Computer Assisted Mechanics and Engineering Sciences, 1: 177–189, 1994.

An application study of parallel processing to the particle transport simulation

Makoto Sasaki
The Japan Research Institute, Nuclear Engineering Group, Ltd.,
16, Ichibancho, Chiyoda-ku, Tokyo 102, Japan

Masayuki Nakagawa, Takamasa Mori
Japan Atomic Energy Research Institute, Tokai Research Establishment,
Tokai Mura, Ibaraki-ken 319-11, Japan

(Received August 1, 1994)

New methods of the neutron and photon transport Monte Carlo simulation suitable for vector computers have been investigated and general purpose multigroup and continuous energy codes have been developed. On vector supercomputers, the codes achieved high speed-up gains of an order of ten or more compared with the conventional scalar codes. To achieve more speed-up, the Monte Carlo codes are applied to three types of parallel processing environments; (1) a massively parallel computer, (2) a vector-parallel type supercomputer, and (3) a cluster of workstations connected to a network. On the massively parallel computer and the vector-parallel supercomputer, speed-ups almost proportional to the number of processors are achieved by simply assigning particles uniformly to each processor, and the speed-up with the vector-parallel supercomputer is enhanced by vector processing. On the other hand, in the workstation cluster, the computational power of each workstation may differ and the simple particle assignment may not be successful. By modifying particle assignment methods, effective parallel processings are made possible in such an environment.

1. INTRODUCTION

The particle transport Monte Carlo method is widely used as a powerful tool in nuclear engineering. Neutron and/or photon transport Monte Carlo codes are used to obtain solutions as precisely as possible in reactor core neutronics analyses, radiation shielding calculations and design calculations of radiation measurement equipments, etc. The Monte Carlo method, however, generally requires much computation time to reach an acceptable statistical uncertainty level even if the fastest large computers now available are used.

Fortunately, neutrons and photons can be treated as non-interactive particles so we can simulate flights or reactions of more than one particle concurrently, that is, the problems are suited for parallel processing.

Vector processing is a SIMD type parallel processing. General-purpose Monte Carlo codes, however, have shown speed-ups of < 2 in spite of efforts to vectorize conventional codes such as MORSE-CG, KENO-IV, VIM and MCNP [1, 2, 3, 4]. The algorithms used in the conventional Monte Carlo codes are ill-suited for vectorization because the statistical nature of the calculations requires coding a small number of loops and many conditional "IF" statements which inhibit vector processing. However, it was demonstrated that Monte Carlo calculations could be successfully vectorized by adopting appropriate algorithms for vector computers [5, 6, 7, 8].

We developed two general-purpose codes; GMVP [9, 10, 11], which uses multigroup cross-section data, and MVP [12], which uses continuous energy cross-section data. The codes accomplished a speed-up of about ten to several tens on the vector supercomputers FACOM VP-100 and VP-2600 compared with their scalar versions or other conventional scalar codes in many problems including

reactor core analyses, criticality calculations, fusion reactor analyses and shielding calculations. The codes are designed to be applicable to various types of nuclear engineering calculations by adopting a powerful geometry description method (combinatorial geometry and lattice geometry) and several variance reduction methods. They have a flexible user-interface which makes them easy to use for unaccustomed users. The main capabilities of the codes, which include most functions for production use, are as follows:

1. Problem to be solved: eigenvalue problems of neutron transport, and fixed source problems of neutron and/or photon transport.
2. Geometry description: combinatorial geometry with rectangular and hexagonal lattices (available geometry objects: right parallelepiped, arbitrary polyhedron, cylinder, sphere, cone, ellipsoid, triangular prism, hexagonal prism and elliptic torus)
3. Particle source type: fixed (energy-, angle-, time- or space-dependent) and fission sources for eigenvalue problem.
4. Cross-section type: multigroup P_L and double differential (GMVP) and continuous energy cross-sections (MVP).
5. Variance reduction technique: Russian roulette kill and splitting, importance sampling, and weight window.
6. Boundary condition: perfect and white reflection and vacuum.
7. Estimator: track length and collision for flux, reaction rates and eigenvalue. Point detector for flux and reaction rates.

Recently, applications of MIMD or non-vector SIMD parallel processing to the transport Monte Carlo calculation have been reported [21,22,23]. We have studied the speed-up of the codes in parallel processing environments including a massively parallel processor, a vector-parallel supercomputer and workstations connected as a virtual parallel machine. Our codes achieved high performance both in parallel processing and vector processing.

In Section 2 of this paper, we describe vectorization methods used in the GMVP and MVP codes and examples of calculation performed with their use. Applications to parallel processing environments are described in Section 3.

2. Vector processing in particle transport Monte Carlo calculation

2.1. Method of vectorization

Scalar Monte Carlo codes track the history of one particle at a time. In the vectorized Monte Carlo code, many particle histories are tracked simultaneously. By resolving histories into "events" such as free flight, collision and boundary crossing, each event can be presented as a vector whose components correspond to the particles waiting to be processed for that event. This vectorization method is called an "event-based" algorithm and the conventional method is called a "history-based" algorithm. The difference between the event-based and history-based algorithms is shown in Fig. 1. All vectorized Monte Carlo codes use an event-based algorithm, but differences in individual approaches to vectorization significantly affect the performance of the vectorized codes. For the MVP and GMVP codes we developed a method named "stack-driven zone selection method".

