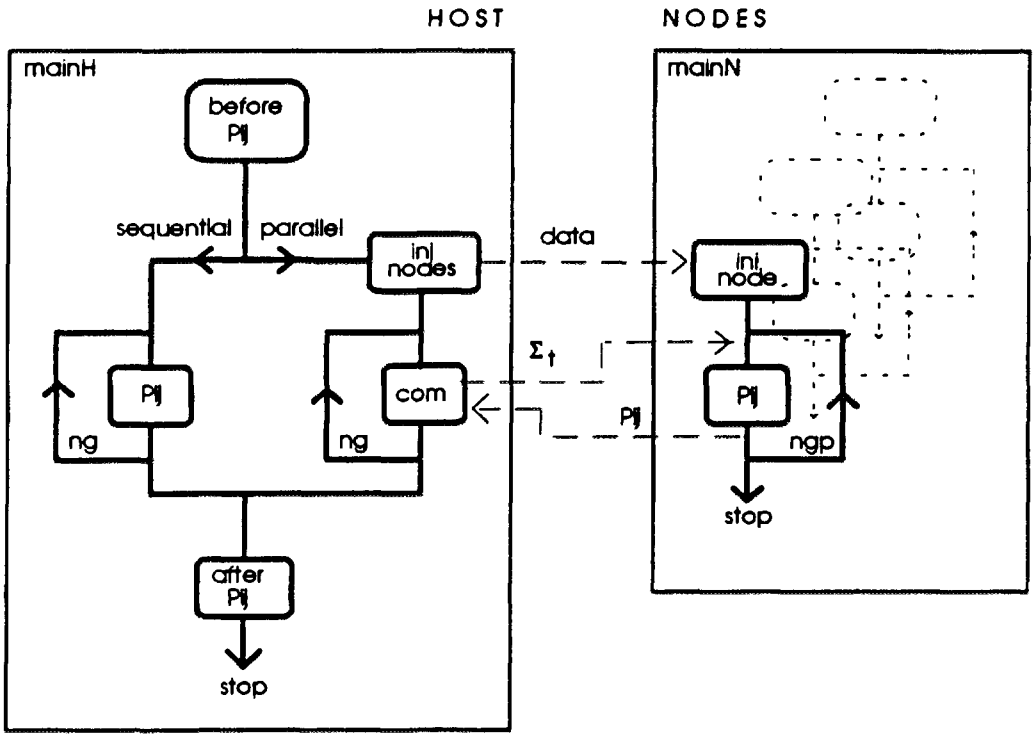

Figure 1. Flow-chart diagram

The parallel part of the code begins by broadcasting the geometrical data to all processors (from "ini nodes" to "ini node"). Afterwards the loop "com" manages the communications between the host and the nodes. The cross sections of the first (more transparent) NPROC energy groups are distributed to all processors. Then, every time a processor completes the CP evaluation for a group it returns them to the host, which host sends back the cross sections of the next more transparent energy group. When there are no more groups, the host asks each node to stop. This way the nodes will be occupied only the necessary time, and we can expect that calculation times for all processors will be nearly equal.

In order to make optional tracking data storage, only a slight restructuration of the code was necessary. Sequential/parallel computer portability was preserved and sequential performances were not deteriorated.

Besides of the host we used up to NPROC=32 processors on CRAY T3D, 8 on IBM SP1 and 16 workstations on a HP network. The standard APOLLIB-II library has NG=99 groups: NG>NPROC and NG/NPROC is not an integer. It is clear that the greater the ratio NG/NPROC the more efficient the algorithm. For NG<NPROC, one should look for a different parallelization technique.

The use of architecture like CRAY 90 - T3D has another advantage for a code like APOLLO-II. Once the CP calculation parallelized, the next more important time consumer is the flux module. The work can be favourably shared between the numerous scalar processors of T3D for the CP numerical integrations and the vector CRAY 90 processor for the matrix inversions of the flux calculations.

III. NUMERICAL RESULTS

A dilated RSM (Tight Lattice) hexagonal PWR assembly, Figure 2, is used to demonstrate the proposed algorithm. It is composed of a central water-hole cell, a tube-guide cell, a fertile cell and 13 typical fuel cells. The motif is divided to 94 regions. Reflexion boundary conditions are assumed. The quadrature formula used, ΔR=0.2 cm and $\Delta\varphi$=30°, required tracking of 147 trajectories.

