Technical Report UEC-IS-2004-6, Version 2004-06-14
Graduate School of Information Systems, The University of Electro-Communications

Solving the 24-queens Problem using MPI on a PC Cluster

Kenji Kise†, Takahiro Katagiri†, Hiroki Honda†, and Toshitsugu Yuba†
† Graduate School of Information Systems
The University of Electro-Communications

Abstract

The N-queens is a problem to place N queens on an N × N chess board such that no queen can attack another. We designed a sequential program which attains an improvement in speed of from 7% to 14% compared to current programs. The proposed program is parallelized using MPI, and the number of solutions for the 24-queens problem is calculated **for the first time**. *The main findings of the present study are as follows: (1) The optimization of memory reference and control structure increases speed from 7% to 14% in the sequential program. (2) A master-worker scheme is effective for parallelization. (3) The Hyper-threading technology of the Pentium4 processor provides an increases in speed of approximately 30%. (4) In the solution of a real problem, it is important to consider the efficiency of the entire system.*

1 Introduction

The N-queens is a problem to place N queens on an N × N chess board such that no queen can attack another. For example, the 6-queens problem has four solutions, as shown in Figure 1. This problem is commonly used as a benchmark program [1, 2] for computer systems and as an example in computer science.

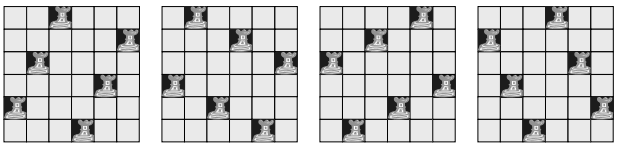

Figure 1. Four solutions to the 6-queens problem.

The N-queens problem was originally introduced as the 8-queens problem by a chess player named Max Bazzel in 1848[3]. Since its introduction, this problem has attracted the attention of several famous mathematicians, including Carl Gauss. Recently, the N-queens problem has been discussed in the context of computer science and has been used as an example for the backtracking algorithms[4], the divide and conquer paradigm[5], and constraint satisfaction problems.

We attempted to speed up and parallelize the sequential N-queens program for the purpose of calculating the solution of the 24-queens problem, which was not obtained until recently[6].

2 Estimation of the computation time of the solution to the 24-queens problem

The proposed program is based on a heavily optimized C program written by Jeff Somers[7]. Somers program is almost twice as fast as the world record breaking program used to calculate the solution to the 23-queens problem.

The proposed program is from 7% to 14% faster than the Somers program. This increase in speed is achieved by the optimization of memory reference and control flow structure.

The processing speed of the proposed program is measured, and the time required to solve the 24-queens problem is estimated. We measured the time on a computer having a Pentium4 Xeon 2.8-GHz processor, which is widely used and cost-effective. The computation time from N=17 to N=20 is acquired by measurement. Based on the obtained rate of increase, the computation time from N=21 to N=24 is estimated. The problem size N and computation time (seconds and days) are summarized in Table 1. The estimated time is indicated in bold.

The estimation summarized in Table 1, reveals that 2,815 days are required in order to calculate the solution of the 24-queens problem.

3 Parallelization of the N-queens Program

The estimated computation time of 2,815 days or 7.7 years for the calculation of the solution of the 24-queens problem is not realistic. However, the program can be parallelized and the computation time can be shortened to 44 days by using a PC cluster having 64 CPUs. In this section,

Table 1. Number of solutions and estimated time required to solve the 24-queens problem on a computer having one processor.

size N	number of solutions	our code (sec)	our code (day)	Somers code (sec)
17	95,815,104	55.5	0.00	59.5
18	666,090,624	384.8	0.00	440.5
19	4,968,057,848	3168.6	0.04	3417.5
20	39,029,188,884	24329.7	0.28	27745.8
21	314,666,222,712	**243297**	**2.82**	–
22	2,691,008,701,644	**2432970**	**28.16**	–
23	24,233,937,684,440	**24329700**	**281.59**	–
24	–	**227239317**	**2815.94**	–

a technique for the efficient parallelization of an N-queens program is summarized.