Particles have 13 descriptors: Cartesian coordinates (x, y, z), flight direction (u, v, w), weight, energy or energy group, geometry zone, time and descriptor for lattice geometry. Memory areas storing descriptors of particles are called the particle bank. Random walk processes are resolved into the following basic tasks: source particle generation, collision, free flight, next zone search, lattice, reflection, leakage or kill. A "stack" is used to memorize particles queued for a task. Connections

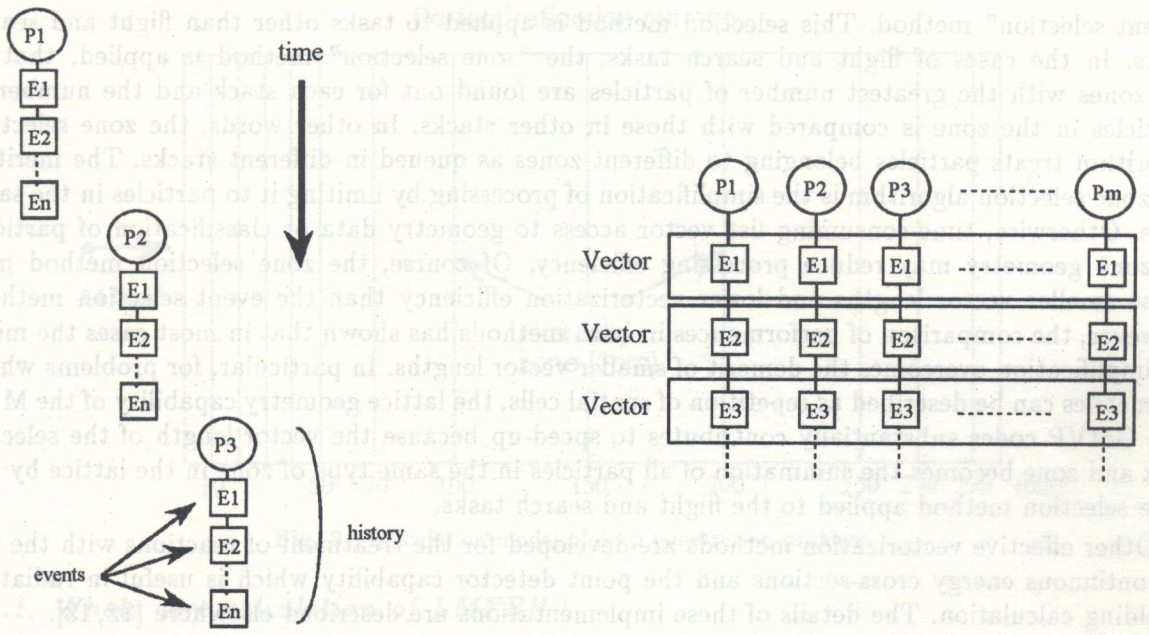

Fig. 1. Comparison of scalar and vector processing methods of Monte Carlo calculation. Left: history based scalar algorithm, right: event based vector algorithm

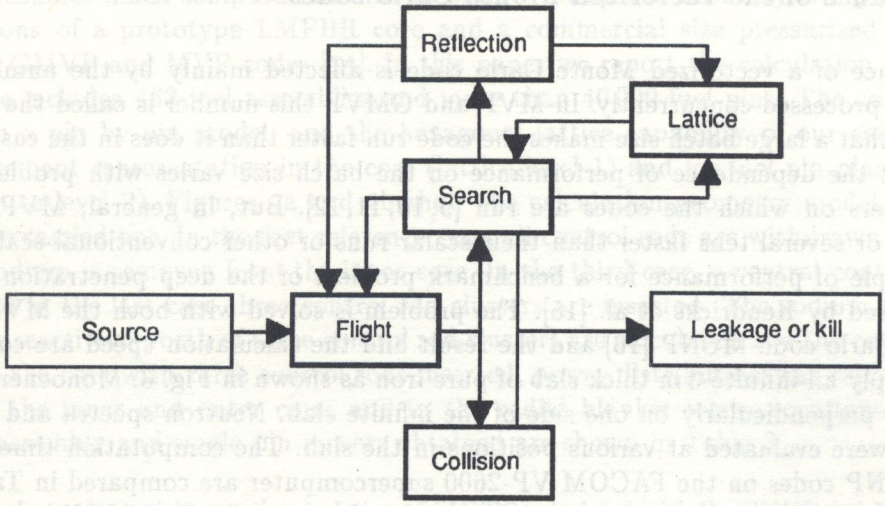

Fig. 2. Connections between Monte Carlo calculation tasks in the GMVP and MVP codes

between tasks are shown in Fig. 2. All particles are generated from fixed sources or by fission reactions and all their descriptors are determined in the source task. In the following step, the flight task calculates the free flight distance to the nearest boundary of the present zone and determines whether a particle crosses the boundary or collides with the matter in that zone. For particles crossing the zone boundary, the neighbouring zone that a particle enters is found in the search task. Depending on the next zone, the search task scatters particle pointers to three other tasks: flight, reflection or lattice. The reflection task calculates reflected flight directions on the reflective boundary and the lattice task processes particles entering or escaping from a lattice region, or moving from one lattice cell to another. The collision task determines outgoing directions and energy groups for colliding particles.

