ICOT Technical Memorandum: TM-0131

TM-0131

A New Parallel Inference Mechanism Based on Sequential Processing

by
Y. Sohma, K. Satoh, K. Kumon,
H. Masuzawa and A. Itashiki
(Fujitsu Ltd.)

July, 1985

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

A NEW PARALLEL INFERENCE MECHANISM BASED ON SEQUENTIAL PROCESSING

Yukio Sohma, Ken Satoh, Koichi Kumon,
Hideo Masuzawa, Akihiro Itashiki
(Fujitsu Laboratories Limited)

We propose a new parallel inference mechanism which we call the KABU-WAKE method. In this method, an inference is made by a depth-first search in a processor element. However, if requested by another processor, a job is split up between PEs for OR parallel inference.

Thus, the overhead in each processor and for communication between processors can be minimized.

Through our experimental system, we found that the KABU-WAKE method was particularly effective for the application with large search tree.

1. INTRODUCTION

This paper discusses a new parallel inference method, and gives an evaluation of this method using our experimental system.

We have been researching the feasibility of realizing a high-speed inference machine that uses parallel processing. We propose a new parallel inference mechanism based on the idea of sequential inference, but extended further to include parallel inference. Chapter 2 describes our basic concept for parallel inference. Then Chapter 3 describes the features and operating principles of the KABU-WAKE method. In Chapter 4, an experimental system dedicated to the KABU-WAKE method is explained. Chapter 5 summarizes the results of experiments using this system.

2. OUR APPROACH TO PARALLEL INFERENCE

The inference speed can be improved by operating a number of identical processing elements (PE) in parallel. However, each PE must provide outstanding performance. Concerning a single PE i.e. sequential inference, technological accumulation is abundant. And there are already a number of speed-up techniques available. As for technological progress, it is usually gradual. And, technological advancement of slow and steady is necessary, especially in such

difficult research as parallel inference. Then we decided to proceed along the "graceful growth" spirit, and take a parallel inference method based on a sequential inference machine.

3. THE KABU-WAKE METHOD

The KABU-WAKE method is one of the parallel inference mechanisms for current Fifth Generation Computer Project.

3.1 Features

The KABU-WAKE method has the following features:

- Each PE sequentially processes a search tree by a depth-first search.

- Each PE splits its current tree for OR parallel processing only when requested by another PE.

These features have the following advantages:

- Low overhead in each PE
  In a PE, processing is the same as for sequential inference -- no special processing is necessary for parallel inference. Therefore, each PE works at the same speed as for sequential inference.

- Little communication between PEs
  A job is split and given only when requested by another PE, which reduces the amount of required communication. In addition, since a job is split near the root of the search tree, granularity of the job can be made as big as possible.

- Suitable number of OR processes
  Since the number of OR processes is limited to the number of PEs, there is no danger of an unexpected increase in the number of OR processes.

3.2 Principles of Operation

The operating principles of the KABU-WAKE method are explained here with some reference figures. Figure 1-a is an example of a data base and inquiry written in Prolog. We will omit arguments of the predicate to simplify the explanation. The arrow shows the ordinary flow of sequential inference. Figure 1-b shows the operating principles of the KABU-WAKE method. The part enclosed by thick lines shows the processing of a certain PE and corresponds to the processing indicated by the arrow in Figure 1-a, namely, sequential inference.

If a job request is made by another PE, the job is split

at the branch closest to the root of the search tree and one half is given to the PE. If another request comes, the job is split again at the branch next closest to the root of the tree and one half is given to the second PE. The job can be further split up among other PEs.

4. EXPERIMENTAL SYSTEM

We built an experimental system of parallel inference machine to test the effectiveness of the KABU-WAKE method quantitatively. We developed a hardware suitable for our method and installed the KABU-WAKE interpreter on that hardware.

4.1 System Configuration

Figure 2 shows the hardware system configuration. The system consists of 16 PEs (one PE for input/output) which are connected by two kinds of exclusive networks depending on their functions.

The system components are as follows:

- PE:
  This is a processing element on which is installed a parallel inference interpreter based on the KABU-WAKE method. If a PE receives a request from another PE during inference, the PE splits its current job and gives one half to the other PE.

- CONT network:
  This is a communication route for requesting job. PE status information, whether a PE is processing or not, is circulated on the network. A PE which is doing inference can check the status information to give a free PEa part of its job.

- DATA network:
  This is a communication route for transferring a split job(KABU).

4.2 Implementation

The followings are key points taken to realize the system.