Parallelization of a sequential program requires the calculation to be divided into a large number of tasks. An algorithm that places queens in order from the upper row to the lower row on an N $\times$ N chess board is adopted. After the initial placement of a number of queens, the processing of the remaining queens is handled as one task. Each generated task is calculable in parallel as an independent problem. The first four decomposed tasks for the 6-queens problem are shown in Figure 2. In the figure, each task is generated after the placement of the first two queens.

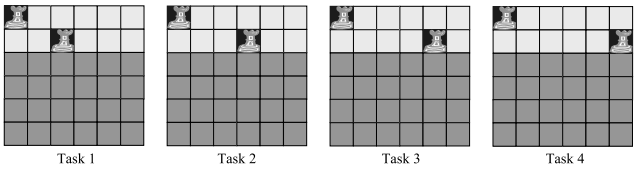

Figure 2. The first four decomposed tasks out of 24 tasks for the 6-queens problem. Each task is generated after the placement of the first two queens.

Table 2. Number of decomposed tasks and estimated time per task.

size N	number of tasks	time (sec) per task
20	30,880	0.91
21	41,176	6.13
22	49,480	45.93
23	64,072	354.66
24	75,516	3009.15

The calculation after the placement of the first four queens is considered to be one task with respect to both the number of tasks and the calculation time for an individual task. Table 2 summarizes the problem size N, the number of tasks, and the average time required to complete one task. In the case of 24-queens, the set of calculations is broken up into 75,516 tasks. The average time to complete one task is as long as 3,009 seconds, and it is possible to conceal the parallelization overhead.

The task-scheduling algorithm is examined next. In the case of the 23-queens problem, the calculation is broken up into 64,072 tasks. Measurement revealed that the computation time of each task varied from 45 seconds to 840 seconds. The 64,072 tasks were sorted according to calculation time, and the distribution of the calculation times is shown in Figure 3. The vertical axis shows the computation time. Prediction of the calculation time and the adoption of static scheduling appear difficult. We therefore parallelize our N-queens program using a master-worker scheme. One process is assigned as a master and all other processes operate as workers. MPI, a standard of parallel programming, is used for parallelization.

4 Parallelization efficiency of the N-queens program

4.1 Efficiency of the parallelized program

This section discusses the efficiency of the parallelized program. The measured increase in speed when the program is run on a PC cluster having four Pentium4 Xeon 2.8-GHz processors is summarized in Figure 4. The horizontal axis is the problem size N, and the vertical axis is the speedup based on the elapsed time of the sequential program without the parallelization overhead. Hyper-threading technology, which uses one physical CPU as two logical CPUs is available on Pentium4 Xeon processor. The increase is measured for configurations in which the Hyper-threading is either on or off.

The speedup without Hyper-threading technology is discussed. When the problem size is N=16, the elapsed time is short and the overhead for parallelization becomes notable. The speedup of 3.16 is much lower than the number of processors. On the other hand, the speedup approaches

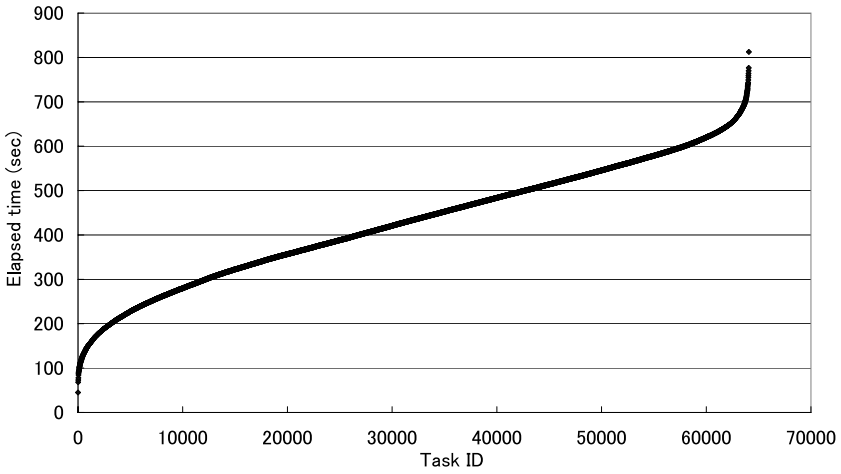

Figure 3. Distribution of the elapsed time of 64,072 tasks in 23-queens. All tasks are sorted by elapsed time.

