Computer Assisted Mechanics and Engineering Sciences, 1: 191–204, 1994.
Copyright © 1994 by Polska Akademia Nauk

Development of JAERI Monte Carlo machine and its effective performance

Kenji Higuchi[1], Kiyoshi Asai[2], Masayuki Akimoto[1], Shaw Kambayashi[1], Shinji Tokuda[3], Yukihiro Hasegawa[4], Akira Asami[5] and Makoto Sasaki[6]

[1] *Computing and Information Systems Center, Tokai Research Establishment, Japan Atomic Energy Research Institute (JAERI), Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan*

[2] *Office of International Affairs, JAERI, Fukoku-seimei Bldg. 2-2-2 Uchisaiwai-cho, Chiyoda-ku, Tokyo 100, Japan*

[3] *Plasma Theory Laboratory, Department of Fusion Plasma Research, Naka Fusion Research Establishment, JAERI, Naka-machi, Naka-gun, Ibaraki-ken 311-01, Japan*

[4] *Nuclear Energy Data Center, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan*

[5] *Scientific Engineering Analysis Department, NEC Scientific Information System Development, Ltd. R&D Bldg., 100-1 Sakado, Takatsu-ku, Kawasaki-shi, Kanagawa-ken 213, Japan*

[6] *The Japan Research Institute, Nuclear Engineering Group, Ltd., 16, Ichibancho, Chiyoda-ku, Tokyo 102, Japan*

(Received August 1, 1994)

The JAERI Monte Carlo Machine has been developed mainly to enhance the computational performance of numerical simulations with particle models such as Monte Carlo methods. The features of the JAERI Monte Carlo machine are i) vector processing capability for arithmetic operations, ii) special pipelines for fast vector processing in categorizations of particles, iii) enhanced load/store pipelines for indirectly addressed vector elements, iv) parallel processing capability for spatially and phenomenologically independent particles. This paper describes the design philosophy and architecture of the JAERI Monte Carlo machine and its effective performance through practical applications of the multi-group criticality safety code KENO-IV, the continuous-energy neutron/photon transport code MCNP and other codes for particle simulation.

1. INTRODUCTION

The JAERI Monte Carlo Machine [1], Monte-4, has been developed mainly to enhance the computational performance of numerical simulations by means of particle models such as Monte Carlo methods. In particular, this machine is used for the HASP (Human Acts Simulation Program) being conducted at JAERI. In HASP, human acts to be performed by a human shaped intelligent robot in a nuclear power plant are simulated including the dynamic dose evaluation of a human shaped robot by the Monte Carlo method and the visualization of the simulated results with the use of ray tracing methods.

As for particle models, along with the current increase in computer power, there has been a growing interest in computational science in which numerical simulations with particle models are indispensable to consistently understand physical phenomena or to predict new physical phenomena based on the first principle. In particular, Monte Carlo methods have been used for numerical simulations not only in the fundamental science but also in various engi-

neering areas. In addition to conventional applications such as those in reactor physics and radiation shielding in nuclear engineering, Monte Carlo methods are being widely applied to the new research and development in the nuclear field to make understandable physical processes such as those in the transmutation of actinides by nuclear spallations, in the dilute gas flow of evaporated uranium in the enrichment of natural uranium with laser technology and so on. As mentioned above, particle models including those simulated with Monte Carlo methods seem to be more important for new research aiming at the expansion of nuclear energy utilization.

Monte Carlo simulations using random variables have some principal advantages: many natural physical processes show fundamental stochastic behaviour, which can be simulated numerically, and the evaluation of integrals and the solution of linear integral equations in complex spaces can be efficiently executed. Due to these advantages, Monte Carlo codes such as KENO-IV, MCNP, VIM and MORSE [2, 3, 4, 5] for particle transport problems were developed more than fifteen years ago. The excessive requirement for computational resources, however, was pointed out as a principal drawback in the use of Monte Carlo methods. The Monte Carlo method may need a huge amount of CPU time to give a reasonable statistical result because the accuracy of the simulated result depends on the number of particle histories.

With the increasing computer capability, many efforts were made over the past decade to improve the computational performance of Monte Carlo codes by means of developing algorithms that could be suitable for vector processing. Some of Monte Carlo codes were reported to have been successfully vectorized by changing the history-based algorithms used in the conventional Monte Carlo codes into the event-based ones [6]. The successful vectorizations owe to global algorithmic changes such as comprehensive changes in the data structures and a complete rewriting of the codes to make them have pseudo-lattice structures in order to set up a greater vector length. From such a point of view, a new code was developed [7]. On the other hand, we have shown that the successful vectorization strongly depends on the structures of the original codes and the degree of effort made for the algorithmic changes through our experience with vectorizations of the four codes: KENO-IV, MCNP, VIM and MORSE [8, 9, 10, 11].

Another approach to the improvement of computational performance is to develop a Monte Carlo machine to meet the algorithms used in the existing codes. We have chosen this approach because it is a very time-consuming and expensive task to develop new Monte Carlo codes equivalent to the four codes mentioned above, and it would take a long time to obtain an established reliability of the newly developed codes. The features of the JAERI Monte Carlo machine are i) vector processing capability for arithmetic operations, ii) special pipelines for fast vector processing in categorizations of particles, iii) enhanced load/store pipelines for indirect memory access, iv) parallel processing capability for spatially and phenomenologically independent particles. We have verified that the effective performance of the machine has improved more than ten times for KENO-IV and more than nine times for MCNP codes. This paper describes the design philosophy and architecture of the JAERI Monte Carlo machine and its effective performance through practical applications of the multi-group criticality safety code KENO-IV, the continuous-energy neutron/photon transport code MCNP and other codes for particle simulation.