The task processing order depends on the number of particles queued in the stacks. The principle is to select a task with the greatest number of particles to process an event with the largest vector length. When the task is selected by simply comparing the total number of particles in each stack irrespective of zones they belong to or surfaces they lie on, the selection process is called the

"event selection" method. This selection method is applied to tasks other than flight and search tasks. In the cases of flight and search tasks, the "zone selection" method is applied, that is, the zones with the greatest number of particles are found out for each stack and the number of particles in the zone is compared with those in other stacks. In other words, the zone selection algorithm treats particles belonging to different zones as queued in different stacks. The merit of the zone selection algorithm is the simplification of processing by limiting it to particles in the same zone. Otherwise, time-consuming list vector access to geometry data or classification of particles by zone geometry may reduce processing efficiency. Of course, the zone selection method may cause smaller vector lengths and lower vectorization efficiency than the event selection method. However, the comparison of performances in both methods has shown that in most cases the merit of simplification overcomes the demerit of smaller vector lengths. In particular, for problems whose geometries can be described as repetition of spatial cells, the lattice geometry capability of the MVP and GMVP codes substantially contributes to speed-up because the vector length of the selected task and zone becomes the summation of all particles in the same type of zone in the lattice by the zone selection method applied to the flight and search tasks.

Other effective vectorization methods are developed for the treatment of reactions with the use of continuous energy cross-sections and the point detector capability which is useful in radiation shielding calculation. The details of these implementations are described elsewhere [12, 13].

2.2. Performance of the vectorized Monte Carlo codes

The performance of a vectorized Monte Carlo code is affected mainly by the number of source particles to be processed concurrently. In MVP and GMVP this number is called the "batch" size. It can be said that a large batch size makes the code run faster than it does in the case of a smaller batch size, but the dependence of performance on the batch size varies with problems solved or vector computers on which the codes are run [9, 10, 11, 12]. But, in general, MVP and GMVP run about ten or several tens faster than their scalar runs or other conventional scalar codes. We show an example of performance for a benchmark problem of the deep penetration of a 14 MeV neutron proposed by Hendricks et al. [15]. The problem is solved with both the MVP code and a scalar Monte Carlo code MCNP [16] and the result and the calculation speed are compared. The problem is simply an infinite 3 m thick slab of pure iron as shown in Fig. 3. Monoenergetic 14 MeV neutrons enter perpendicularly on one side of the infinite slab. Neutron spectra and the averaged cross-sections were evaluated at various positions in the slab. The computation times of both the MVP and MCNP codes on the FACOM VP-2600 supercomputer are compared in Table 1. In the second case, 10 cm thick iron and concrete slabs are placed repeatedly. The "CPU/track" means the total CPU time divided by the total numbers of collisions and boundary crossings. The speed-up of the MVP code compared with the MCNP code is 15–21 in this problem.

Table 1. Performance of MVP and MCNP for the deep penetration problem[a)]

Material	Performance	MVP	MCNP
Iron	CPU/track (μs)	2.19	47.6
	Speed-up	21.8	1.0
	Vectorization (%)	98.0	–
Iron+concrete	CPU/track (μs)	3.34	52.4
	Speed-up	15.7	1.0
	Vectorization (%)	98.0	–

a) on FACOM VP-2600

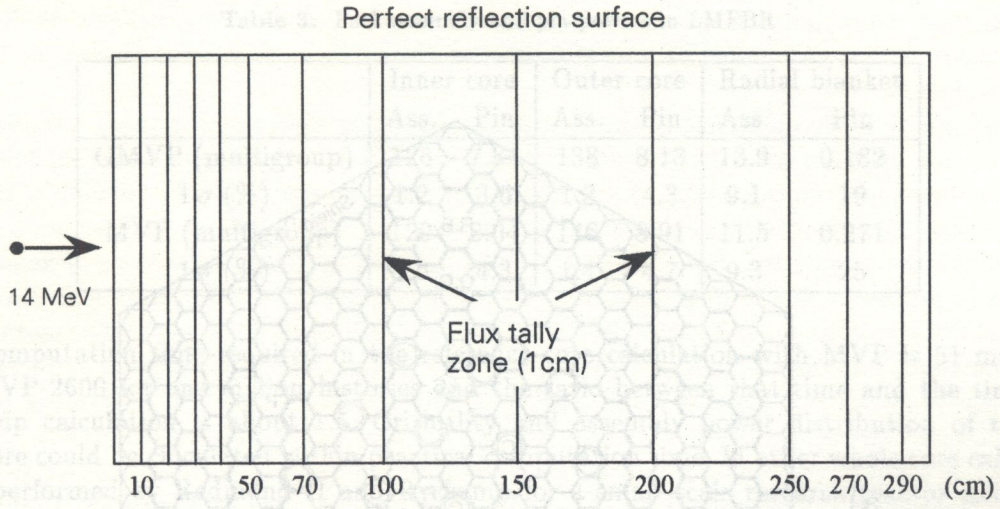

Fig. 3. Calculation model of deep penetration problem

2.2.1. Whole core calculation of LMFBR

The application of the Monte Carlo codes to whole core calculations in full detailed geometry has been limited by the fact that the reactor core generally has very complex geometry, whose calculation consumes much computation time to achieve a required accuracy. We performed whole core calculations of a prototype LMFBR core and a commercial size pressurized water reactor core with the GMVP and MVP codes [14]. In this paper we report the calculation of an LMFBR core. The core includes 463 fuel assemblies and more than 40 000 fuel pins. The core geometry is represented in a pin by pin model, and the hexagonal lattice capability of our codes is used for assembly placement representation in the core (lattice level 1) and for fuel pin placement in each assembly (lattice level 2). Figures 4a and 4b show the calculation geometry model. Four cases of calculation are carried out. In the first reference case, all control rods are withdrawn. In the second case, whole sodium is removed from the inner core. In the third case, a central control rod cluster is inserted and in the last case three control rod clusters are inserted. The sodium void reactivity worth and the reactivity worth of three control rod clusters are calculated and the results are shown in Table 2. In the core with three control rods inserted, power distribution was calculated at three assemblies in the inner and outer cores and in the radial blanket whose locations are shown in Fig. 4a. The assembly and single pin powers obtained are shown in Table 3.

