31

Development of efficient general purpose Monte Carlo codes used in nuclear engineering

M. Nakagawa
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.
Tel: +81-29-282-6120, Fax: +81-29-282-6122
Email: nakagawa@mike.tokai.jaeri.go.jp

Abstract

The Monte Carlo method has high potential to provide very accurate solutions for particle transport problems appeared in nuclear engineering field because it can directly treat complex geometry, accurate physics models and fine structure nuclear data. New Monte Carlo codes have been developed for general purpose use on vector supercomputers and parallel computer environments. The quality of Monte Carlo codes depends significantly on their computation speed. Our codes realize a speedup factor of ~15 compared with conventional scalar codes. Validation of codes was carried out by solving various benchmark problems and analyzing experiments.

Keywords

Particle transport problem, Monte Carlo method, vector supercomputer, parallel computer, speedup, geometry description, quality, validation, benchmark, portability

1 INTRODUCTION

Much large scale numerical software is used in nuclear engineering field. These are used in various phases of basic research, nuclear reactor design, safety and operation, and fuel management of power reactors. The Monte Carlo method is a powerful tool for particle transport calculations such as neutron and photon, which are necessary for determining reactor core characteristics and radiation shielding analyses. This method has high potential to provide very accurate solutions because it can directly treat complex geometry, accurate physics models, and fine structure nuclear data. On the other hand, conventional

R. F. Boisvert (ed.), *Quality of Numerical Software*

deterministic methods must rely on various approximation methods such as finite difference, finite element or nodal methods in solving an integro-differential equation (Boltzman transport equation).

Existing Monte Carlo codes, however, are for conventional scalar main frame machines. If we intend to solve large problems with acceptable accuracy, high computation cost is incurred due to its statistical nature. The figure of merit for an important measure of quality is given by $1/(\sigma t)$, where t the total computation time and σ is the standard deviation. Accordingly, reducing the computation time is essential to realizing a high quality solution. Unfortunately, the conventional Monte Carlo codes are badly structured for vectorization (e.g., many conditional IF statements). Our motivation to develop completely new Monte Carlo codes is to realize high efficiency by adapting new algorithm appropriate for vector supercomputers.

Another way to achieve speedup of computations is to use parallel computers. Our codes are working on both scalar parallel and vector parallel machines. Parallel computation is relatively simple in case of the Monte Carlo method if enough memory is available to store fine structure nuclear data.

In addition to computation speed, the quality of Monte Carlo codes depends on the capability of geometry descriptions in three dimension, variance reduction techniques, portability and so on. Our codes adopt an elegant method to describe very complex nuclear reactor core geometry.

2 BRIEF DESCRIPTION OF MONTE CARLO METHOD TO SOLVE PARTICLE TRANSPORT PROBLEMS

The Monte Carlo method is used to solve equations or simulations of physical phenomena based on a statistical method. Although it is not new, its application is rapidly expanding due to recent great progress of computers. We briefly describe below the method used to analyze accurately neutron and photon transport and nuclear reaction processes which are essentially important in the nuclear engineering field. Collective motion of neutron and photon can be described by the Boltzman transport equation given by

$$\frac{1}{v}\frac{\partial\phi(\mathbf{r},\mathbf{\Omega},E,t)}{\partial t}+\mathbf{\Omega}\cdot\nabla\phi(\mathbf{r},\mathbf{\Omega},E,t)+\Sigma(\mathbf{r},E,t)\phi(\mathbf{r},\mathbf{\Omega},E,t)$$
$$=\int\int\Sigma(\mathbf{r}',\mathbf{\Omega}',E'\rightarrow\mathbf{r},\mathbf{\Omega},E)\phi(\mathbf{r}',\mathbf{\Omega}',E',t)d\mathbf{\Omega}'dE'+Q,$$

where v is the velocity of a particle, Σ the cross section of nuclear reaction, Q the source term by fission or external source. The first term of right hand side is the scattering term which presents the probability that phase space varies from $(\mathbf{r}',\mathbf{\Omega}',E')$ to $(\mathbf{r},\mathbf{\Omega},E)$ by collision. In this equation, interaction between particles is neglected. It is impossible to solve this equation in complex geometry by deterministic methods even for a steady state problem without introducing approximations such as discretization of variables, simplification of geometry and physics models.