2. Design philosophy and architecture of Monte-4

2.1. Design philosophy of Monte-4

Monte-4 has been designed on the basis of the following concepts: i) since the existing Monte Carlo codes have been widely used on large-scale scalar computers or vector processors, the machine should have a similar architecture to that of the existing computers such as a high-speed clock cycle, vector and parallel processing capability and a large memory size, ii) since there remains a scalar computation part in Monte Carlo calculation, the machine should have a high-speed clock

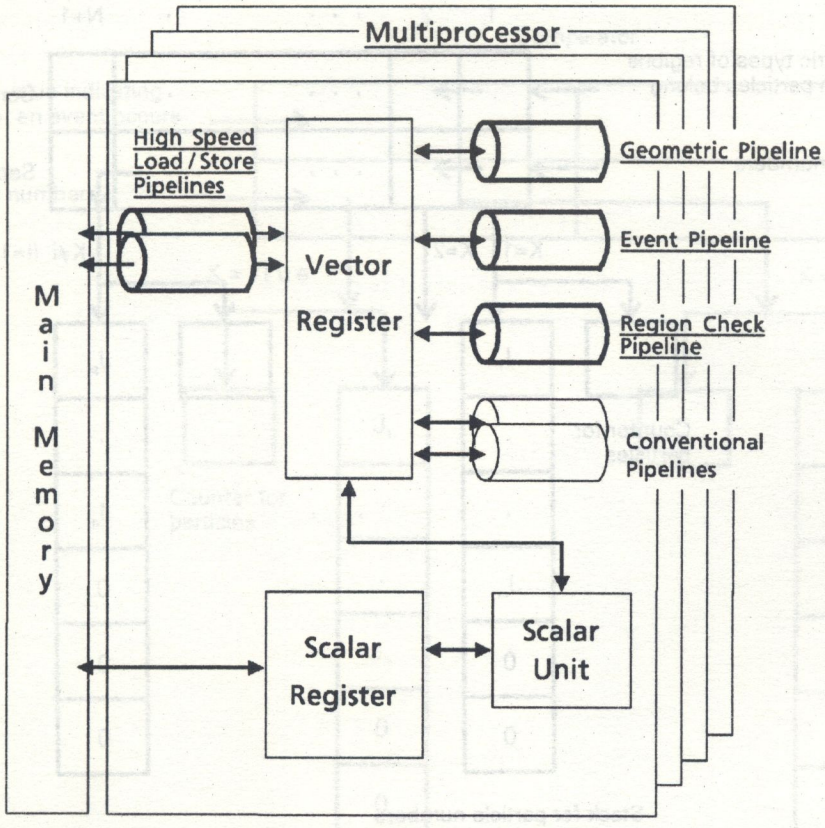

Fig. 1. Architecture of the JAERI Monte Carlo machine

cycle, and iii) the machine should be well balanced between the data transfer rate and vector processing capability. We have chosen SX-3/11 as a basis of our Monte Carlo machine because it had the highest clock cycle in the world when we started the design.

2.2. The architecture of Monte-4

The architecture of Monte-4 with the clock cycle of 2.5 ns is shown in Fig. 1. Each of its processors has one pipeline set for arithmetic operations, three special pipelines called Monte Carlo pipelines, plus an enhanced load/store pipeline. In addition to these pipelines, Monte-4 has parallel processing capability with four processors, which multiplies the speed-up ratio is multiplied. The features and performance of the newly developed and improved hardware are as follows:

2.2.1. Monte Carlo pipelines

Geometric pipeline

In Monte Carlo simulation, the distance of a particle to the collision is given by a macroscopic total cross-section of the medium and uniformly distributed random numbers. When particles are tracked along their paths and the distances from their current positions to the boundaries of the regions are calculated, it is necessary to categorize particles according to the geometries of the regions they belong to. This procedure is carried out by means of multi-way conditional branches. It is impossible to vectorize DO loops with such multi-way conditional branches on conventional vector processors. Monte-4 has been equipped with special pipelines called geometric pipelines to quickly process such DO loops. The concept [12] of the pipeline is shown in Fig. 2.

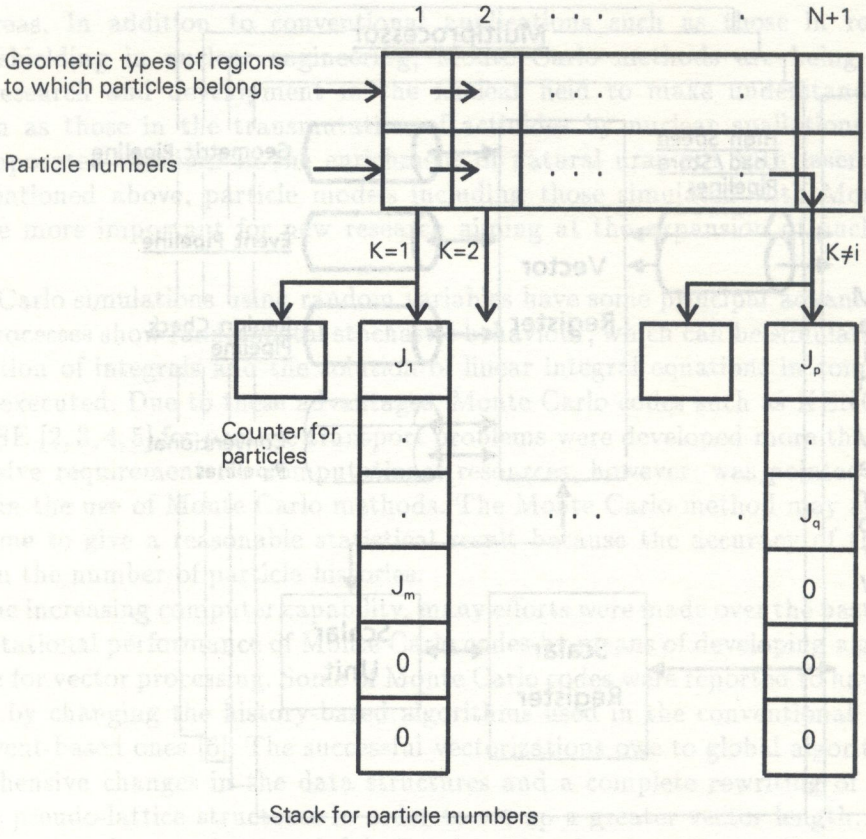

Fig. 2. Concept of the geometric pipeline

Event pipeline

In vectorized Monte Carlo simulation, particles in the same event are gathered and processed together. This pipeline quickly processes categorizations of the particles by events, such as collisions, boundary crossings and so on. The concept of the pipeline is shown in Fig. 3.