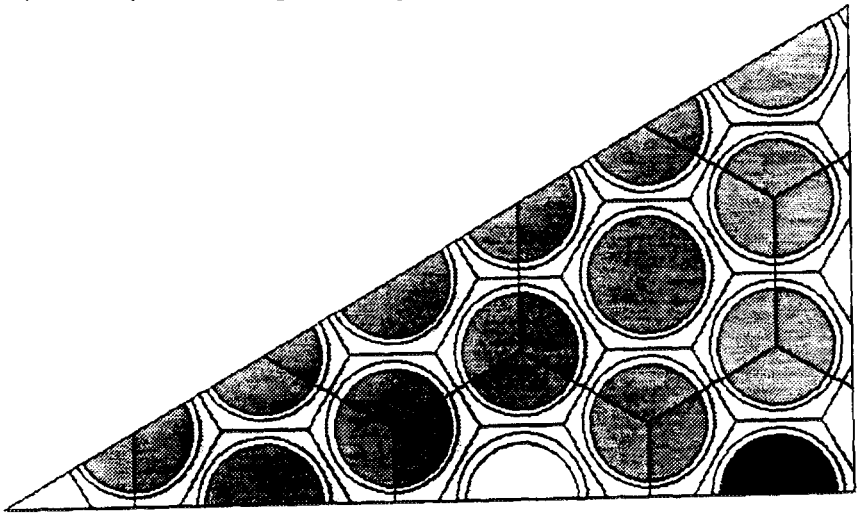

Figure 2. RSM assembly

The algorithm was tested on three architectures: CRAY T3D, with a CRAY 90 as host, IBM SP1 and HP workstations network. On the network we used: HP712/80 (5),

and HP712/60, HP715 and HP710, (11 of each); in the calculations which needed only one computer we used a HP712/80, for the parallel calculations the host was HP712/60.

For different test calculations we present on Table 1 the total **WC** (Wall Clock) and the maximum occupied processor and total **CPU** (Central Processor Unit) times, in seconds. In parallel computing it is essential to control communications between processors, so we show also the amount of data exchanged with the disk (in and out) and between the host and the nodes (pvm), in megawords. The only criterion we find to estimate the effect of the host/node communications was the WC time. The CRAY 90 and IBM SP1 computers work in time sharing mode, and there is always unknown number of processes running on different workstations, so we measured the WC times by running several times the same job, during the off-peak hours, and keeping the best results.

First we executed our job in sequential mode on CRAY 90, one IBM SP1 processor and HP712/60 workstation. Two series of calculations were done: with only one tracking (**OT**) for all groups (to a distance of Lmax=100 cm) and with repeated tracking (**RT**) for each group (to a distance of $\tau max=10$). The OT case corresponds to a standard sequential calculation, as before parallelization. The RT case permits to measure the additional calculations of the parallelized code.

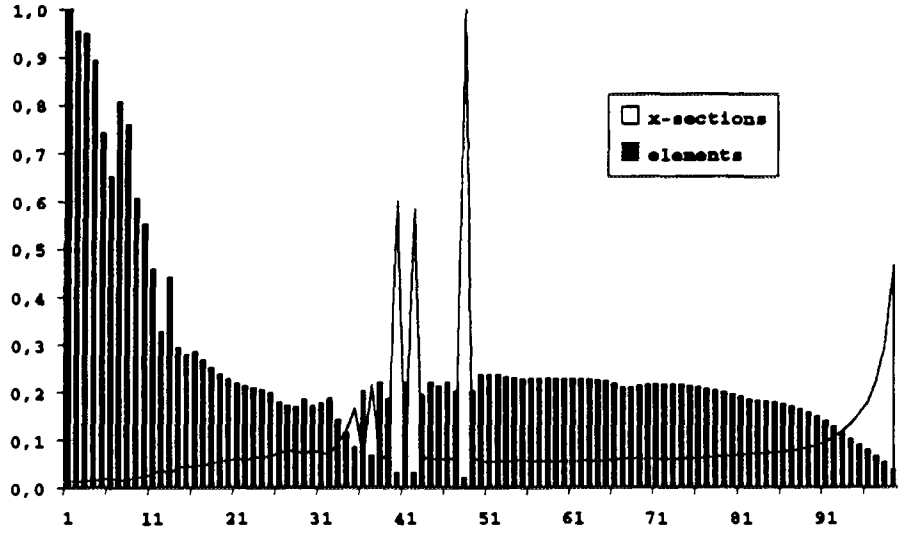

Figure 3. Mean cross section of the assembly and the number of intersections for each energy group, normalized to 1.

