Proceedings of the Korean Nuclear Society Spring Meeting
Pohang, Korea, May 1999

A Benchmark of Beowulf Cluster for Nuclear Engineering Advanced Computation

Jeong Hwan Park, Nam Zin Cho, and Myun Soo Kim

Korea Advanced Institute of Science and Technology
Department of Nuclear Engineering
373-1 Kusong-dong, Yusong-gu
Taejon, Korea 305-701

Abstract

Network-of-Workstations technology is applied to the challenge of implementing very high performance workstations for nuclear engineering applications. Our Beowulf parallel workstation employs 12 PC-based processing modules integrated with dedicated fast ethernet network.

This paper presents results from a series of computations that measure the scaling characteristics of Beowulf. The evaluation includes the Monte Carlo transport code MCNP4B based on PVM and a FDM code for neutron diffusion equation, which we developed, based on MPI.

In MCNP calculations, the parallel efficiency of $\sim 90\%$ is observed for the shielding problems, and much higher speedups are possible if the number of histories is increased by orders of magnitude. However, only minor additional speedups are expected to be possible for the criticality problems.

In the case of FDM, the parallel efficiency is lower relative to MCNP calculations. But if we increase the problem size or apply the realistic three-dimensional neutron diffusion or transport problems, we will get good parallel efficiency.

Our Beowulf cluster shows that this architecture provides a new operating point in performance to cost for high performance workstations.

1. Introduction

Beowulf is a combination of hardware, software, and usage which yields a domain of computing that is scalable, exhibits excellent price-performance, provides a sophisticated and robust environment, and has broad applicability[1]. All Beowulf-class systems comprise clusters of PCs, sometimes referred to as a “pile of PCs”. Beowulf-class systems exclusively employ common off-the-self technology, usually targeted to the mass market where cost benefits of mass production and distribution drive prices down and PVM[2] and MPI[3] are preferred distributed application programming environments.

Our Beowulf parallel workstation architecture comprises 1 console node and 12 compute node which incorporates the Intel Celeron 300A processor, which delivers greater cost-effective computational power than other members of the Pentium II processor family. Each compute node has installed 128 MBytes of SDRAM for a total system main memory of 1536 MBytes. The processing elements are interconnected by dedicated Baystack 350T-HD fast ethernet switch with backplane bandwidth of 1.2 Gbps. Console node includes separate ethernet interfaces to external LAN's for remote access and data transfer.

The nuclear field, with its high demand for accuracy, reliability, and large volume of data in a typical calculation, is a case example which needs high performance computing. Two of the nuclear fields most demanding computational problems, namely the neutron diffusion and neutron transport problems are composed of deterministic and stochastic methods.

Monte Carlo method as stochastic transport method is becoming more widely used in the field of radiation transport. While the Monte Carlo method is desirable for its ability to simulate particle transport through complex geometries and its continuous energy and angular representation, it is a very computationally intensive method. In fact, the computational requirements of the Monte Carlo method have restricted its use in the past and in some cases, continue to restrict its use even now. The steady increase in computational performance has made Monte Carlo calculations for real applications possible. However in order to make these calculations practical, order of magnitude increases in performance are necessary. Fortunately, the Monte Carlo method is inherently parallel (particles are simulated independently), and thus has the potential for near linear speedup with respect to the number of processors[5].

Finite difference method in order to solve neutron diffusion equation requires long computing time and a large amount of computer memory to obtain accurate solutions for a reactor analysis which typically involves calculation of the neutron flux distributions and effective multiplication factors of a reactor core under various configurations and operating conditions. One way to decrease the computing time is parallel processing. However, in the case of linear algebraic calculation, the time required to update one iterative value is very small in comparison to the time required to communicate it between processors of message passing computer in this case. To reduce the amount of interprocessor data communication, domain decomposition techniques are widely used.