Motion of neutrons or photons is random like molecules in gas and nuclear reaction processes is probabilistic phenomena. The Boltzman transport equation presents their

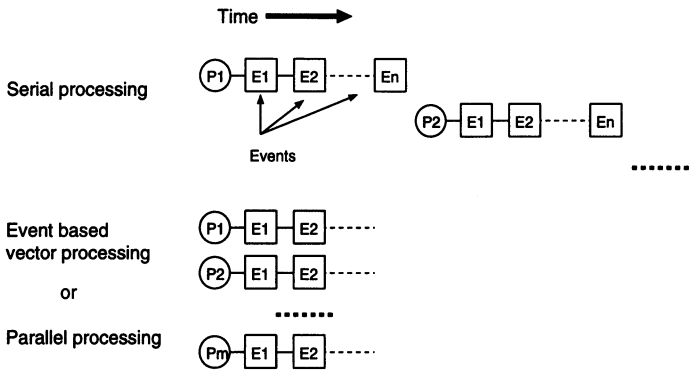

Figure 1 Comparison of scalar and vector processing method of Monte Carlo calculation.

behavior as collective motion, while the Monte Carlo method can accurately simulate the random walk of a particle based on basic physics models. Each particle history is tracked from its birth by fission reaction or an external source till its death by absorption, kill or leakage from a system. By statistically processing the results of many particles, this method can give a solution equivalent to that of the Boltzman equation.

3 SPEEDUP OF MONTE CARLO CALCULATIONS

3.1 Vectorization

In the Japan Atomic Energy Research Institute, GMVP and MVP codes, for multigroup and continuous energy Monte Carlo calculations have been developed on vector supercomputers (Nakagawa et al., 1991, 1990 and Mori et al., 1992). These have been completely rewritten to achieve large speedup by adopting new algorithms appropriate for vector processing. Conventional scalar Monte Carlo codes track a history of one particle at a time. On the other hand, in a 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 represented by vectors with components that correspond to the particles waiting to be processed for that event. This vectorization method is called "event-based" (Brown, 1984), while the conventional method is called "history-based". The difference between event-based and history-based algorithms is shown in Figure 1. All vectorized Monte Carlo codes use an event-based algorithm, but differences in individual approaches to vectorization significantly affect performance. For the MVP and GMVP codes we developed a method called "stack-driven zone selection method".

Particles have 13 descriptors: Cartesian coordinates (x, y, z), flight direction (u, v, w), weight, etc. Random walk processes are resolved into the following basic tasks: source particle generation, collision, free flight, next zone search, lattice (repeated geometry often appears in nuclear reactor cores), reflection, leakage or kill. A "stack" is used to memorize

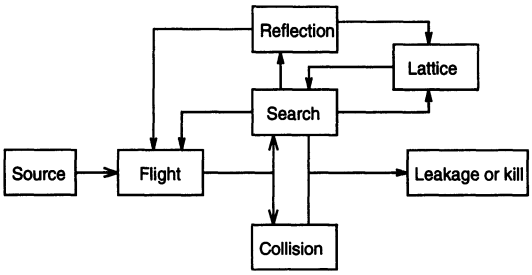

Figure 2 Connections between Monte Carlo calculation tasks in the GMVP and MVP code.

particles queued for a task. Connections between tasks are shown in Figure 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 neighboring zone that a particle enters is found in the search task. The reflection task calculates reflected flight directions on reflective boundary and the lattice task processes particles entering into or escaping from a lattice region, or moving from one lattice cell to another. The collision task determines outgoing directions and energy groups for collided particles.

The order of processing of tasks depends on the number of queued particles in the stacks. The principle is to select a task with the most particles, thus yielding the largest vector length among events. This selection method is called as the "event selection" method. In the case of flight and search tasks, the "zone selection" method is applied, that is, the zones with the most 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 to simplify processing by limiting it to particles in the same zone. For problems whose geometries can be described as repetition of spatial cells, the lattice geometry capability of MVP and GMVP codes substantially contributes to speedup 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 for the flight and search tasks.