(1) Unbinding of variables

When splitting a search tree, the variables contained in the split tree part must be changed to a status as if all the processes located left below of the split position had failed and backtracked.

To speed up this operation, we prepared an area for each variable which memorize the time it was bound. Each time a subgoal is made, new level number (which corresponds to the depth of subgoal) is assigned. And, for variables bound during the processing of that subgoal, the same level number with the subgoal is

tagged. When a tree is split, we can determine whether binding should be released or not by comparing the level of subgoal to be split and the variable's level number.

In this method, the time required to release variables for splitting is prportional to the number of variables in the subgoal to be split. To use trailing stack will be another alternative, but we think it would take more time.

(2) Use of rule numbers

When a tree is split and transferred, it is transferred in the form of a subgoal. However, since a part of several definitions for that subgoal is already being processed, the other PEs must start from a subsequent definition. Therefore, we adopted a method in which a special predicate called a rule number is prepared and attached to the subgoal before transfer to indicate the position from which to start execution.

(3) Control of request

Some jobs cannot be split even when a request is received from another PE. If this happens frequently, PE performance will deteriorate.
To prevent this, we used a request reception flag to indicate whether a job can be split. This enables a PE to concentrate on its job without being disturbed by other PEs.

5. RESULTS OF THE EXPERIMENT AND THEIR EVALUATION

Data is still being collected from the experimental system. Although it is too early to make a proper evaluation, we would like to report so far and our preliminary evaluation.

5.1 Degree of Total Performance Improvement

Figure 3 shows the relationship between the number of PEs and execution time using an n-queen problem as a benchmark program. The values were obtained by averaging several or several tens of measured values. For the problem with large search tree (such as 8,9,10-queen), the execution time can be shortened in almost linear proportion to the number of PEs.

On the basis of the same data, Figure 4 shows the performance improvement ratio due to the increase of the number of PEs. Average activity ratio of PE is about 99% when the thirteen PEs are doing parallel inference for 10-queen problem.

5.2 Detailed Analysis

The following are the results of detailed analysis of the collected data.

(1) Performance of PE

Absolute performance

One PE has a performance of approximately 1 KLIPS, almost equal to C-prolog on vax11/780.

Overhead during sequential execution

We tried to create the system on the basis of sequential inference and reduce the overhead for parallel processing as much as possible. As mentioned before, however, overhead is required for memorizing binding timings of variables.

Therefore, we measured the percentage of the binding time processing out of the total processing. When the benchmark program (n-queen, quicksort) was executed using one PE, the value was less than 6%. In other words, when continuous processing is performed in one PE, our parallel inference interpreter performs as well as the sequential inference interpreter.

Overhead during parallel execution

Now let's look the processing time of one PE when a job is executed in parallel by several PEs.

We sampled the processing of a PE in time series to check the processing type, while a 7-queen was being executed by 13 PEs. Figure 5 shows the results.

The time required for UNIFICATION, GOAL FRAME and BTRACK is the inherent time in sequential processing. Others are the time required for parallel processing, in other word, overhead.

We found that the main overhead was for communications, splitting of job (find the splittable part, release the constraints and create a subgoal), and for renumbering variables to convert a given job into its internal expressions.

(2) Number of communications

Total number

Figure 6 shows the total number of communications made from the start to the end of a program. As you can see from the figure, the absolute number of communications varies slightly with the program but is almost linearly proportional to the number of PEs. In other words, if the number of PEs increases, the granularity of the split job becomes smaller, which increases communication.

If a perfect OR parallel processing is executed, the number of communications will increase exponentially in an n-queen problem, for large n. On the other hand, trying to reduce communication would require considerable memory for storing the OR processes. The KABU-WAKE method was found to achieve an almost ideal ratio of number of communications to number of PEs.

Time dependency

In the KABU-WAKE method, a job is split only when requested. When the input job is big, the amount of communication between processors is large until the job is split up and the processors become busy. However, while the PEs are busy processing portions of the job, there is almost no communication.

Figure 7 shows the characteristics of communication we obtained by executing an n-queen job. For a 6-queen problem, the amount of communication is almost constant from beginning to end. For 8-queen and 9-queen problem, however, there tends to be more communication at the beginning and end of the job, while there is little communication in between.

Since search tree is expected to become enormous in practical applications, this tendency will become more remarkable, and the ratio of communication time to total time should decrease greatly. Thus, KABU-WAKE method should prove most effective in practical applications.