The parallel performance of any code is dependent on several factors. These include: (1) the grain size, (2) the fraction of the code that is executed in parallel, (3) the amount of data that is passed between processors, and (4) load imbalances.

This paper presents results from a series of computations that measure the scaling characteristics of Beowulf. The evaluation includes the Monte Carlo transport code MCNP4B based on PVM and a FDM code for neutron diffusion equation, which we developed, based on MPI. Both the scaling properties of these two applications and the communication overhead encountered will be presented. All computations are done on disk-less compute nodes.

In this paper, parallel efficiency $E(P) = [S(P)/P] \times 100$ with speedup $S(P) = T_s/T_p$ is used in order to analyze parallel performance, where T_s and T_p are sequential and parallel computing times, respectively, and P is the number of processors.

For all calculations, the DQS batch system was used in an attempt to get improved system performance, more equally distributed loads, and subsequently meaningful comparisons of parallel performance. While it is more desirable to perform this type of analysis on dedicated processors, the DQS software seems to do a pretty good job of resource distribution.

2. Parallel MCNP Calculations

Recognizing the nature of the Monte Carlo method and the trends in available computing, the code developers at Los Alamos National Laboratory have implemented the message-passing software package Parallel Virtual Machine (PVM) into the general purpose Monte Carlo radiation transport code MCNP (version 4B)[4]. The PVM package was chosen by the MCNP code developers because it supports a variety of communication networks, several UNIX platforms, and heterogeneous computer systems. This PVM version of MCNP has been shown to produce speedups that approach the number of processors, and thus, is a very useful tool for transport analysis.

As mentioned, the current version of MCNP makes use of the PVM message passing library. The parallel MCNP calculation proceeds in the following manner: (1) the master task reads and processes the input, and sends the common information to the subtasks, (2) the master task then calculates the work load for each processor based on the total number of histories to be done and a user supplied synchronization setting, and sends their work assignment to the subtasks, (3) the master task saves the results of its calculations, receives the results from the subtasks, and then combines the results to produce the tally and statistical information for output processing. Steps (2) and (3) are then repeated until the total number of particle histories have been completed.

For non-criticality problems with the MCNP code, the grain size can be controlled by simply reducing the number of synchronizations. For criticality problems, on the other hand, processors must share information after each cycle to estimate the fission generation and to provide the fission source for the next cycle. However, depending on the problem (in particular, the spatial distribution of the source), it may be possible to increase the number of histories per cycle and decrease the number of total cycles, thus increasing the granularity.

The fraction of MCNP that is not executed in parallel is the initial input file processing and the final output processing. For small problems this is an insignificant amount of time, while for large problems the penalty may be noticeable. However, generally for large problems, which require a correspondingly large number of histories, the total calculation time is large, and thus the penalty becomes insignificant.

Load imbalances are possible for problems with small granularity (relatively few histories per processor) and when running on non-dedicated machines. The small granularity load-imbalances can be directly related to the random nature of particle transport (i.e. particle histories may differ significantly, and thus the time required to simulate them may also differ significantly) but the current version of MCNP offers dynamic load-balancing and fault tolerance even if in the heterogeneous computing networks.

With these considerations in mind, two real-world radiation transport problems were selected for the performance analysis. These include: (1) a fusion reactor shielding type problem and (2) a medical reactor criticality problem.

The first problem is a three-dimensional model of the KSTAR tokamak[6], and is depicted in Fig. 1. The objective of this problem is to calculate nuclear heating rates at the superconducting magnets in the KSTAR tokamak. Instantaneous nuclear heating of the superconducting magnets is of concern for safe operation of cryogenically cooled superconductors. The supercritical liquid helium maintains the coil temperature around 4.5K. It is important to assess any extra heat loading to these coils, which could cause quenching.