Table 2. k_{eff} and reactivity worth calculation of LMFBR calculated with the GMVP and MVP codes

	Reference core	Reactivity worth		
		I.C. void	Central C.R.	3 C.R.
Number of histories	10^6	5×10^5	5×10^5	5×10^5
GMVP (multigroup)				
k_{eff}	1.0411	1.0503	1.0232	0.9989
1σ	±0.00050	±0.00077	±0.00067	±0.00066
$\delta k/kk'$		0.0084	−0.0168	−0.0405
1σ (%)		10.5	4.8	2.0
MVP (continuous energy)				
k_{eff}	1.0379	1.0461	1.0207	0.9982
1σ	±0.00055	±0.00066	±0.00066	±0.00068
$\delta k/kk'$		0.0076	−0.0163	−0.0383
1σ (%)		11.0	5.1	2.3

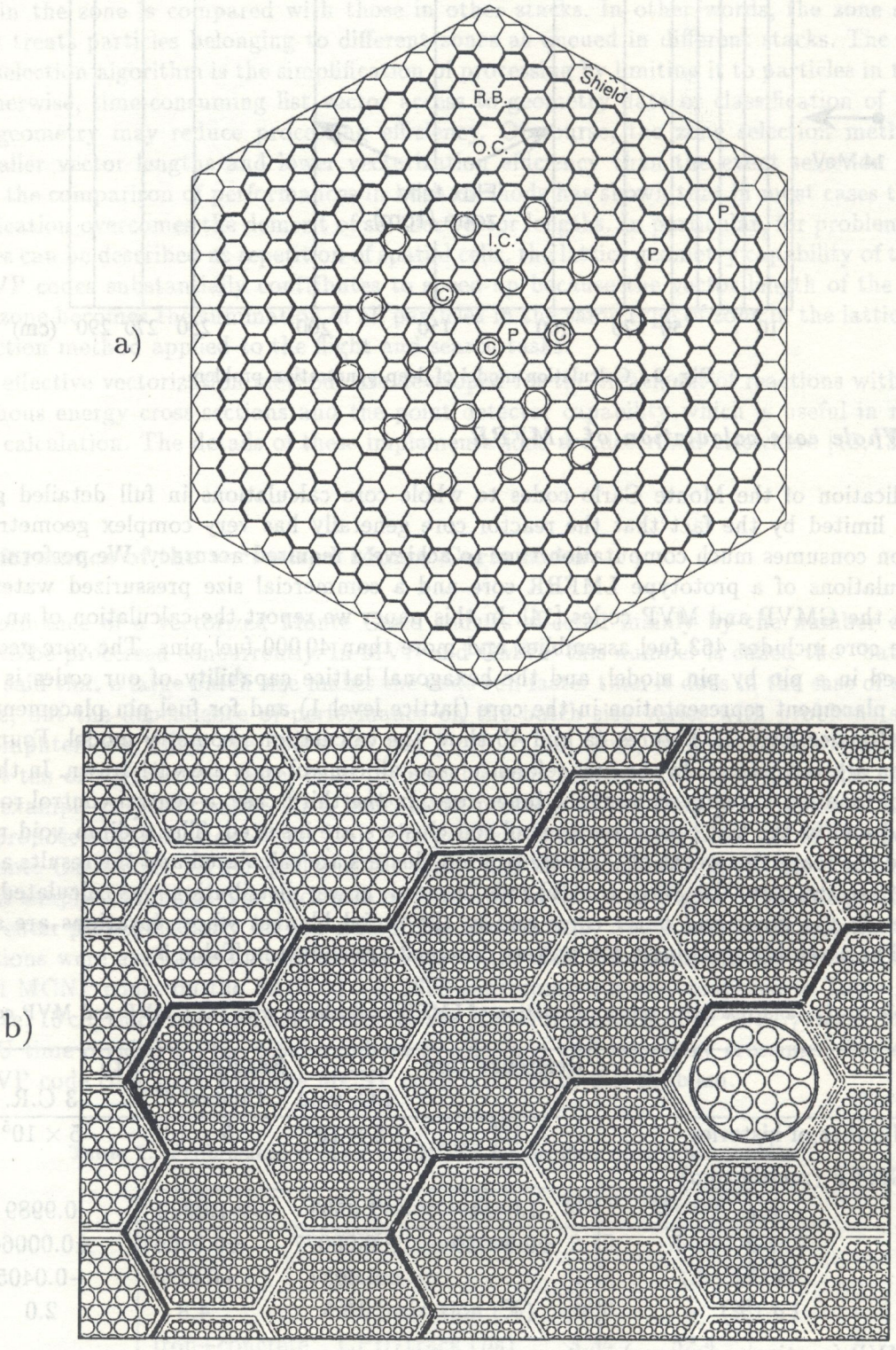

Fig. 4. Calculation model of prototype LMFBR, a) whole core lattice model, b) close-up view of prototype LMFBR