In order to have Lmax=100 cm for the OT case, the total number of trajectory intersections is 173,467. To have τmax=10 for the RT case, it is necessary to have 182,054 intersections in the first group and only 4,191 in group 48. Note that the first group (also the second, but only the first and the second in our case) are not calculated with enough precision in OT case. Figure 3 shows, normalized to 1, the assembly averaged cross section and the number of intersections for each energy group.

At this point we can make two conclusions: **a)** the difference between WC and CPU times is very important for the OT calculation on CRAY and HP, while it is negligible for the others, **b)** for the IBM SP1 computer the RT time is shorter than the OT one. We presume that these two phenomena are due to different IO treatment by different architectures.

In order to show the difference between the CRAY 90 and T3D processors we continue by one node "parallel" calculations. The CRAY time nearly doubled: CRAY 90 is two times faster than ALPHA processor.

Parallel calculations are done on 2, 4, 8, 16 and 32 nodes on CRAY T3D, up to 8 processors of IBM SP1 and up to 16 on HP. Figure 5 shows the distribution of CPU times, in seconds, for every group on each processor. The host, processor N° 0 on the figure, reads the geometrical data, cross sections and the tables of Bickley functions, and manages the work.

		C90	T3D (sec)		IBM SP1 (sec)			HP (*) (sec)			data transf. (Mw)		
			max	cpu		max	cpu		max	cpu			
		wc	cpu	total	wc	cpu	total	wc	cpu	total	in	out	pvm
Only one tr.		134		73	209		200	102		78	34,6	0,35	
Repeated tr.		148		146	151		147	151		149			
	1	302	279,9	287	158	148,8	152	173	149,9	155			1,78
host +	2	168	146,8	300	82	74,6	152	99	90,0	183			1,79
	4	98	74,2	301	50	42,1	152	58	47,3	171			1,80
nodes	8	42	35,9	287	25	20,7	154	37	30,4	229			1,83
	16	24	17,8	285				27	17,0	270			1,90
	32	27	11,4	303									2,03

(*) 5*hp712/80 + 11*(hp712/60 + hp715 + hp710)

Table 1. Total WC and the maximum occupied processor and total CPU times, in seconds

The speed-ups compared to the original (tOT/tm) and modified (tRT/tm) code are given in Table 2 and Figure 4. We define the speed up as: tXX/tm, where tXX is the OT and RT time in sequential mode and tm is the time spent by the most occupied processor, for the WC and CPU times. The "ratio" is the CPU speed-up divided by the number of nodes. It was difficult for us to make the sum: 1 C90 + 32 ALPHA = 33, that is why we did not counted the host in the ratio. The other reason is that for all computers the host works in time sharing mode, it is not unoccupied when it does not work for us. The actual speed-up is tOT/tm, comparison with the original, non parallelized code - this is the speed-up seen by the code user. Note that is a severe criterion, the practice in the

literature is to use the comparison with the one-processor run - more interesting for the developer, so we present also the tRT/tm speed-up.

The most surprising and unexpected result is the OT CPU speed-up of 9,67 for 8 processors on IBM SP1 (8,36 for the WC time), due to the lack of disk input/output. As shown on Figure 6, in the OT case, the code have to read the voluminous tracking data, for each group, even if for thermal groups it needs only a very small amount of these data.

The CRAY speed-up seems less satisfactory, there is a factor of two between the original and the parallelized code. Note also that we compare the sequential time on CRAY 90 processor with parallel time of ALPHA processors. But, up to 16 processors, the work is always well equilibrated with a constant ratio of 0.5 for RT and 0.25 for OT speed-up. The 32 processor time distribution diagram on Figure 4, shows why the corresponding speed-up is less than 7: the long computation time of fast groups could not be equilibrated in a 32 partition.

There is also a factor of two between the sequential and parallel calculations on the HP workstation network. The ratios are less uniform due to the performance heterogeneity of the workstations used. Nevertheless, results show the incontestable interest to use a workstation network for parallel calculations.

The great difference of calculational times permits a very good work equilibration. The CRAY T3D NPROC partition is reserved for the same user, while every processor of the IBM SP1 or the workstations works in the time sharing mode. The proposed algorithm distributes group calculations dynamically. Even if the availability of all processors is not the same, the WC times will be equilibrated: if some processors are busy with work for other users, the host demands them less work (see processors 4, 5 and 6 on the first diagram of Figure 5.).

IV. CONCLUSIONS

This work allowed us to demonstrate the feasibility and the advantages of the parallelization of the CP method. It also allowed us to appreciate tree different parallel architectures, from the point of view of our problem: a production code using the integral method.