In an effort to balance the criteria of a realistic calculation and the criteria of a reasonable amount of required computer time, calculations were performed for one million neutron particle histories and the induced photons were also considered and the number of dumps to the MCNP restart (runtpe) file was limited to two (one at the beginning and one at the end). On a single processor, this calculation required 217 minutes of wall-clock time (on a HP C180 platform, this calculation required 210 minutes of computing time). The parallel speedups and efficiency associated results with using 2, 4, 8, and 12 processors are given in Table 1. I/O processing times through NFS are also included in the computing times in these MCNP calculations.

A BNCT medical reactor[7] for criticality calculation is shown in Fig. 1. This problem uses the MCNP lattice capabilities and its objective is to calculate the value of k_{eff} for the system. In the interest of computational time, the calculation was performed for 200 cycles with 5000 neutrons per cycle. The number of dumps to the MCNP restart (runtpe) file was again limited to two. On a single processor, this calculation required 77 minutes of computing time. The parallel speedups and efficiency associated with using 2, 4, 8, and 12 processors are given in Table 2.

The decrease in efficiency with increasing number of processors is again observed. However, for criticality calculations the grain size does not increase with larger number of cycles, because the processors must communicate after every cycle. The grain size does, however, increase with greater number of histories per cycle. Nevertheless, dramatic increases in the efficiency are not feasible with criticality calculations.

Table 1: Performance of KSTAR tokamak Nuclear Heating Problem

Number of Processors	Computing Time (min)	Speedup (T_s/T_p)	Efficiency (%)
1	217	-	-
2	110	1.97	96
4	57	3.81	95
8	30	7.23	90
12	21	10.33	86

Table 2: Performance of a Medical Reactor Criticality Problem

Number of Processors	Computing Time (min)	Speedup (T_s/T_p)	Efficiency (%)
1	153	-	-
2	79	1.94	0.97
4	42	3.64	0.91
8	26	5.88	0.74
12	22	6.95	0.58

3. Parallel Finite Difference Method

The solution of steady-state two-group neutron diffusion equations is obtained through the standard inner/outer iteration algorithm. Throughout the iterations, the total computing time is mainly determined by such factors as the computing time for one inner iteration, average number of inner iterations, number of neutron groups, and number of outer iterations. The inner iteration is usually the most time-consuming process. Note that flux does not have to be completely converged in each inner iteration. The number of inner iterations is frequently fixed and the SOR iterative method is frequently used as inner iteration method[8].

The SOR iterative method is an important solver for large linear systems. It is also a robust smoother as well as an efficient solver of the coarsest grid equations in the multi-grid method. However, the SOR method is essentially sequential in its original form. With the increasing use of parallel computers, several parallel versions of the SOR method have been studied by a number of authors. In this paper, we elect to use the Red-Black line SOR method to complete the inner iterations in a parallel manner. For a two-dimensional geometry, this approach produces L/2 tridiagonal matrix systems that can be solved in parallel, where L is the total number of lines and one proceeds to both forward and backward sweeps through L/2 lines in this manner.

The whole iterative procedures in parallel computing are summarized as follows:

1. Divide the reactor into subdomains suitable to the number of processors available.

2. Guess initial k, ϕ_g.

3. Perform the inner iterations in parallel. Also, each processor calculates a partial inner product for the eigenvalue calculation, checks the convergence of fluxes simultaneously, and then sends the results to a central processor.

4. The central processor updates k by using the partial inner products fanned in from other processors and checks the convergence of k and ϕ_g on whole domain. If the convergence conditions are satisfied, the processor sends a message of "stop" to the waiting processors; otherwise, go to step 5.

5. The central processor distributes the updated global eigenvalue k to other processors, orders them to continue, and resumes its own assigned work.

6. Calculate the new fission source using the new updated k and ϕ_g and then go to step 3.

We present some timing study results for a real PWR core at the beginning of cycle 1 of Ulchin-1, which is a Framatome-type PWR rated at 2775 MW (thermal). The reactor core consists of 157 fuel assemblies, each of which is 21.504 cm wide, with different types of 17×17 pin cells. The octant core configuration showing the fuel loading pattern with three different enrichments is illustrated in Fig. 2.