Other effective methods for vectorization have been developed for the treatment of reactions using continuous energy cross sections. One example is a table search method which is necessary for several purposes. Two methods, binary and linear searches, are generally used in a Monte Carlo code. Probability data are given at different incident energy grids depending on nuclides and reaction types. The energy grids to be searched for each neutron are stored in different tables. Accordingly, the ability to perform simultaneous searches of various tables with different length is essential to obtain a high computational gain. To implement this capability, we have developed a new scheme for binary searches which uses pointers indicating different tables together with Brown's method for a single table (Brown, 1983). Simultaneous linear searches for many neutrons are vectorized by the

same approach as that for loops containing a feed-backward type IF-test (Miura, 1990) which appears in the rejection sampling method.

3.2 Parallelism

We parallelized our codes on the following four types of parallel processor or parallel processing environment (Sasaki et al., 1994).

1. **Massively parallel processor.** This type of processor has many scalar processor elements. Memories are distributed on each processor and the communications between processors are carried out by passing messages. Recently many machines of this type have been developed by computer vendors. The AP-1000 of Fujitsu corporation described in this paper is an example.
2. **Vector-parallel processor.** This type of parallel processor has relatively small number of vector processor units each of which shares a common memory space or has its own distributed memory. For example, the CRAY XMP series of Cray Research and the SX-3 of NEC belong to the former type, and the VPP/500 of Fujitsu is of the latter.
3. **Workstation cluster.** We used the PVM (Sunderam, 1990) (Parallel Virtual Machine) software developed by the Oak Ridge National Laboratory to construct a virtual parallel processor having distributed memory.

Our parallelization scheme is currently based on particle- or history-based parallelization to reduce interprocessor data communication in distributed memory parallel environments. The AP-1000 by Fujitsu is a parallel computer of MIMD (Multi Instruction stream Multi Data stream) architecture. The AP-1000 has up to 512 microprocessor nodes, and each cell has a RISC processor with a 25 MHz clock cycle and 16 megabytes of memory. In the continuous energy Monte Carlo code, a difficulty arises because the large amount of cross section data may exceed the local memory limit in realistic problems.

The parallelization on a vector parallel machine with distributed memory is quite similar to the above case. The difference is that vector processing is used instead of scalar. Of course, the speedup is achieved by both vector and parallel processing. In the case of a vector parallel machine with a single shared memory, different programming techniques are necessary, especially for memory management. 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. Unchanged data during random walk processes such as geometry descriptions and cross section data are defined as shared data, while changeable data such as particle bank, event stacks and tally data are defined as task local data.

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

allelization strategy of is particle-based and similar to that adopted in the AP-1000 case. 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 computational 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 a deviation in lifetime of particles may cause heterogeneity even in homogeneous configuration of CPU power. In a heterogeneous environment, the net processing speed is bounded by the slowest processor. To avoid such difficulties, we 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 an idle state after finishing processing of previously assigned histories. Good load balance is achievable using this approach. This procedure, however, is applicable only to external source problems.

4 PERFORMANCE EVALUATION

4.1 Vectorization

We have made a variety of evaluations of computation speed, calculation accuracy and reliability. Some examples demonstrate the efficiency achieved by vectorization and parallelism are described in the following.

Radiation shielding problem of deep penetration

As an example of an external source problem, we solved a deep penetration problem through an infinite slab of iron or iron and concrete with 3m thickness. The problem for iron was proposed by Hendricks et al. (Hendricks, 1981) as a benchmark problem. In the latter case, 10cm thick iron and concrete are repeatedly placed. The infinite slabs were modeled in three dimensional geometry by using perfect reflecting surfaces. A monodirectional 14MeV neutron source is located on the one side of the infinite slab. We compared the results by our code (MVP) and those by the MCNP code developed at Los Alamos National Laboratory (Briesmeister, 1986) which has the largest number of users around the world. The computation times of both codes on the FACOM VP-2600 supercomputer are compared in Table 4.1. The "CPU/track" means the total CPU time divided by the total numbers of collisions and boundary crossings. This quantity does not strongly depend on the problem to be solved but depends on the vectorization method used, the computer and vector length. The speedup of MVP compared with MCNP is 15-21 for these problems.