Region check pipeline

When particles are tracked along their paths and the intersections of the paths with the regions are calculated, it is necessary to judge whether locations (to be intersections) are in the correct regions or not. This procedure consists in evaluating logical expressions and it cannot be vectorized because of nests of conditional branches. Monte-4 has been equipped with a special pipeline called the region check pipeline to perform this procedure quickly. An example of the procedure is shown in Fig. 4.

2.2.2. Enhanced load/store pipeline

We have enhanced load and store pipelines in order to improve the low data transfer rate between vector unit and memory unit, especially for indirect accesses required for simultaneous tracking of particles with different behaviours.

2.2.3. Parallel processing capability

The parallel processing capability must be major in the near future, considering the performance limit of the present vector processor. The implementation of parallel processing capability in our machine can provide a preliminary parallelism environment. Four processors have been implemented.

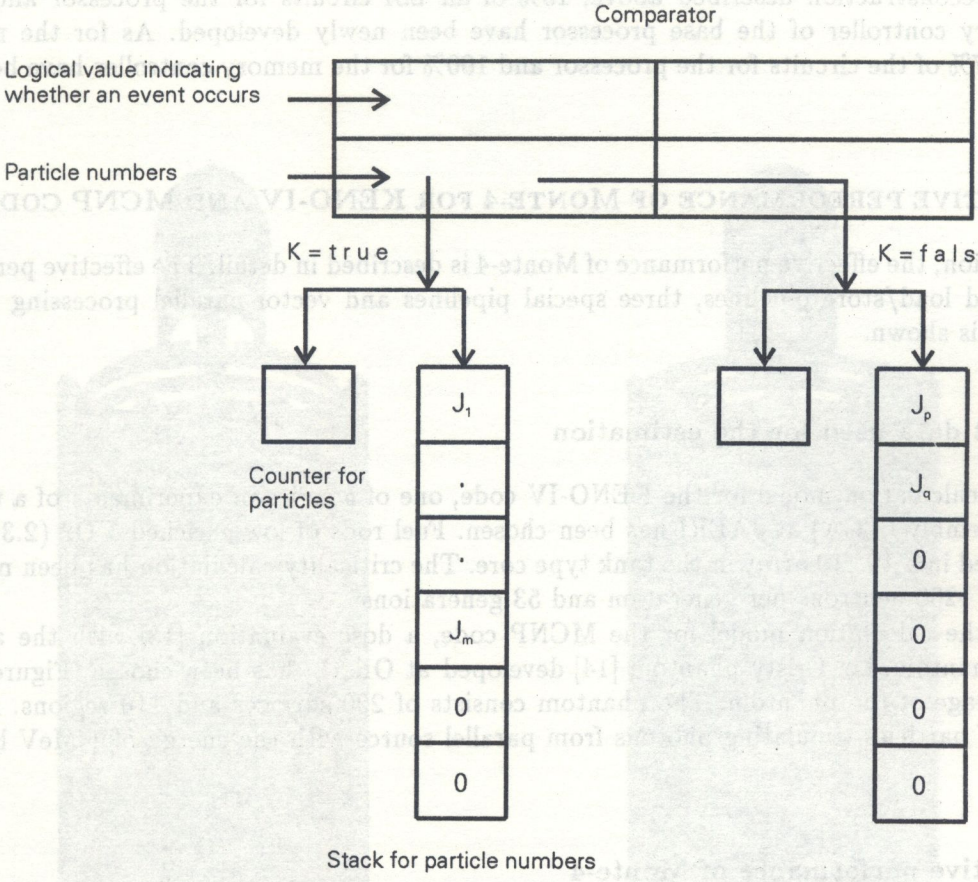

Fig. 3. Concept of the event pipeline

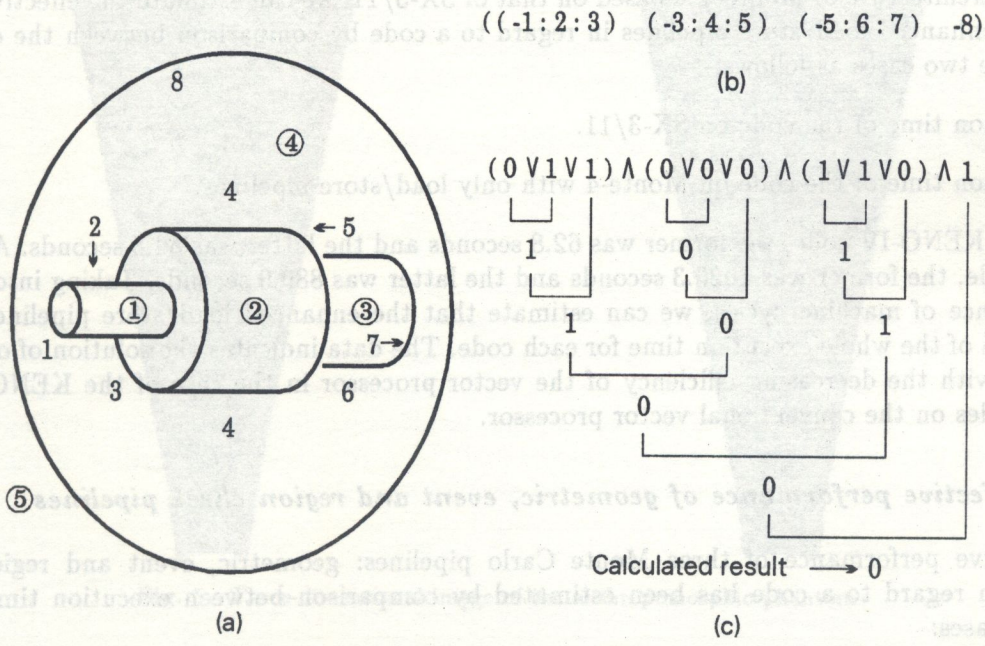

Fig. 4. Concept of the region check procedure. (a) Example of region definition by surfaces 1–8. (b) Logical expression of the cell ④ (the residua of the sphere after three cylinders are removed). (c) Evaluation of the binary expression of the cell ④ with respect to a point (particle location). Such an evaluation is performed by means of a nest of conditional branches, which cannot be vectorized on conventional vector processors

In the reconstruction described above, 15% of all LSI circuits for the processor and 32% for the memory controller of the base processor have been newly developed. As for the multi-chip package, 67% of the circuits for the processor and 100% for the memory controller have been newly developed.

3. Effective performance of Monte-4 for KENO-IV and MCNP codes

In this section, the effective performance of Monte-4 is described in detail. The effective performance of enhanced load/store pipelines, three special pipelines and vector-parallel processing with four processors is shown.

3.1. Input data used for the estimation

As for the calculation model for the KENO-IV code, one of a series of experiments of a tank-type critical assembly (TCA) at JAERI has been chosen. Fuel rods of low-enriched UO2 (2.3% U-235) are arranged in a 19×19 array in the tank type core. The criticality calculation has been made with a model of 1200 neutrons per generation and 53 generations.

As for the calculation model for the MCNP code, a dose evaluation [13] with the anthropomorphic phantom, i.e. Cristy phantom [14] developed at ORNL, has been chosen. Figure 5 shows the 3-D image of the phantom. The phantom consists of 200 surfaces and 110 regions. A million histories of particles simulating photons from parallel source with the energy of 1 MeV have been tracked.

3.2. Effective performance of Monte-4

3.2.1. Effective performance of enhanced load/store pipelines

Since the architecture of Monte-4 is based on that of SX-3/11, we can estimate the effective performance of enhanced load/store pipelines in regard to a code by comparison between the execution time in the two cases as follows:

1. Execution time of the code on SX-3/11.
2. Execution time of the code on Monte-4 with only load/store pipelines.

As for the KENO-IV code, the former was 62.8 seconds and the latter was 54.8 seconds. As for the MCNP code, the former was 1020.3 seconds and the latter was 889.9 seconds. Taking into account the difference of machine cycles, we can estimate that the enhanced load/store pipeline reduces nearly 13% of the whole execution time for each code. The data indicates the solution of one of the problems with the decreasing efficiency of the vector processor in the case of the KENO-IV and MCNP codes on the conventional vector processor.

3.2.2. Effective performance of geometric, event and region check pipelines

The effective performance of three Monte Carlo pipelines: geometric, event and region check pipelines in regard to a code has been estimated by comparison between execution time in two following cases:

1. Execution time of the code on Monte-4 with only load/store pipeline.
2. Execution time of the code on Monte-4 with load/store, geometric, event and region check pipelines.