With uniform mesh sizes, both sequential and parallel computations of the problem are performed using the Red-Black line SOR method combined with the power method. The Red-Black line SOR method gives the same convergence history regardless of serial or parallel so we fix outer and inner iteration numbers to compare parallel performance. Outer and inner iteration numbers are 300 and 4, respectively. I/O processing times are not included in the computing times in this calculation.

When the whole domain is discretized into a 170×170 uniform mesh system ($\Delta x = \Delta y = 1.0752 cm$), this calculation required 69 seconds of computing time on a single processor. The parallel speedups and efficiency associated results with using 2, 4, 8, and 12 processors are given in Table 3. In the case of a 340×340 uniform mesh system ($\Delta x = \Delta y = 0.5376 cm$), this calculation required 275 seconds of computing time on a single processor. The parallel speedups and efficiency associated results with using 2, 4, 8, and 12 processors are given in Table 4. As shown in Table 3, the communication overhead is fairly large for all processors because the problem size is small, and thus, the aspect ratio (interface to volume nodes ratio) is fairly large.

Table 3: Parallel Performance of 170 × 170 Mesh System

Number of Processors	Computing Time (sec)	Speedup (T_s/T_p)	Efficiency (%)
1	69	-	-
2	38	1.82	91
4	21	3.29	82
8	13	5.31	66
12	9	7.67	64

Table 4: Parallel Performance of 340 × 340 Mesh System

Number of Processors	Computing Time (sec)	Speedup (T_s/T_p)	Efficiency (%)
1	275	-	-
2	140	1.96	98
4	74	3.72	93
8	41	6.71	84
12	30	9.17	76

4. Conclusions

The high demand for powerful computers that can reduce the long computing times required for rigorous design and accurate analysis of nuclear design leads us to parallel calculations. In this work, we constructed Beowulf which uses commodity software like the GNU/Linux operating system, Parallel Virtual Machine (PVM) and Message Passing Interface (MPI) for this parallel calculation.

The price-performance of Beowulf-class systems is excellent, often substantially exceeding any other form of computing system. As a result, it makes possible access to parallel computing at an unprecedented degree due to its low entry level cost.

In MCNP calculations, the parallel efficiency of $\sim 90\%$ is observed for the shielding problems, and much higher speedups are possible if the number of histories is increased by orders of magnitude. However, only minor additional speedups are expected to be possible for the criticality problems. The use of parallel computing (even non-dedicated) is capable of decreasing the amount of time required for real-world transport calculations by more than an order of magnitude.

In the case of FDM, the parallel efficiency is lower relative to MCNP calculations. But if we increase the problem size or apply the realistic three-dimensional neutron diffusion or transport problems, we will get good parallel efficiency.

References

[1] Daniel Ridge, Donald Becker, Phillip Merkey, Thomas Sterling Becker, and Phillip Merkey, "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs," Proceedings, IEEE Aerospace, 1997

[2] A. GEIST, et al, "PVM 3 User's Guide and Reference Manual," Oak Ridge National Laboratory ORNL/TM-12187, 1993