PWR (pressurized water reactor) fuel assembly problem

As a fission source problem, a calculation was performed for a three dimensional PWR fuel assembly with 17×17 pins. The results are compared with a conventional scalar code VIM developed at Argonne National Laboratory (Blomquist, 1980). Table 4.1 shows computation time, speedup, the computed eigenvalues (k_{eff}) and their fractional standard

Table 1 Performance of MVP and MCNP for deep penetration problem.

Material	*performance*	*MVP*	*MCNP*
Iron	CPU/track (μs)	2.19	47.6
	Speedup	21.8	1.0
	Vectorization (%)	98.0	—
Iron+concrete	CPU/track (μs)	3.34	52.4
	Speedup	15.7	1.0
	Vectorization (%)	98.0	—

a) on FACOM VP-2600

Table 2 Performance comparison for PWR fuel assembly problem.

Code	*MVP*	*VIM*
Total CPU time /10^3 histories (s)	0.485	4.72
CPU/track (μs)	2.97	28.9
Speedup	9.7	1.0
k_{eff}	1.218	1.219
FSD (%)	0.11	0.17
Vectorization rate (%)	96.	—

deviations (FSD). The eigenvalues (k_{eff}) calculated by these two codes agree with each other within their standard deviations. The FSD value of MVP is smaller than that of VIM due to an improved evaluation method. A speedup of 9.7 is achieved on the FACOM VP-2600. The above comparisons show higher performance (quality) of the present code achieved by adopting algorithm appropriate for vector processing.

4.2 Parallelism

Massive parallel processing

The first sample problem to evaluate parallel efficiency was solved on AP-1000 for an eigenvalue problem of a fast critical assembly. The calculations were performed by a multigroup code, GMVP. Two sets of calculations which differ in the batch size (number of neurons per batch or generation) were compared. In the second case the batch size is ten times larger than that of the first case. Figure 3 shows the speedup factor versus the number of processors, where the "total time" is the time elapsed from startup on the host computer to the termination of calculations, and the "random walk time" is the time to perform all random walks. As seen from Figure 3, a speedup factor of several hundreds is achieved by using 512 processors for case 1, but the efficiency of speedup 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 idle time waiting for the termination of the slowest

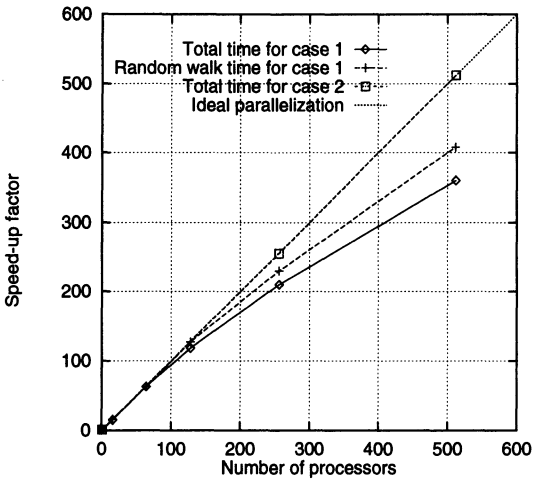

Figure 3 Speedup by massive parallel processing.

processor cell. On the other hand, case 2, in which ten times larger histories per batch are tracked in each processor, achieved almost an ideal speedup factor. Thus the granularity is an important parameter to achieve higher gain.

Using the AP-1000, we have shown that the Monte Carlo code can be successfully parallelized on a MIMD type massively parallel computer, 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. The absolute computation time using about 400 PEs is roughly equal to that by the FACOM VP-2600. Absolute time strongly depends on clock time of processors, so the efficiency is an important measure in case of parallel computations.