6. CONCLUSIONS

We proposed the KABU-WAKE method as a new parallel inference mechanism based on sequential inference processing. Then, we built an experimental system, which we used to test the method. Our first evaluation indicates the method is effective. Specifically, overhead for parallel processing in PEs little, the amount of communication between PEs is relatively low for the number of PEs in the system, and large trees are handled very effectively.

We will study the following items in the future:

- Handling of multiple solutions in practical application and clarifying support functions required to the machines
- Decreasing overhead between PEs during subgoal transfer
- Feasibility of compilation in KABU-WAKE method

7. ACKNOWLEDGEMENT

The authors would like to thank to Division Manager Tanahashi and Department Manager Sato for their unfailing encouragement in our research, and Managing Director Yamada for giving us the opportunity to conduct the research. We are also grateful to Mr. Hayashi and Mr. Hattori for their helpful discussion on the comparisons of performance of sequential inference and parallel inference.

This research was entrusted by ICOT as part of the Fifth Generation Computer Project.

REFERENCES

[1] Kumon, et al., "PARALLEL INFERENCE PROCESSING SYSTEM -- IMPROVED CLAUSE UNIT PROCESSING METHOD", 30th meeting of Information Processing Society (First half of 1985) 7C-8 Japanese

[2] Itashiki, et al., "PARALLEL INFERENCE PROCESSING SYSTEM -- EXPERIMENT OF IMPROVED CLAUSE UNIT PROCESSING METHOD", 30th meeting of Information Processing Society (First half of 1985) 7C-7 Japanese

[3] J.S.Conery, D.F.Kibler, "Parallel Interpretation of Logic Programs", Proc. of the ACM conference on FPLACA, 1981.

[4] T.Moto-oka, H.Tanaka, et al., "The Architecture of a Parallel Inference Engine - PIE -", Proc. of the International Conference of Fifth Generation Computer Systems, 1984.