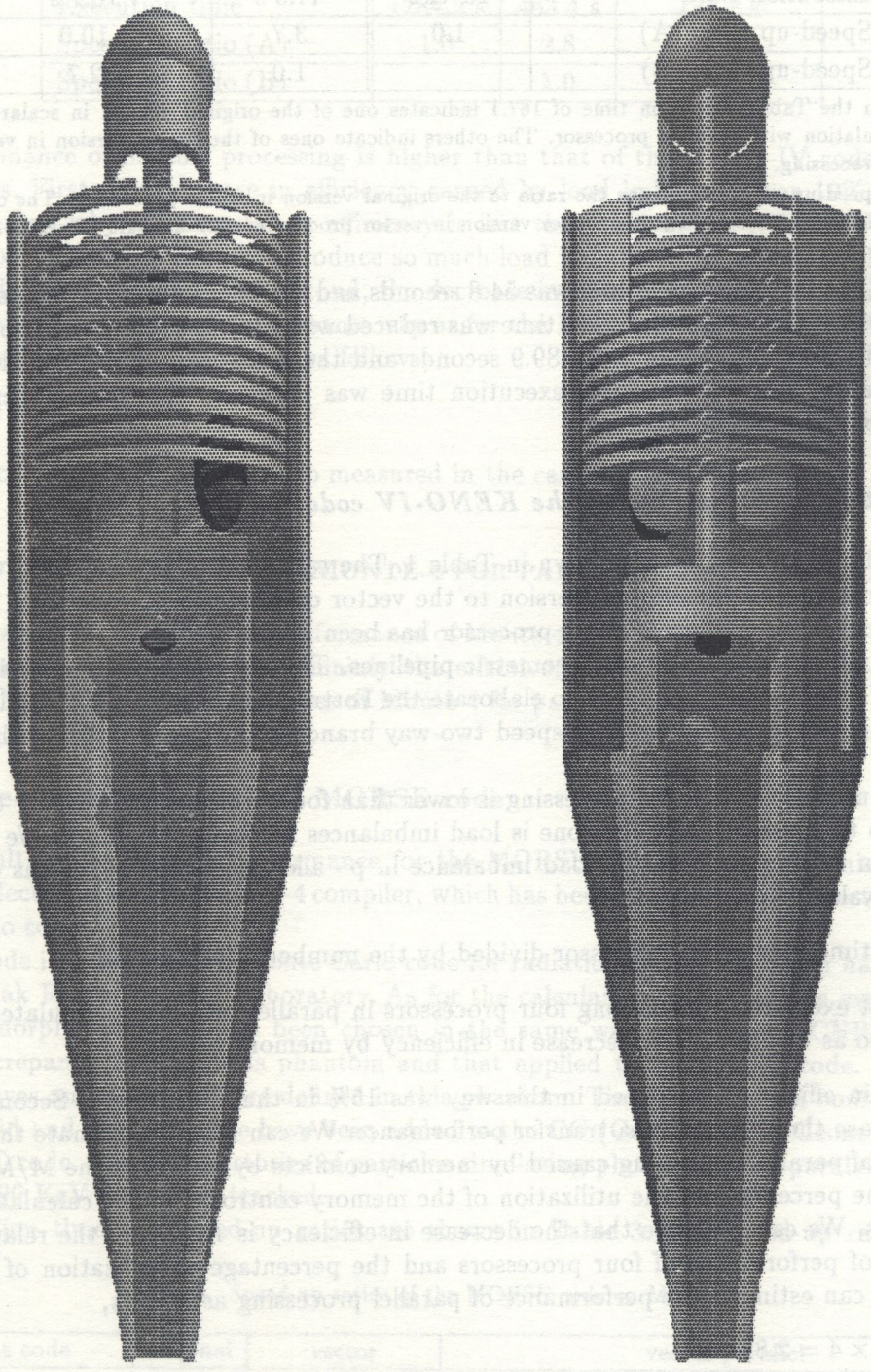

Fig. 5. Three-dimensional image of the anthropomorphic phantom

Table 1. Speed-up ratios of the KENO-IV code on Monte-4

	1 processor		4 processors
Processing mode	scalar	vector	vector-parallel
Execution time	167.1 s	44.9 s	16.7 s
Speed-up ratio (A)	1.0	3.7	10.0
Speed-up ratio (B)		1.0	2.7

i) In the Table, execution time of 167.1 indicates one of the original version in scalar calculation with a single processor. The others indicate ones of the vector version in vector processing.

ii) Speed-up ratio (A) means the ratio to the original version in scalar processing. The other one means the ratio to the vector version in vector processing with a single processor.

As for the KENO-IV code, the former was 54.8 seconds and the latter was 44.9 seconds. This result shows that 20% of the whole execution time was reduced with the three Monte Carlo pipelines. As for the MCNP code, the former was 889.9 seconds and the latter was 463.4 seconds. This result shows that nearly 50% of the whole execution time was reduced. This mainly depends on the effectiveness of region check pipelines.

3.2.3. Effective performance for the KENO-IV code

The measured speed-up ratios are shown in Table 1. The ratios are defined as CPU time ratios of the scalar calculation of the original version to the vector calculations of vectorized versions. The reduction of execution time on a single processor has been mainly achieved through the hardware capability of indirect addressing and geometric pipelines. The region check pipeline is not used in our KENO-IV version. It is necessary to elaborate the Fortran compiler to get a good performance of the event pipeline designed for high-speed two-way branches and the revision of the compiler is planned.

The speed-up ratio of parallel processing is lower than four, which is the number of processors. This is due to two reasons. The first one is load imbalances between processors. We can estimate the decrease in efficiency caused by load imbalance in parallel processing by means of comparing the following values:

1. Execution time on a single processor divided by the number of processors.

2. The longest execution time among four processors in parallel processing simulated on a single processor so as to remove the decrease in efficiency by memory conflicts.

The decrease in efficiency estimated in this way was 15% in that calculation. Secondly, memory conflicts decrease the effect of data transfer performance. We can roughly estimate the decrease in the efficiency of parallel processing caused by memory conflicts by means of the M/M/1 queueing model [15]. The percentage of the utilization of the memory controller in the calculation was 20% in our analysis. We can estimate that the decrease in efficiency is 15% from the relation between the efficiency of performance of four processors and the percentage of utilization of the memory controller. We can estimate the performance of parallel processing as follows,