Table 3. Fuel assembly and pin powers in LMFBR

	Inner core		Outer core		Radial blanket	
	Ass.	Pin	Ass.	Pin	Ass.	Pin
GMVP (multigroup)	126	7.84	138	8.13	13.9	0.182
1σ (%)	1.2	3.6	1.3	4.3	9.1	19
MVP (multigroup)	129	8.04	146	8.91	11.5	0.271
1σ (%)	1.5	4.3	1.4	4.7	9.3	25

The computation time required in the reference core calculation with MVP is 51 minutes on FACOM VP-2600 for one million histories and the ratio between that time and the time of the multi-group calculation is about 1.5. Criticality and assembly power distribution of the whole reactor core could be calculated within practical computation time. In other whole core calculations recently performed by Redmond II and Ryskamp for a small scale research reactor ANS [17], it is reported that the MCNP code required 157 minutes of calculation time on CRAY-2 for 60 000 histories for the core with D_2O moderator. So our vectorized Monte Carlo codes are proved to be very powerful tools compared with other conventional scalar codes. However, to achieve higher accuracy in small reactivity worth calculation necessary for core design, the required number of histories is higher than the present one by one order or more.

3. Parallel processing in particle transport Monte Carlo calculation

We parallelized our codes on the following three types of parallel processor or parallel processing environment:

1. Massively parallel processor: This type of processor has many scalar processor units and memories are distributed on each processor and communication between processors is carried out by passing messages. Recently, a lot of parallel machines of this type have been provided by many computer venders. The AP-1000 [18] of Fujitsu corporation described in this paper is an example of this type.

2. Vector-parallel processor: This type of parallel processor has a relatively small number of vector processor units, each of which shares a common memory space. For example, the CRAY XMP series of Cray Research, the SX-3 of NEC corporation and the JAERI Monte Carlo Machine MONTE-4 [20] belong to this category.

3. Workstation cluster: We used the PVM (Parallel Virtual Machine) [19] developed by the Oak Ridge National Laboratory as software for constructing a virtual parallel processor with distributed memory.

3.1. Method of parallelization

There are two typical methods to perform the Monte Carlo simulation with parallel processes:

1. Each particle is assigned to one of the parallel processes and the histories of the assigned particles are processed within the process (particle or history based parallelization).

2. A process treats only a part of random walk events or tasks as shown in Fig. 2. Particles "flow" from process to process according to their events (event or task based parallelization).

The latter method seems to be suited to our codes because they are designed to be event-driven for vectorization. There is an example of adapting such a parallelization method to the Connection Machine by R.S. Baker [23].

Our parallelization scheme, however, is currently based on the first method because much inter-processor data communication might occur for the latter case on distributed memory parallel environments.

We must mention the difference in the strategies of particle assignment to each process between the fixed-source problem and the eigenvalue problem. In fixed source problems, all histories are truly independent and we can assign them to any processors in any order. For example, in processing N histories on n processors, each processor can process N/n histories independently of other processors. On the other hand, in eigenvalue calculations, the spatial distributions of fission sources are generally unknown at the beginning of calculations, so a batch of histories which are started from initial distributions must themselves generate fission sources for the next generation. Because of this constraint, histories belonging to two successive generations in eigenvalue problems cannot be processed in parallel. When the total number of histories in non-parallel processing is given as $N = g \cdot Nb$ in an eigenvalue problem, where g is the number of generations and Nb is the number of histories in a batch which is a group of histories of the same generation, there are two ways to parallelize eigenvalue calculations. One is to reduce the number of generations, for example, from g to g/n and conserve the batch-size Nb for each process, and in the other way, the number of generations on each processor is not changed but the sizes of the batch are reduced to Nb/n. Since the number of generations should not be too small to approximate the fission source distribution of a desired eigenfunction, we adopted the latter way to parallelize eigenvalue problems, but in any case, flexibility in parallel processing is reduced compared with fixed source problems.

3.2. Parallel processing on AP-1000

The AP-1000 [18] by Fujitsu corporation is a parallel computer of MIMD (Multi Instruction stream Multi Data stream) architecture. AP-1000 has up to 512 micro processor nodes, and each processor cell has a RISC type processor of 25 MHz clock cycle and has 16 megabytes of memory. AP-1000 has three sets of networks to achieve both high-speed data transfer and flexibility in processor cell control. One is for synchronization of processor cells (S-net), one for message broadcasting (B-net) which is used for communication between the host computer and processor cells, and one is for inter-processor communication (T-net). Calculations on processor nodes of AP-1000 are launched by a user program running on the host computer.

The sample problem solved on AP-1000 is an eigenvalue problem of a fast critical assembly shown in Fig. 5. The assembly consists of 14×14 drawers and has a core region and an axial and radial blanket. Calculations were performed with a multi-group code GMVP. Two sets of calculations which differ in the batch size were compared. In the second case, the batch size is ten times larger than in the first one. Figure 6 shows the speed-up factor versus the number of processors, where the "total time" is the time elapsed from the start-up on the host computer to the termination of calculations, and the "random walk time" is the time of performing all random walks. As seen from Fig. 6, a speed-up of several hundreds is achieved with the use of 512 processors for the case 1, but the efficiency of the speed-up compared to that of the ideally parallelized case decreases to about 80% in the random walk time and 70% in the total time when 512 processors are used. This is because the small number of histories per processor increases the idling time of waiting for the termination of the slowest processor cell, and because the time overheads for data transfer are no longer negligible. On the other hand, in the second case, in which ten times larger histories are performed in each processor, almost ideal speed-up factor is achieved.