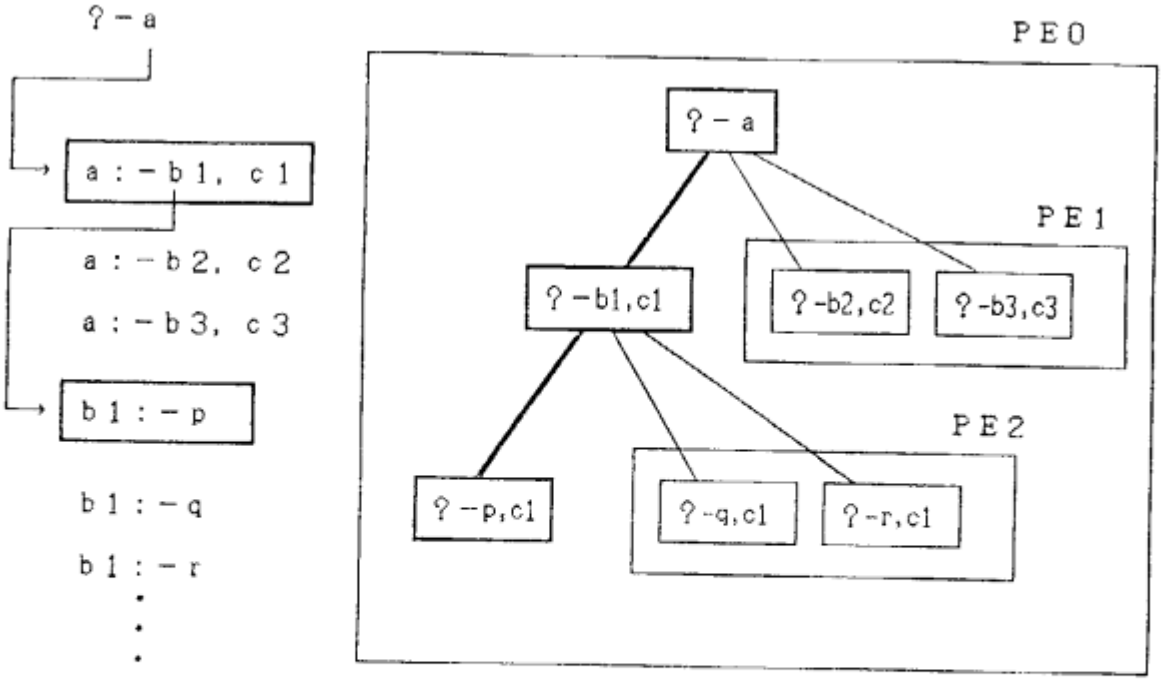

Fig.1 - a　　Fig.1 - b

Fig.1 Basic mechanism of KABU-WAKE

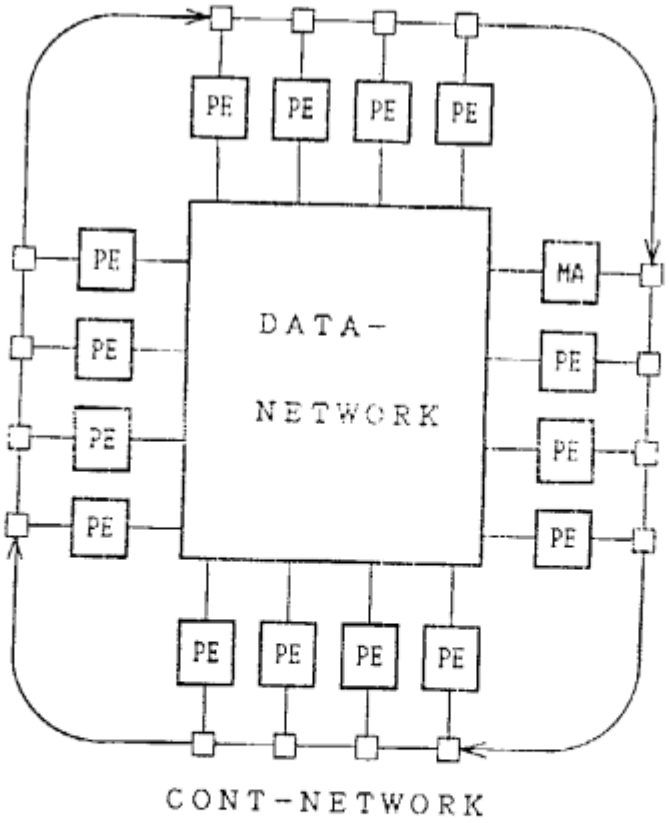

Fig.2 System Configuration of Experimental machine

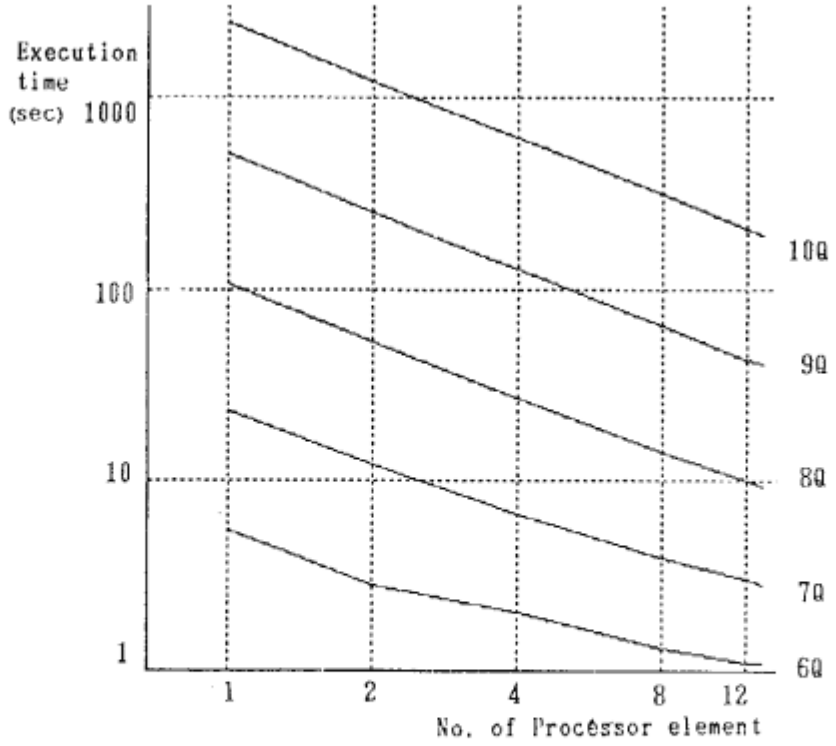