$$0.85 \times 0.85 \times 4 = 2.89.$$

This value is close to the speed-up ratio measured in the case of parallel processing.

3.2.4. Effective performance for the MCNP code

The speed-up ratios are shown in Table 2. The reduction of execution time on a single processor has been mainly achieved due to the region check pipeline.

Table 2. Speed-up ratios of the MCNP code on Monte-4

	1 processor		4 processors
Processing mode	scalar	vector	vector-parallel
Execution time	1284.2 s	463.4 s	137.8 s
Speed-up ratio (A)	1.0	2.8	9.3
Speed-up ratio (B)		1.0	3.4

The performance of parallel processing is higher than that of the KENO-IV code. This is due to two reasons. First, the decrease in efficiency caused by load imbalance was 10% in the calculation. The reason why the decrease in efficiency is less than that of the KENO-IV code is that synchronization for tallying does not produce so much load imbalance, as it does in the case of the KENO-IV code as described above. Second, the decrease in efficiency caused by memory conflicts is 7%, which has been estimated in the same way as for the KENO-IV code. Thus, we can estimate the performance of parallel processing as follows,

$$0.90 \times 0.93 \times 4 = 3.35.$$

This value is close to the speed-up ratio measured in the case of parallel processing.

4. Effective performance of Monte-4 for particle simulation codes

In the previous section, the effective performance of Monte-4 with the KENO-IV and MCNP codes has been described. The results show mainly the effectiveness of Monte Carlo pipelines. In this section, the effectiveness of other features of Monte-4 for particle simulation codes is shown.

4.1. Effective performance for the MORSE code

First, the result of the effective performance for the MORSE code is shown. In this example we indicate the effectiveness of the Monte-4 compiler, which has been improved for the vector processing of Monte Carlo codes.

MORSE code is a multi-group Monte Carlo code for radiation transport, which has been developed at the Oak Ridge National Laboratory. As for the calculation model, a dose evaluation with the anthropomorphic phantom has been chosen in the same way as for the MCNP code. There is a small discrepancy between this phantom and that applied to the MCNP code. It lies in the fact that the eyes and esophagus are defined in this phantom. Three geometrical bodies, the torus, general ellipsoid and elliptical cone have been added to the CG (Combinatorial Geometry) package of the MORSE code. A million histories of particles simulating photons from a parallel source with the energy of 90 KeV have been tracked.