Through parallelization on AP-1000, it was shown that the Monte Carlo code can be parallelized successfully on a MIMD type massively parallel computer AP-1000, but the parallelization by distributing histories over many processors may cause a decrease in parallelization efficiency if the number of histories processed on each processor becomes very small. In the continuous energy Monte Carlo code, whose parallelization is not described in this paper, a difficulty arises because the large amount of cross-section data may exceed the local memory limit of 16 megabytes of AP-1000 in realistic problems.

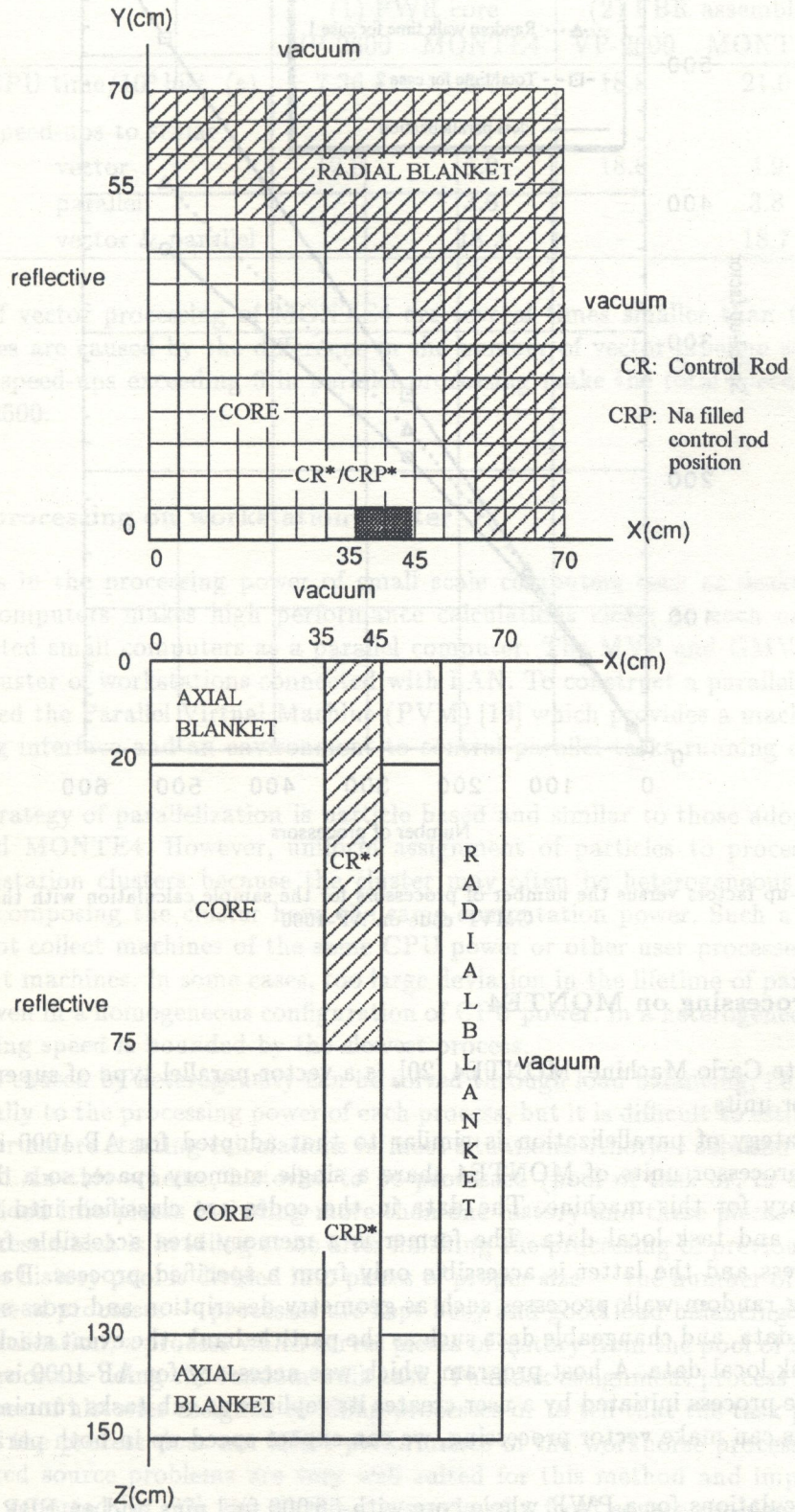

Fig. 5. Core configuration of the sample calculation on AP-1000

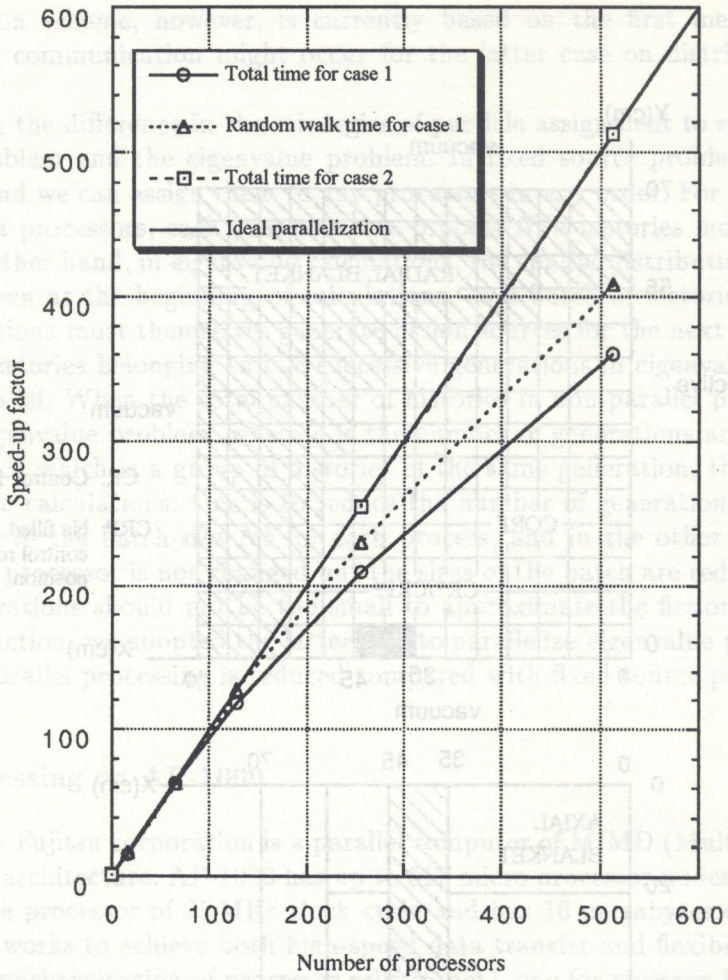

Fig. 6. Speed-up factors versus the number of processors for the sample calculation with the parallelized GMVP code on AP-1000

3.3. Parallel processing on MONTE4

The JAERI Monte Carlo Machine, MONTE4 [20], is a vector-parallel type of supercomputer with 4 vector processor units.

The basic strategy of parallelization is similar to that adopted for AP-1000 in the previous section, but all processor units of MONTE4 share a single memory space, so a different way of coding is necessary for this machine. The data in the codes are classified into two categories: task shared data and task local data. The former is a memory area accessible from more than one parallel process and the latter is accessible only from a specified process. Data that remain unchanged during random walk processes such as geometry description and cross-section data are defined as shared data, and changeable data such as the particle bank, the event stack and tally data are defined as task local data. A host program which was necessary for AP-1000 is not needed for MONTE4 and the process initiated by a user creates its replicas as sub-tasks running concurrently. Since each process can make vector processing, we can expect speed-up in both parallel and vector processing.

Eigenvalue calculations for a PWR whole core with 56 000 fuel pins and an FBR assembly with 91 fuel pins are performed on 4 processor units of MONTE4. Table 4 shows the speed-up factors to scalar calculation by vectorization and parallelization and the total speed-up as product of the two factors. For comparison with a non-parallel type computer, a speed-up factor of the VP-2600 supercomputer with vector processing is also provided.

Table 4. Speed-ups of 4 process parallel processing on MONTE4 compared to single processor scalar processing (with speed-ups of VP-2600)

	(1) PWR core		(2) FBR assembly	
	VP-2600	MONTE4	VP-2600	MONTE4
CPU time/10^4 hist. (s)	7.36	5.7	18.8	21.0
Speed-ups to scalar				
vector	12.0	5.0	18.0	4.9
parallel	–	3.6	–	3.8
vector & parallel	–	18.2	–	18.7

Speed-ups of vector processing of MONTE4 are several times smaller than that of VP-2600. These differences are caused by the difference in the number of vector pipeline sets per processor unit. However, speed-ups exceeding 3 in parallel processing make the total speed-ups comparable to that of VP-2600.