Vector parallel processing

In the following, we compare the speedup between the single vector processor machine (VP-2600) and the vector parallel machine (NEC SX-3). Two eigenvalue problems were solved. Case 1 is the whole core of a PWR which has 56000 fuel pins, and case 2 is a fuel assembly with 91 fuel pins. The computation times and speedup factor are compared in Table 3. Speedup factors of 12 or 18 are achieved on the VP-2600 by vectorization. In case of the SX-3, the speedup by vectorization is about a factor of 5 and that by parallelism is about 3.7, so the total speedup is about a factor of 18 for both cases. The efficiency of a vector parallel machine with shared memory such as VPP/500 is almost proportional to the number of processors used and significantly depends on the batch size of each processor (granularity) because the length of vectors is determined by the number of particles simultaneously processed.

The quality of codes with respect to computation speed and accuracy was evaluated as described above. The values of figures of merit for MVP and GMVP are larger by a factor of 10 to 100 compared with conventional scalar codes.

Table 3 Performance of vector parallel computations.

	CPU/10^4particles(s)		*Speedup by vectorization*		*Speedup by parallelism*	*Total speedup*
	VP-2600	*SX-3*	*VP-2600*	*SX-3*	*SX-3*	*SX-3*
Case 1	7.36	5.5	12	5.0	3.6	18.2
Case 2	18.8	21.	18	4.9	3.8	18.7

5 OTHER COMPONENTS WITH RESPECT TO QUALITY OF MONTE CARLO CODES

Input data reduction by adopting multiple lattice geometry

In order to treat general geometry, Monte Carlo codes adopt surfaces or bodies whose geometry can be represented by 2nd or 4th order polynomials. Our codes adopt combinatorial geometry in which every component is described as combination of predefined bodies such as spheres, cylinders, boxes, cones, ellipsoids, triangular and hexagonal prisms, tori and so on. Almost all geometries appearing in nuclear reactor cores can be simply described by such bodies. In a reactor core, the same geometry appears repeatedly. Such repeated structures are called lattices, and the components making up a lattice are called cells. A lattice usually appears repeatedly in a core.

We have developed a new capability to treat flexibly multiple lattices (Nakagawa, 1991). As a result, the required input data and core memory are much reduced. In addition, computation time is reduced by one half because the algorithm adopted is suitable for a vectorized code. The essence of the method is that the same type cells, and those which are symmetric with respect to a plane or a rotation, are treated as if they exist in a single cell through coordinate transformations. As a result, the number of particles which can be simultaneously processed increases; significantly improving vectorization.

Validation by solving benchmark problems

Validation is an important phase of a codes lifecycle. A typical method of validation is to solve benchmark problems and compare the results with the solutions obtained by other methods or experiments. International comparisons on three dimensional transport benchmarks were made in 1990 (Takeda, 1991) with the sponsorship of OECD/NEA. Various physics quantities of four types of reactor cores were calculated, and 41 solutions based on various methods were submitted from various institutes. The GMVP code has given solutions close to the averaged values of all the solutions. In 1992, the OECD/NEA benchmark on the calculation of power distribution within nuclear reactor assembly (Cavarec et al., 1994) was conducted by EDF, France. The benchmark participation involved 19 institutions from 11 countries. The results by the above code were reasonable in comparison with all the solutions. In addition to computational benchmarks, we have analyzed many benchmark experiments which are very useful in evaluating the reliability and accuracy of data and codes. Using these benchmarks, validation has been successfully performed.

Portability

Portability is an important component of software quality, especially for general purpose and production codes. We have tested our codes on various computer environments. Since high efficiency on vector processors is a key feature, the efficiency of these codes was measured on several vector supercomputers developed in Japan and the U.S. The speedup factor is similar among computers belonging to the same generation. New generation computers show a higher gain. Based on this testing, we are convinced that the present codes have good portability, even though vector supercomputers have fairly different architectures. In addition, recent operating systems seem to be focusing on UNIX and many users want to run programs on workstations. To realize user needs, a workstation version has been developed, for which preprocessing modules produce a program adjusted to the user's computer environment. This permits a rapid increase of the number of users.