The execution time and speed-up ratios are shown in Table 3. The speed-up ratio in vector

Table 3. Speed-up ratios of the MORSE code on Monte-4

Version of the code	original	vector		vector-parallel		
Number of processors	1	1	1	2	3	4
Processing mode	scalar	scalar	vector	vector-parallel	vector-parallel	vector-parallel
Execution time (s)	380.8	568.8	147.7	80.1	60.2	49.8
Speed-up ratio (A)	1.0	0.67	2.6	4.8	6.3	7.6
Speed-up ratio (B)			1.0	1.8	2.5	3.0

Speed-up ratio (A) means the ratio to the original version in scalar processing. The other one means the ratio to the vector version in vector processing with a single processor.

processing was 2.6. The speed-up ratio we obtained in vector processing of the MORSE code on a conventional vector processor, was 1.5 as described in reference [10]. The progress was due to Monte Carlo pipelines, enhanced load/store pipelines and improvements of the compiler. The most pronounced improvement of the compiler was switching of vector and scalar processing. Figure 6 shows the relation between the performance of vector processing and the vector length with respect to a DO loop in the MORSE code. The performance in scalar processing is expressed as 1.0 in the figure. In vector processing with a small vector length, we cannot obtain a higher performance than that in scalar processing. Table 4 shows the processing time of one particle in the DO loop in the case of vector processing with the vector length less than or equal to four and with the vector length more than four. The result shows that the performance in the former case compared with the latter is nearly 1 to 10. However, we cannot avoid vector processing with small vector lengths. In the vector processing of the Monte Carlo codes for particle transport, although a large number of particles are simultaneously tracked at the beginning of the calculation, the number of particles decreases whenever one of them is killed through absorption or escape. In the extreme case, we are forced to perform vector processing with one particle at the end of the calculation. Such a small and diminishing vector length is one of the features of vector processing in the Monte Carlo codes as described before. The compiler on Monte-4 was improved so as to give it the capability to dynamically switch vector processing to scalar processing if the vector length is less than or equal to 4. Nearly 40% of the execution time of the DO loop was reduced due to this capability. Nearly 18% of the execution time of the code was reduced.

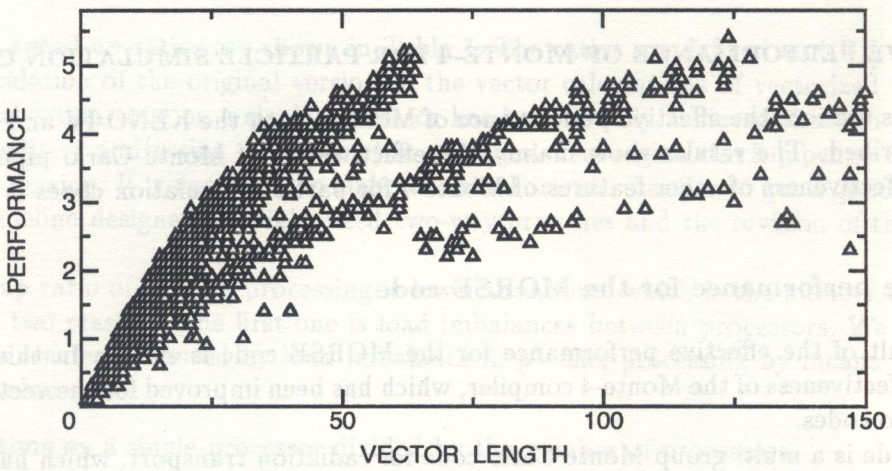

Fig. 6. Vector performance in a sample DO loop

Table 4. Comparison of processing time of one particle by vector lengths

	Case 1	Case 2
Number of particles (N)	14 347	116 803
Execution time (T)	66 ms	48 ms
Processing time of one particle (T/N)	4.6 μs	0.41 μs

In this DO loop, the procedure to check whether a particle crosses the boundary of a region or not is carried out by comparison of the length of a particle path with the distance to the region boundary. Case 1 denotes vector processing with the vector length (number of particle) less than or equal to 4. Case 2 denotes vector processing with the vector length more than 4. N means the total number of particles processed in the DO loop in each case. T means the total execution time in each case.

We obtained the speed-up ratio of 4 compared with the scalar processing of the vector version. However, the speed-up ratio compared with the scalar processing of the original version was 2.6 only, because the execution time of the vectorized code in scalar processing increased after the

modification for vector processing. This overhead due to the initializations of DO loops and address calculations is commonly observed in the vectorization of Monte Carlo codes. As for the MORSE code, such an overhead was large compared with other Monte Carlo codes such as KENO-IV and MCNP.

We obtained the speed-up ratio of 3 in parallel processing with four processors. The parallelization ratio of the code was 0.99. The parallelization ratio is defined as the ratio of the execution time in scalar processing for the parallelized part to the whole execution time of the code. The theoretical speed-up ratio with that parallelization ratio is nearly four. However, the actual speed-up ratio is lower than this value because of memory conflicts and the overhead for parallel processing.

4.2. Effective performance for the EM3PJ code

In this section, the result of the vector parallel processing of the EM3PJ code is described. The result in this example shows that we have obtained a high speed-up ratio by the coding that is suitable to the hardware features of Monte-4.

This code was parallelized on Monte-4 with the use of the micro-tasking facility that makes possible the parallel execution of DO loops as well as independent sections of a code. The parallel optimization can be applied to the DO loops which contain recursive operations or procedure calls that cannot be easily vectorized.