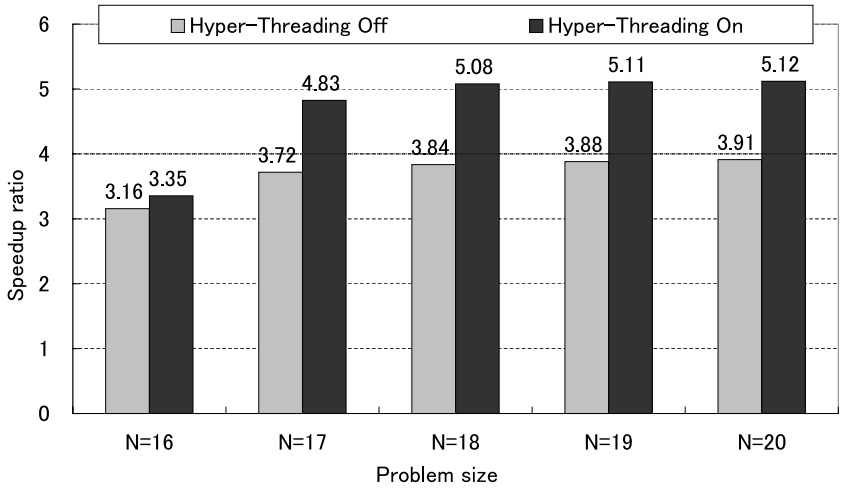

Figure 4. Increase in speed for calculation performed on a PC cluster having four physical CPUs.

the ideal value 4 as problem size becomes large. When the problem size is N=20, the speedup is 3.91 using four CPUs. This is as high as 97.8% of the ideal speedup for four CPUs.

The speedup using the Hyper-threading technology is discussed. In solving the 24-queens problem, the part of where the problem size becomes large is important. When the problem size is N=20, the high rate of speedup of 5.12, which exceed the number of physical CPUs, is achieved. Compared to the case in which Hyper-threading is off, a 30% improvement in speed is achieved.

These results show that Hyper-threading technology works effectively for the N-queens program. In the solution of a real problem, it is important to consider the efficiency of the entire system.

4.2 Speedup with respect to the increase in the number of nodes

This section discusses the speedup with respect to the increases in the number of nodes. The 34-node PC cluster called FireCore is used in our evaluation. Each computer node is an SMP having two Pentium4 Xeon 2.8-GHz processors. Clustermatic 4[8] is used as middleware. Photos of a computer node and the FireCore cluster are shown in Figure 5.

The speedup for the 20-queens problem as the number of nodes increases is shown in Figure 6. The vertical axis is the speedup based on the elapsed time with one node. Each node uses Hyper-threading technology and four processes are generated per node. Figure 6 indicates that an improvement in speed of approximately 35 times is attained using