3.4. Parallel processing on workstation cluster

Recent progress in the processing power of small scale computers such as desktop workstations and personal computers makes high performance calculations closer to each engineer by using network-connected small computers as a parallel computer. The MVP and GMVP codes are parallelized on a cluster of workstations connected with LAN. To construct a parallel processing environment, we used the Parallel Virtual Machine (PVM) [19] which provides a machine independent message passing interface and an environment to control parallel tasks running on machines in a network.

The basic strategy of parallelization is particle based and similar to those adopted in the cases of AP-1000 and MONTE4. However, uniform assignment of particles to processes may not be suited for workstation clusters because the cluster may often be heterogeneous, that is, not all the processors composing the cluster have the same computation power. Such a situation occurs when you cannot collect machines of the same CPU power or other user processes are running on some component machines. In some cases, too large deviation in the lifetime of particles may cause heterogeneity even in a homogeneous configuration of CPU power. In a heterogeneous environment, the net processing speed is bounded by the slowest process.

The problem caused by heterogeneity can be solved through load balancing, i.e. assigning particles proportionally to the processing power of each process, but it is difficult to estimate the relative processing power before starting calculations in most situations. Another solution is to use a "pool of task" method. In this scheme, histories to be processed (pool of task or, in our case, pool of history) are divided into pieces including more than one history and these pieces are assigned one by one to a process which is in idling state after finishing the processing of previously assigned histories. When the history pool is divided into pieces of proper size — the number of pieces should be greater than that of processes — processes are kept busy and good load balancing will be achieved. In current parallelization, a process which serves pieces of history from the pool of history is created together with processes doing the random walk task. The task assignment process is only to inform about the number of histories assigned to idling processes or to tell that the task pool is empty, so the existence of the process does not affect performance of the workhorse processes. It can easily be seen that fixed source problems are very well suited for this method and implementations on other codes are reported [21, 22]. We show an example of a fixed source problem calculated with the MVP code on a workstation cluster in Table 5. In this example, 4 workstations with different performances compose the PVM virtual machine. The timing data of a calculation by the history pool method are compared with those by the uniform assignment strategy. The time elapsed for random walk is substantially reduced by the history pool method.