EM3PJ is a three-dimensional code used to analyse the dynamics of plasmas and beams self-consistently. In this code, nonlinear interaction between the particles and electromagnetic field is simulated by a Particle-In-Cell model. One of the main loops in the code is a four-fold loop computing charge and current density. Another loop computes the electromagnetic forces on the particles. More than 90% of the computational cost is localized on these loops for the input data used. The code was modified so that parallel processing with the micro-tasking facility could be applied to the four-fold loops, whose iteration frequency is determined by the number of cells in the 3-D space and by the number of particles, that is $16\times16\times16$ and 4096, respectively. In addition to the insertion of directions for parallel processing, we optimized the code by means of such techniques as unrolling of the outer loop into the inner loop, unification of I/O operations, modification to avoid bank conflicts, modification to reduce cache misses, insertion of a direction indicating a vectorized loop and exchange of loops. Unrolling of the outer loop was carried out for the reduction of data transfer between the main memory and processor units. We can decrease the frequency of data transfer of an array into a quarter by unrolling the outer loop in the inner loop four times because the frequency of loading the array becomes one though the array is used four times in inner loop. The unification of the I/O operation (optimization of implied-DO lists in the I/O statement), which reduces the number of I/O operations resulting in the reduction of the scalar processing time, was performed with the use of an optimization tool that automatically generates an optimized source on Monte-4. Modifications to avoid bank conflicts and reduce cache misses are mentioned later. Directions indicating a vectorized loop were inserted to carry out vector processing with a greater vector length. The exchange of loops was carried out to remove redundant operations. Among them, a modification of the array size to avoid bank conflicts was efficient for vector-parallel processing, as described below.

In order to obtain a higher memory throughput, the main memory of the supercomputer consists of small regions, called banks, which can be accessed simultaneously. A conflict of data access to the same bank is called a 'bank conflict'. In the vector processing of this code on Monte-4 with four 512-byte banks, bank conflicts occurred frequently because some arrays were referred by a distance of power of 2 such as 1, 257, 513, 769, ... Note that if A is an array of 8-byte elements, $A(1)$, $A(257)$ and $A(513)$ are located in the same bank. We have changed the data structure of the code so as to avoid bank conflicts.

The execution time and speed-up ratios are shown in Table 5. The execution time in scalar processing was improved to 400 seconds from the original execution time of 1 400 seconds. Such a great time reduction is due to the unification of I/O operations and modifications to reduce cache

misses. In a computer, the cache memory is used for fast access to the data in scalar processing. A cache miss occurs when there is no data in the cache. We have changed the data structure of the code so as to reduce cache misses, as was the case with bank conflicts. However, we cannot remove cache misses completely because the capacity of the cache is limited and the data in the cache is replaced by hardware control according to the data access pattern.

Table 5. Speed-up ratios of the EM3PJ code on Monte-4

Version of the code	original	vector		vector-parallel
Number of processors	1	1	1	4
Processing mode	scalar	scalar	vector	vector-parallel
Execution time (s)	1415.9	433.7	10.2	5.7
Speed-up ratio	1.0	3.3	138.8	248.4

The vector version was optimized for both vector and scalar processing

We seem to have obtained the speed-up ratio of nearly 40 in vector processing compared with the scalar processing of the modified version. However, this speed-up ratio is overestimated because of the effect of cache misses described above. Reasonable speed-up ratio seems to be nearly 20 since we estimated the execution time without cache misses of 200 seconds. The excellent speed-up ratio is due to the high vectorization ratio and the great vector length. On the other hand, the speed-up ratio we obtained in parallel processing was just 1.8 in spite of the use of four processors. The reason for such a low speed-up ratio is as follows. The parallelization ratio is defined by B/A, where A is the execution time on a single processor and B is the execution time for the parallelized part on a single processor. The actual parallelization ratio of the code was 0.85. The theoretical speed-up ratio by parallel processing with four processors is

$$1.0/((1.0-0.85)+0.85/4)=2.76.$$

The actual speed-up ratio is

$$10.2/5.7=1.8.$$

The reason why the latter is low compared with the former is as follows:

1. The parallelized code has additional operations such as task generation, invocation of micro-tasks, synchronization, generation of mutually exclusive areas and loop iteration distribution control.
2. Execution of this code causes a heavy traffic between memory and vector processing units, resulting in memory conflicts that affect the performance adversely.
3. Cache misses increase because of the operations to maintain the cache coherence in parallel processing.

As for the EM3PJ, a significant speed-up ratio was obtained compared with scalar processing due to the coding suitable for the hardware features of Monte-4, including the optimization for scalar processing.

4.3. Effective performance for the molecular dynamics code ISIM

In this section, the result of the vector parallel processing of the ISIM code is described. The result in this example shows that we have obtained a high speed-up ratio through enhanced load/store pipelines of Monte-4.

The ISIM code performs molecular dynamics simulations for liquid metals [16] maintaining the temperature of the subjective liquid of simulations with an isokinetic constraint introduced

by Hoover et al. [17]. ISIM is currently applied to self-consistent calculations of the interatomic potential of liquid metals based on the theory of the pair distribution function.

The classical molecular dynamics simulation determines the phase space trajectories of atoms by means of integrating the Newtonian equation of motion for each atom:

$$m\ddot{\mathbf{r}}_i = \mathbf{F}_i \,. \tag{1}$$

Here m is the mass of the atom, $\mathbf{r}_i(t)$ is the position of the atom i, and $\mathbf{F}_i$ is the force acting on the atom i. In the ISIM code, Equation (1) is integrated with a 5-th order predictor-corrector algorithm by Bernu [18], and the interatomic force is evaluated with a pairwise additive spherical potential $u(r)$ derived from an appropriate theory:

$$\mathbf{F}_i = -\sum_{j \neq i} \frac{\partial u(r_{ij})}{\partial \mathbf{r}_i} \,, \qquad r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| \,. \tag{2}$$

It is well-known that the calculation of $\mathbf{F}_i$ is the most time consuming part in the molecular dynamics simulations: for example, in the ISIM code nearly 98% of the CPU time is spent on the calculation of Eq. (2).