The host/node programming model is well adapted for parallelization of existing, large, production codes, one should modify only the concerned parts. If the code already works on the same computer, which is the case of APOLLO-II with CRAY 90, IBM RISC and the workstations, the developers job is much easier.

The IBM SP1 results were better than expected, but it is also evident that important progress of the CRAY's operating system may be predicted in the near future. Today the principal drawback of CRAY is the obligation to reserve the partition (number

of processors to be used) for all the time of the execution of a job. So, if 90% of the time is used for CP calculations, for a speed-up of 9 in the CP's, 50% of the APOLLO-II calculation time will be done on sequential mode, while all T3D processors will be unoccupied.

If the best speed-up can be, evidently, obtained for NPROC=NG, the better occupation ratio and the most reasonable choice is around NPROC=NG/5. In our case, the speed-up will be the same for NPROC≥32.

This work shows that performances are very architecture dependent. But one should have in mind that performances depend also on the geometry, the quadrature formula, the boundary conditions and the cross sections of the treated assembly.

V. REFERENCES

1. R. SANCHEZ and N.J. McCORMICK, "A Review of Neutron Transport Approximations," Nuc. Sci. Eng. **80**, 481 (1981).

2. R. SANCHEZ and Z. STANKOVSKI, "SILENE & TDT: A Code for Collision Probability Calculations in XY Geometries," Proc. 1993 ANS Annual Meeting, June 20-24, 1993, San Diego, California.

3. R. SANCHEZ, J. MONDOT, Z. STANKOVSKI, A. COSSIC and I. ZMIJAREVIC, "APOLLO-II: A User-Oriented, Portable, Modular Code for Multigroup Transport Assembly Calculations," Nuc. Sci. Eng. **100**, 352 (1988).

4. W. J. WILSON, J. L. VUJIC, A. G. GU, "Parallel Multiple-Assembly Calculations in GTRAN/2M," Trans. Am. Nucl. Soc. **69**, p. 204(1993).

5. Z. STANKOVSKI, "First Massively Parallel Algorithm to be Implemented in APOLLO-II Code," Int. Conf. on Reactor Physics and Reactor Computations, Tel Aviv Hilton, January 23-26 1994

		C90 T3D			IBM SP1			HP		
		wc	cpu	ratio	wc	cpu	ratio	wc	cpu	ratio
tOT/tm										
Only one	tr.	1,00	1,00		1,00	1,00		1,00	1,00	
Repeated	tr.	0,91	0,50		1,38	1,36		0,68	0,53	
	1	0,44	0,26	**0,26**	1,32	1,34	**1,34**	0,59	0,52	**0,52**
host +	2	0,80	0,50	**0,25**	2,55	2,68	**1,34**	1,03	0,87	**0,43**
	4	1,37	0,98	**0,25**	4,18	4,76	**1,19**	1,76	1,65	**0,41**
nodes	8	3,19	2,03	**0,25**	8,36	9,67	**1,21**	2,76	2,58	**0,32**
	16	5,58	4,11	**0,26**				3,78	4,61	**0,29**
	32	4,96	6,42	**0,20**						
tRT/tm										
Only one	tr.	1,10	2,00		0,72	0,74		1,48	1,90	
Repeated	tr.	1,00	1,00		1,00	1,00		1,00	1,00	
	1	0,49	0,51	**0,51**	0,96	0,99	**0,99**	0,87	0,99	**0,99**
host +	2	0,88	0,99	**0,50**	1,84	1,98	**0,99**	1,53	1,65	**0,83**
	4	1,51	1,97	**0,49**	3,02	3,51	**0,88**	2,60	3,14	**0,78**
nodes	8	3,52	4,06	**0,51**	6,04	7,13	**0,89**	4,08	4,89	**0,61**
	16	6,17	8,22	**0,51**				5,59	8,76	**0,55**
	32	5,48	12,83	**0,40**						