6 SUMMARY

New vectorized Monte Carlo codes have been developed based on the multigroup and continuous energy methods. A new algorithm has been adopted to increase speedup on vector supercomputers. A speedup of 10~25 has been achieved compared with conventional scalar codes on the FACOM VP-2600. These codes can also run on various computer environments, such as parallel computers and workstations. As for parallel processing, any of massive scalar, vector parallel with shared memory, distributed memory computers or workstation clusters can be selected. Parallel efficiency is almost proportional to the number of processors, except for a problem with small granularity. Verification and validation were made in various phases of the code lifecyle. Comparison of solutions for benchmarks is very useful to evaluate reliability and accuracy. Portability, an important measure of quality for a general purpose code, is good, especially on machines with the UNIX operating system.

The capability of the codes is continuously being enhanced after completion of the prototypes. The codes have been released to domestic users and have been used to solved a variety of problems in various computer environments. Troubles which they encounter are reported to developers through e-mail; developers then check them and fix bugs if necessary. High quality software can not be developed without feedback from many users' experiences.

REFERENCES

Brown, F.B. (1983) Vectorized Monte Carlo methods for reactor lattice analysis. *Proc. Am. Nucl. Soc. Top. Mtg. on Advance in Reactor Computations, Salt Lake City, Utah*, 131.

Brown, F.B. and Martin W.R. (1984) Monte Carlo methods for radiation transport analysis on vector computers. *Prog. Nucl. Energy*, **14**, 269.

Cavarec, C. et al. (1994) Benchmark calculations of power distribution within assemblies. NEA/NSD/DOC(94) 28.

Mori, T, Nakagawa, M. and Sasaki M. (1992) Vectorization of continuous energy Monte Carlo method for Neutron transport calculation. *J. Nucl. Sci. Technol.*, **29**, 4.

Miura, K. (1990) Vectorization and parallelization of transport Monte Carlo simulation. *Proc. 1990 Winter Simulation Conf., New Orleans*, 722.

Nakagawa, M., Mori, T. and Sasaki M. (1991) Comparison of vectorization methods used in a Monte Carlo code. *Nucl. Sci. Eng.*, **107**, 58.

Nakagawa, M., Mori, T. and Sasaki, M. (1992) Monte Carlo calculations on vector supercomputers using GMVP. *Prog. Nucl. Energy*, **24**, 183.

Nakagawa, M, Mori, T. and Sasaki, M. (1991) Hexagonal lattice geometry for Monte Carlo calculations. *Ann. Nucl. Energy*, **18**, 8, 467.

Sasaki, M., Nakagawa, M, Mori, T. (1994) An application study of parallel processing to the particle transport simulation. *Comp. Assisted Mechn. Eng. Sci.*, **1**, 177.

Sunderam, V. (1990) PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience*, **2**(4).

Takeda, T. and Ikeda, H. (1991) 3-D Neutron Transport Benchmarks. *J. Nucl. Sci. Technol.*, **28**, 656.

DISCUSSION

Speaker : M. Nakagawa

T. Tsuda : By commendable effort, you have succeeded in satisfactorily vectorizing classical Monte Carlo codes which were nothing but a huge hierarchy of nested if statements. Then you parallelized the codes to gain further speedup. Based on this experience, which machine would you like to buy, and for what reason: a distributed machine like the Fujitsu VPP500 or a shared memory machine like the new NEC SX-4?

M. Nakagawa : If we assume the same speedup by vectorization when we use a single vector processor, the total speedup by a vector-parallel machine depends on the efficiency of the parallelism. Since our Monte Carlo method for particle transport solves a linear problem, the efficiency by parallelism is generally high for both types of machine, though there are some differences. Parallel programming is simpler on a machine with distributed memory and the parallel efficiency is slightly higher compared with a machine with shared memory.

In the case of the latter machine, some data which occupy a large amount of core memory are shared in a global common area. This causes conflicts of data access by processor elements. Hence, the efficiency gets worse. We actually installed our codes on both types of machines (VPP-500/42 and SX-3) and obtained the results mentioned above. I think that there is another advantage for a distributed type in that the conversion to MIMD scalar parallel machines is very simple.