For liquid metals, the pair potential $u(r)$ exhibits a long-range oscillatory tail which is not observed in the widely used Lennard-Jones potential. In order to treat the oscillatory tail of potential $u(r)$ properly, it is necessary to use a large cut-off distance for $u(r)$, typically ranged from 4 to 6 times of the mean interatomic distance, which leads to several hundreds of atoms interacting with the one considered. Owing to the fact that the number of atoms interacting with each atom is large enough for vector processing, the calculation of Eq. (2) is vectorized with respect to the interacting atoms for a given atom with the use of indirectly addressed arrays.

Concerning the present application of the ISIM code, it is important to obtain a very accurate sample of the pair distribution function $g(r)$ defined by

$$g(r) = \frac{1}{N} \left\langle \sum_{i \neq j} \delta(\mathbf{r} + \mathbf{r}_i - \mathbf{r}_j) \right\rangle \tag{3}$$

where N is the number of atoms. In order to obtain highly accurate $g(r)$, it is necessary to perform a simulation for long time-steps with a large number of particles. In the ISIM code the calculation of $g(r)$ is enhanced by the vectorization of this part with the perch algorithm [19].

In the calculation used for the performance evaluation, a molecular dynamics simulation of over 1 000 time-steps for liquid sodium at the temperature 373 K with 16 000 atoms has been performed. The time increment for integrating Eq. (1) is 2.35 femtoseconds. The cut-off radius is chosen to be 15.12 Å which is about 4.5 times the mean interatomic distance.

We summarize the performance for the code on a single processor of Monte-4 as follows. The scalar and vector processing times are 10 hours 47 minutes and 48 minutes, respectively. Consequently, we have obtained the speed-up ratio of 13.4 with vector processing. The average vector length of the calculation of Eq. (2) is around 190. Although the data is mainly accessed by indirect addressing, we have obtained excellent results. The results show that the enhanced load/store pipeline of Monte-4 is a powerful hardware feature in particle simulations.

5. Concluding remarks

We have obtained significant gains in the vector-parallel processing of the KENO-IV, MCNP and MORSE codes on Monte-4 compared with the low performance ratios of 1.5, according to our experience, on the conventional vector processor in the cases of the KENO-IV, MCNP, MORSE and VIM codes. We can also expect better performance in parallel processing due to the elaboration of the load sharing algorithm to improve the load balance and the coding in order to avoid memory conflicts as much as possible.

We have modified other production level codes for particle simulation and obtained successful results. Especially in the vector-parallel processing of the EM3PJ code, a significant speed-up ratio was obtained by means of memory access optimization. These results show that it is possible to attain high performance in vector-parallel processing in regard not only to Monte Carlo codes for particle transport problem but also to other codes for particle simulation through the coding that is suitable to the hardware features of Monte-4.

ACKNOWLEDGEMENTS

The authors express their sincere thanks to H. Matsumoto at Computers Division, NEC Corporation and Y. Seo at C&C Research Laboratories, NEC Corporation for their valuable discussion and comments.

REFERENCES

[1] K. Asai et al. Monte Carlo calculations on high speed machines. *Prog. Nucl. Energy*, **24**: 175, 1990.
[2] L.M. Petrie, N.F. Cross. ORNL KENO IV: An improved Monte Carlo criticality program. *ORNL* 4938, 1975.
[3] LASL Group TD-6. MCNP: A general Monte Carlo code for neutron and photon transport, *LA 7396-M, LASL*, July, 1978.
[4] L.J. Milton. VIM User's Guide. *Applied Physics Division, ANL*, June 1981.
[5] M.B. Emmett. The MORSE Monte Carlo radiation transport code system. *ORNL* 4972, 1975.
[6] F.B. Brown, W.R. Martin. Monte Carlo methods for radiation transport analysis on vector computers. *Prog. Nucl. Energy*, **14**: 269, 1984.
[7] M. Nakagawa et al. Development of Monte Carlo code for particle transport calculations on vector processors. *Proc. Supercomputing in Nuclear Applications*, p. 160, Mito, March 1990.
[8] K. Asai et al. Vectorization of KENO IV code and an estimate of vector-parallel processing. *JAERI-M*, **86**-151, 1986
[9] M. Suganuma et al. Vectorization of continuous energy Monte Carlo code VIM. *JAERI-M*, **86**-190, 1987 (in Japanese).
[10] Y. Kurita et al. Vectorization of Monte Carlo code MCNP. *JAERI-M*, **87**-022, 1987 (in Japanese).
[11] K. Higuchi et al. Vectorization of MORSE-DD code. *JAERI-M*, **87**-032, 1987 (in Japanese).
[12] K. Asai et al. Vectorization of the KENO IV code. *Nucl. Sci. Eng.*, **92**: 298, 1986.
[13] K. Higuchi, Y. Yamaguchi. Investigation of the applicability of MCNP code to complicated geometries. *JAERI-M*, **94**-057, 1994 (in Japanese).
[14] M. Cristy. Specific absorbed fractions of energy at various ages from internal photon sources. *ORNL*, TM-8381, 1981.
[15] L. Kleinrock. *Queueing Systems*, Vol. I. John Wiley & Sons, Inc., New York–London, 1975.
[16] ISIM code is a modified version of ISIS code for soft-sphere fluids; S. Kambayashi. A molecular dynamics simulation code ISIS. *JAERI-M*, **92**-080, 1992 (in Japanese).
[17] W.G. Hoover, A.J.C. Ladd, B. Moran. *Phys. Rev. Lett.*, **118A**: 111, 1983.
[18] B. Bernu. *Physica*, **122A**: 129, 1983.
[19] R.H. Mendez, S.A. Orszag. *Lecture Notes in Engineering*, Vol. **36**: 59–72 Springer-Verlag, New York, 1988.