Table 2. Speed-ups, compared with original and modified sequential calculations

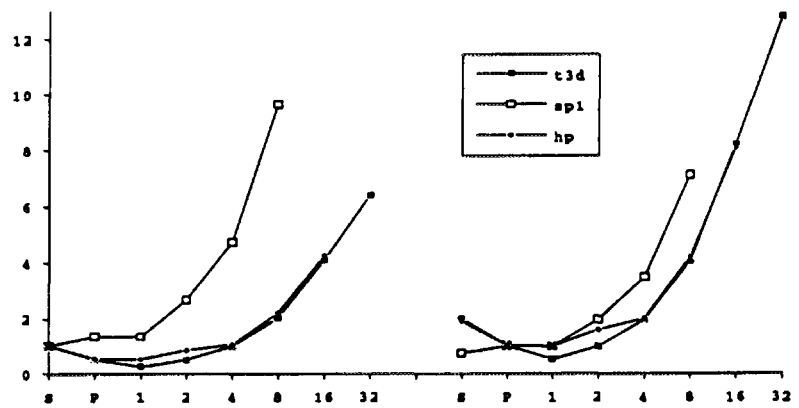

Figure 4. Speed-ups, compared with original (S) and modified (P) sequential calculations

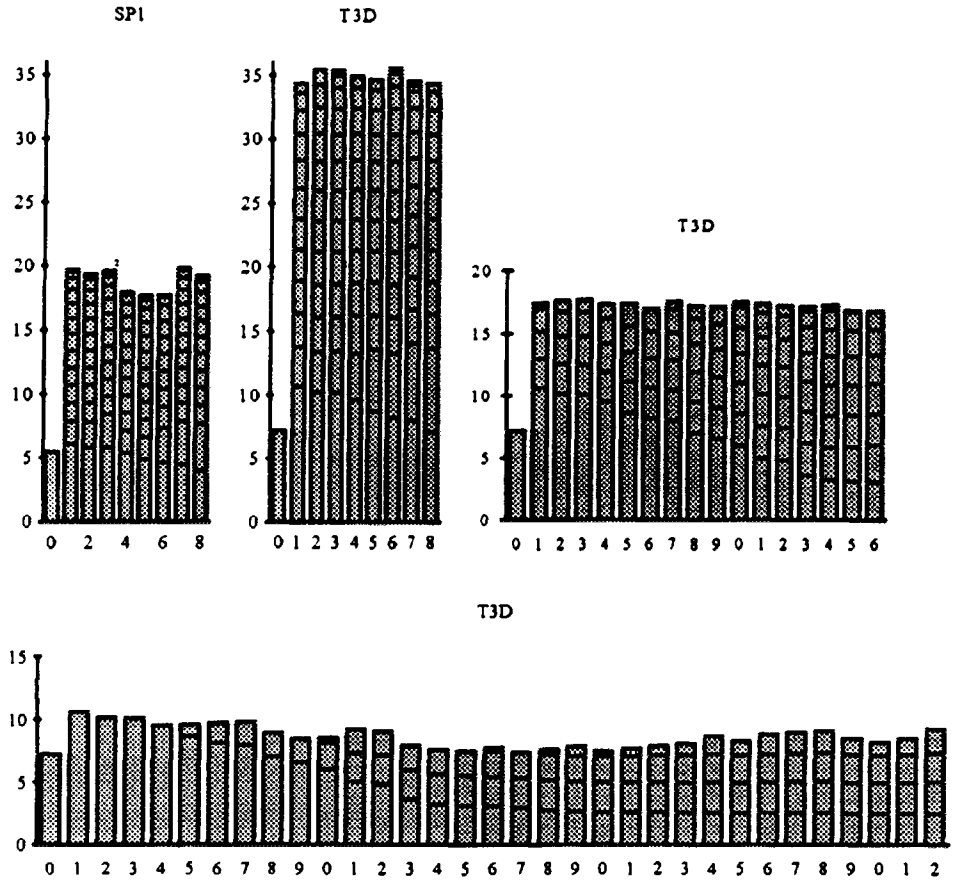

Figure 5. CPU time distributions, per processor, in seconds, for 8 groups on SP1 and 8, 16 and 32 groups on CRAY 90 - T3D.

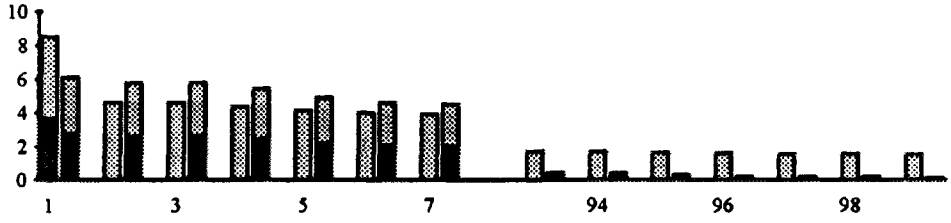

Figure 6. Computation times for first and last seven groups on IBM SP1, in seconds. OT on the left, RT on the right for each group, dark for tracking time and light for CP computations.