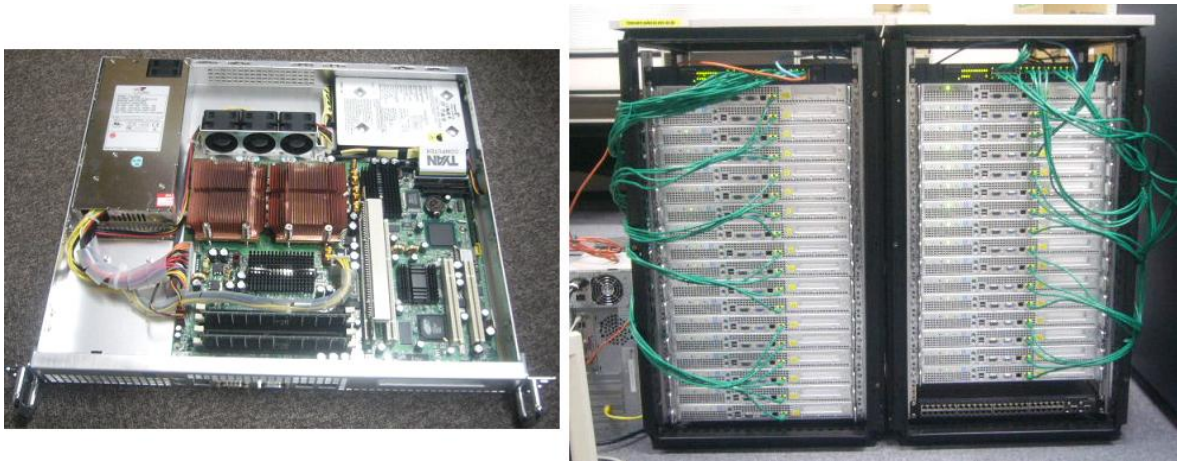

Figure 5. Photographs of a node computer having two processors and the 34-node PC cluster called FireCore.

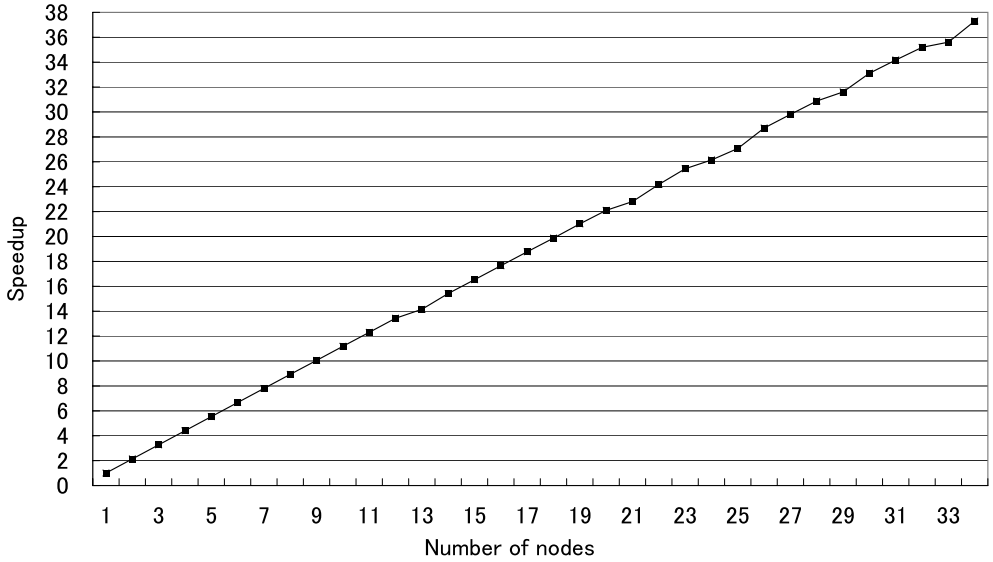

Figure 6. Increase in calculation speed of the 20-queens solution on the FireCore Cluster as the number of nodes increases. Hyper-threading technology is used.

Table 3. Number of solutions calculated and the elapsed calculation time using the FireCore Cluster to solve the N-queens problem.

size N	number of solutions	time (sec)
1	1	–
2	0	–
3	0	–
4	2	–
5	10	–
6	4	–
7	40	–
8	92	–
9	352	–
10	724	–
11	2,680	–
12	14,200	–
13	73,712	–
14	365,596	–
15	2279,184	–
16	14,772,512	–
17	95,815,104	–
18	666,090,624	12.02
19	4,968,057,848	42.56
20	39,029,188,884	286.09
21	314,666,222,712	2497.82
22	2,691,008,701,644	21002.19
23	24,233,937,684,440	204750.38
24	227,514,171,973,736	1879831.94

32 nodes.

The results in Figures 4 and 6 indicate that our parallelization of the N-queens program works well.

5 Calculation result for the N-queens problem

The number of solutions of the N-queens problem calculated using our program is summarized in Table 3. The elapsed time on the FireCore cluster is shown for cases in which problem size is larger than 18.