[3] Marc Snir, et al, "MPI: The Complete Reference," MIT Press, 1996

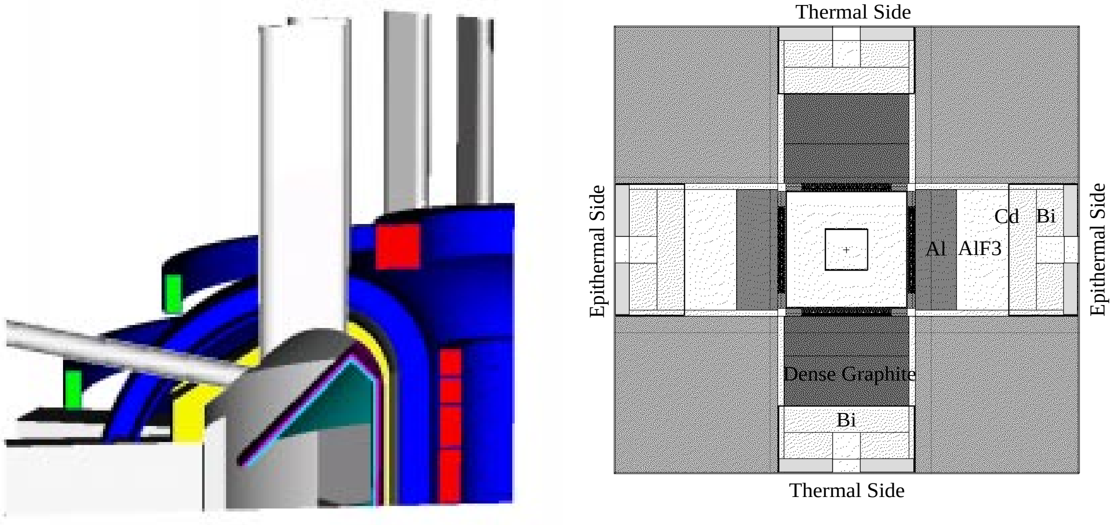
[4] J. F. Briesmeister, "MCNP – A General Monte Carlo N-Particle Transport Code, Version 4B," LA-12625-M, Los Alamos National Laboratory, 1993

[5] J. C. Wagner "Parallel MCNP Monte Carlo Transport Calculations with MPI," internal report, 1996

[6] Jeong Hwan Park and Nam Zin Cho, et al., "Neutronics Aspects of the KSTAR Tokamak," Proc. of the 20th Symposium on Fusion Technology, September 7-11, 1998, p.1621, 1998

[7] Jeong Hwan Park, "Design of a Medical Reactor Generating High Quality Neutron Beams for Boron Neutron Capture Therapy," KAIST MS Thesis, 1997

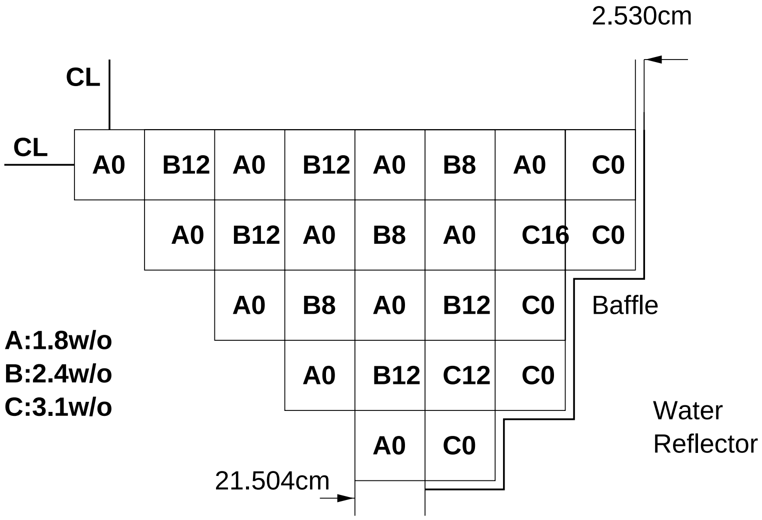
[8] Y. H. Kim and N. Z. Cho, "Parallel Solution of the Neutron Diffusion Equation with Domain Decomposition Method on a Transputer Network," *Nuclear Science and Engineering*, **114**, 252, 1993



(a) KSTAR tokamak

(b) A Medical Reactor

Figure 1: Geometrical Models for MCNP Calculations



A0	B12	A0	B12	A0	B8	A0	C0
	A0	B12	A0	B8	A0	C16	C0
		A0	B8	A0	B12	C0	
			A0	B12	C12	C0	
				A0	C0		

Figure 2: Ulchin-1, Cycle-1, Core Configuration