Table 5. Comparison of timings (s) between two ways of a parallel calculation of a fixed source problem with MVP using 4 workstations (total number of histories is 80 000. Size of batch is 2 000 histories)

	History pool	Uniform assignment
Time elapsed for random walk	394.6	722.3
Workstation 1		
history	10 000	20 000
CPU time	181.5	359.3
elapsed time	363.2	719.8
Workstation 2		
history	22 000	20 000
CPU time	196.4	178.4
elapsed time	391.4	356.5
Workstation 3		
history	34 000	20 000
CPU time	373.8	228.3
elapsed time	374.3	229.2
Workstation 4		
history	14 000	20 000
CPU time	375.2	535.3
elapsed time	378.4	538.8

For eigenvalue problems, however, the history pool method may not be so effective because the size of the history pool should be the size of the batch and the size may be too small to be divided into an appropriate size of pieces. Moreover, inter-processor data communication may become necessary to store or exchange particle data in fission sources for the next generation. Load balancing for eigenvalue problems is currently under study.

4. Conclusions

For the neutron and photon transport Monte Carlo simulation, new methods suitable for vector computers and general purpose codes are developed. The vectorized codes achieved substantial speed-up gains compared to the conventional scalar codes. By using their flexibility in geometry modelling and the continuous energy method in cross-section representation, the whole core calculations of reactors were performed within realistic time and calculation costs. Along with vector processing, the applicability of the Monte Carlo codes to a massively parallel computer, a vector parallel machine and a workstation cluster was studied and high performance gain was observed.

References

[1] T. Suzuki, M. Nakagawa, M. Suganuma. Vectorization of MORSE-CG (unpublished) 1985.
[2] K. Asai, K. Higuchi, J. Katakura. *Nucl. Sci. Eng.*, **92**: 298, 1986.
[3] M. Suganuma et al. Vectorization of the continuous energy Monte Carlo code VIM. *JAERI-M*, 86-190, 1987 (in Japanese).
[4] Y. Kurita et al. Vectorization of the MCNP code. *JAERI-M*, 87-022, 1987 (in Japanese).
[5] F.B. Brown. *Trans. Am. Nucl. Soc.*, **43**: 377, 1982.
[6] F.W. Bobrowicz et al. *Parallel Computing*, **1**: 295, 1984.
[7] F.B. Brown. *Trans. Am. Nucl. Soc.*, **53**: 283, 1986.
[8] W.R. Martin, P.F. Nowak, J.A. Rathkopf. *IBM J. Res. Develop.*, **30**: 193, 1986.

[9] M. Nakagawa, T. Mori, M. Sasaki. Development of Monte Carlo code for particle transport calculation on vector processor. *Proc. Supercomputing in Nuclear Applications*, 160, Mito, Mar. 1990.
[10] M. Nakagawa, T. Mori, M. Sasaki. *Nucl. Sci. Eng.*, **107**: 58, 1991.
[11] M. Nakagawa, T. Mori, M. Sasaki. *Prog. Nucl. Energy*, **24**: 183, 1990.
[12] T. Mori, M. Nakagawa, M. Sasaki. *J. Nucl. Sci. Technol.*, **29**: 325, 1992.
[13] T. Mori, M. Nakagawa, M. Sasaki. *J. Nucl. Sci. Technol.*, **29**: 1224, 1992.
[14] M. Nakagawa, T. Mori. *J. Nucl. Sci. Technol.*, **30**: 692, 1993.
[15] J.S. Hendricks et al. *Nucl. Sci. Eng.*, **77**: 71, 1981.
[16] J.F. Briesmeister (ed.) *LA*-7396-M, Rev. 2, 1986.
[17] E.L. Redmond II, J.M. Ryskamp. *Nucl. Technol.*, **95**: 272, 1991, also idem, *Trans. Am. Nucl. Soc.*, **61**: 377, 1990.
[18] H. Ishihata et al. An architecture of highly parallel computer AP-1000. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, May, 1991.
[19] V. Sunderam. PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience*, **2**: No. 4, Dec. 1990.
[20] K. Asai, K. Higuchi et al. The JAERI Monte Carlo Machine. *Proc. Mathematical Methods and Supercomputing in Nuclear Applications*, 341, Karlsruhe, Apr. 1993.
[21] T. Yamazaki et al. A parallelization study of the general purpose Monte Carlo code MCNP4 on a distributed memory highly parallel computer. *Proc. Mathematical Methods and Supercomputing in Nuclear Applications*, 374, Karlsruhe, Apr. 1993.
[22] F. Schmitz, U. Fischer. MCNP4, a parallel Monte Carlo implementation on a workstation network. *Proc. Mathematical Methods and Supercomputing in Nuclear Applications*, 384, Karlsruhe, Apr. 1993.
[23] R.S. Baker. Implementation of a Monte Carlo algorithm for neutron transport on a massively parallel SIMD machine. *Proc. Mathematical Methods and Supercomputing in Nuclear Applications*, 366, Karlsruhe, Apr. 1993.