We successfully calculated the 24-queens problem for the first time. The bold letters in Table 3 indicate the number of solution and the elapsed time for the solution of the 24-queens problem. It required about 22 days. The computation was started at 3:00 a.m. on March 21, 2004 and ended at 9:00 p.m. on April 11, 2004. The value 227,514,171,973,736 was acquired as the number of solutions of the 24-queens problem.

6 Discussion

6.1 Verification

We next argue the validity of the obtained number of solutions. As one means by which to confirm the validity of the obtained number, we confirmed that the number of solutions for the 17-queens problem to the 23-queens problem calculated using our program was in agreement with the number of solutions obtained in the past. This calculation was performed under an environment identical to that for the 24-queens calculation.

The computer used to obtain the number of solutions for the 24-queens problem uses highly reliable Pentium4 Xeon processors. The main memory functions as the ECC, which performs detection and correction of incorrect values. The FireCore cluster used to obtain the data is located in a temperature-controlled computer room. The obtained data was calculated on a computer having these high-reliability components and was located in a desirable environment. The reliability of the data is high.

6.2 The N-queens problem as a benchmark program

Our program is described compactly using the C language and its code is easy to understand. All codes are described in one file, and compiling and execution are very easy.

From these features, our program is suitable for use as a benchmark program for computer systems. Our sequential N-queens program and an MPI parallelized version are available from our web site. The program can be used as a benchmark program and as a base for solving the 25-queens problem.

6.3 Solution on a large-scale PC cluster

Large-scale PC clusters, such as the ASCI Lightning, having 2,816 processor, were constructed. The solution of large problems may be calculated on a large-scale PC cluster. In the case of 2,816 processors, as compared with the PC cluster of 68 processors used for the 24-queens problem, an improvement in speed of approximately 40 times can be expected. If the large-scale cluster is used, the solution to the 24-queens problem is calculable in approximately 14 hours and the solution to the 25-queens problem may be calculable in approximately six days.

However, in order to run on a large-scale cluster, it is necessary to describe a program under the consideration of the failure of a computer node. Moreover, it is difficult to use a very expensive large-scale cluster for a long time. If these problems are solved, the use of a large-scale cluster will be an attractive choice.

7 Summary

We attempted speedup and parallelization of a sequential N-queens program for the purpose of calculating the number of solution of the 24-queens problem.

We designed a sequential program which achieves a 7% to 14% improvement in speed compared to the currently used program. Our program is parallelized using MPI and the number of solutions for the 24-queens problem is calculated for the first time using a PC cluster.

The primary findings of the present study are as follows: (1) Optimization of memory reference and control structure achieves a 7% to 14% increase in speed for a sequential program. (2) A master-worker scheme is effective in the parallelization. (3) The Hyper-threading technology of the Pentium4 processor achieves an increase in speed of approximately 30%. (4) In the solution of a real problem, it is important to consider the efficiency of the system as a whole.

References

[1] J. R. Gurd and D. F. Snelling. Manchester data-flow: a progress report. In *Proceedings of the 6th international conference on Supercomputing*, pages 216–225, 1992.

[2] Michael D. Noakes, Deborah A. Wallach, and William J. Dally. The J-machine multicomputer: an architectural evaluation. In *Proceedings of the 20th annual international symposium on Computer architecture*, pages 224–235, 1993.

[3] Cengiz Erbas, Seyed Sarkeshik, and Murat M. Tanik. Different perspectives of the N-Queens problem. In *ACM annual conference on Communications*, pages 99–108, 1992.

[4] Solomon W. Golomb and Leonard D. Baumert. Backtrack programming. *J. ACM*, 12(4):516–524, 1965.

[5] Bruce Abramson and Moti Yung. Divide and conquer under global constraints: a solution to the n-queens problem. *J. Parallel Distrib. Comput.*, 6(3):649–662, 1989.

[6] http://www.research.att.com/ njas/sequences/. The On-Line Encyclopedia of Integer Sequences.

[7] http://www.jsomers.com/. The N Queens Problem, a study in optimization.

[8] http://www.clustermatic.org/. CLUATERMATIC! Redesigning the Cluster Architecture.