fig.3 Speed up factor due to parallel inference

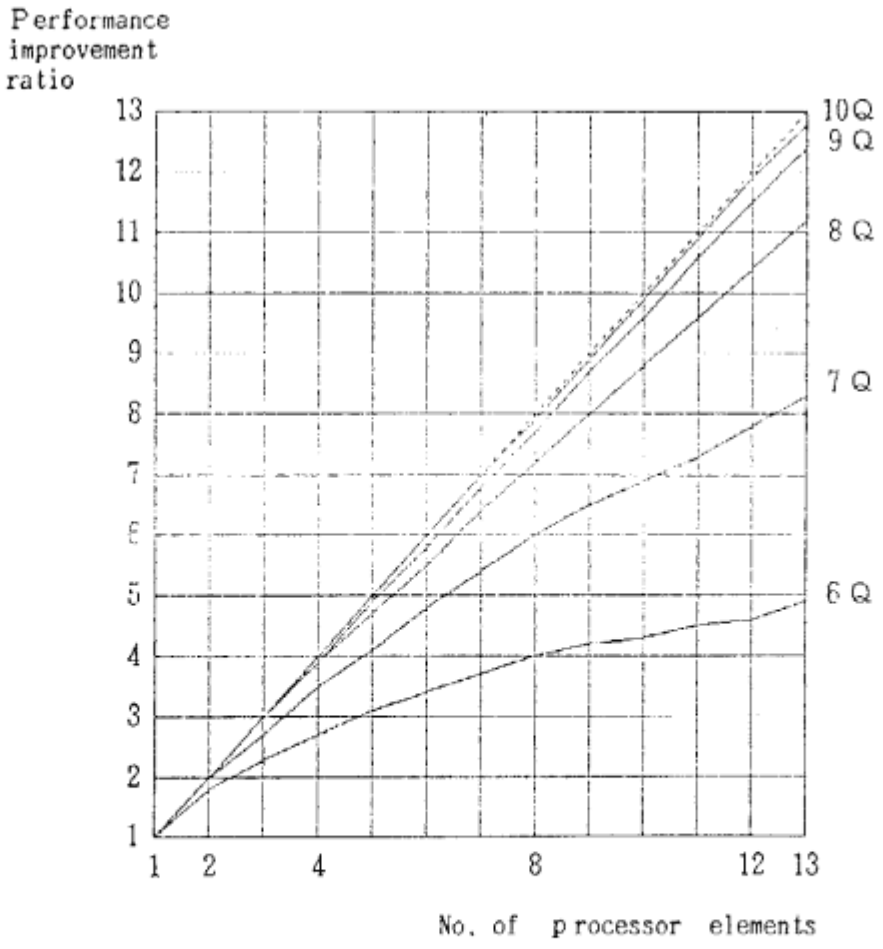

Fig.4 Performance improvement ratio due to parallel inference

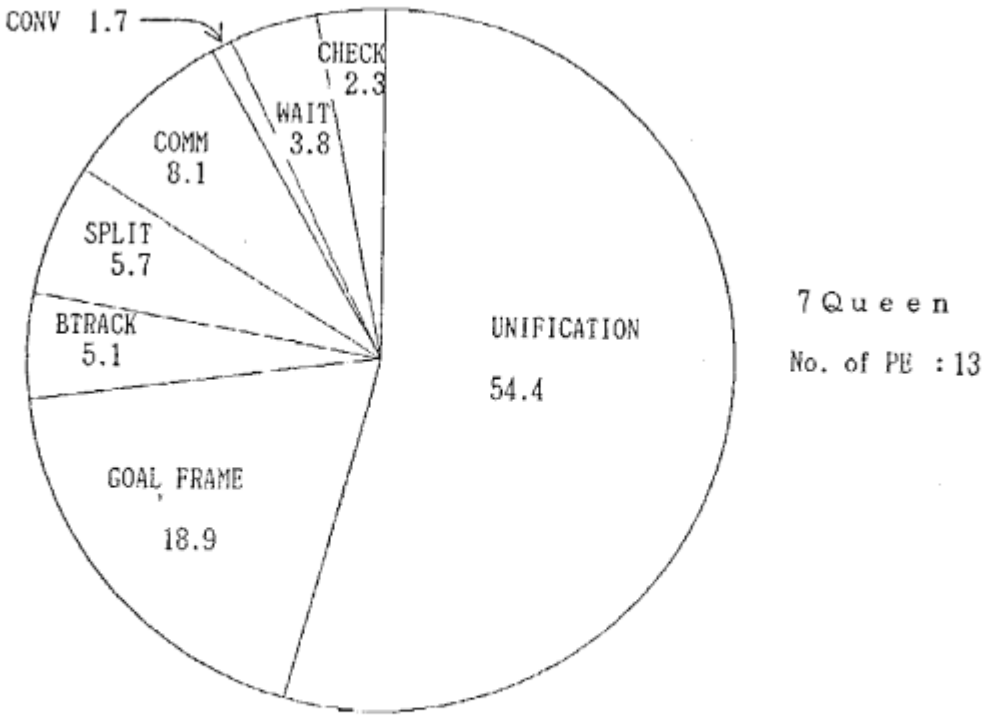

Fig.5 Details of execution time in processor element

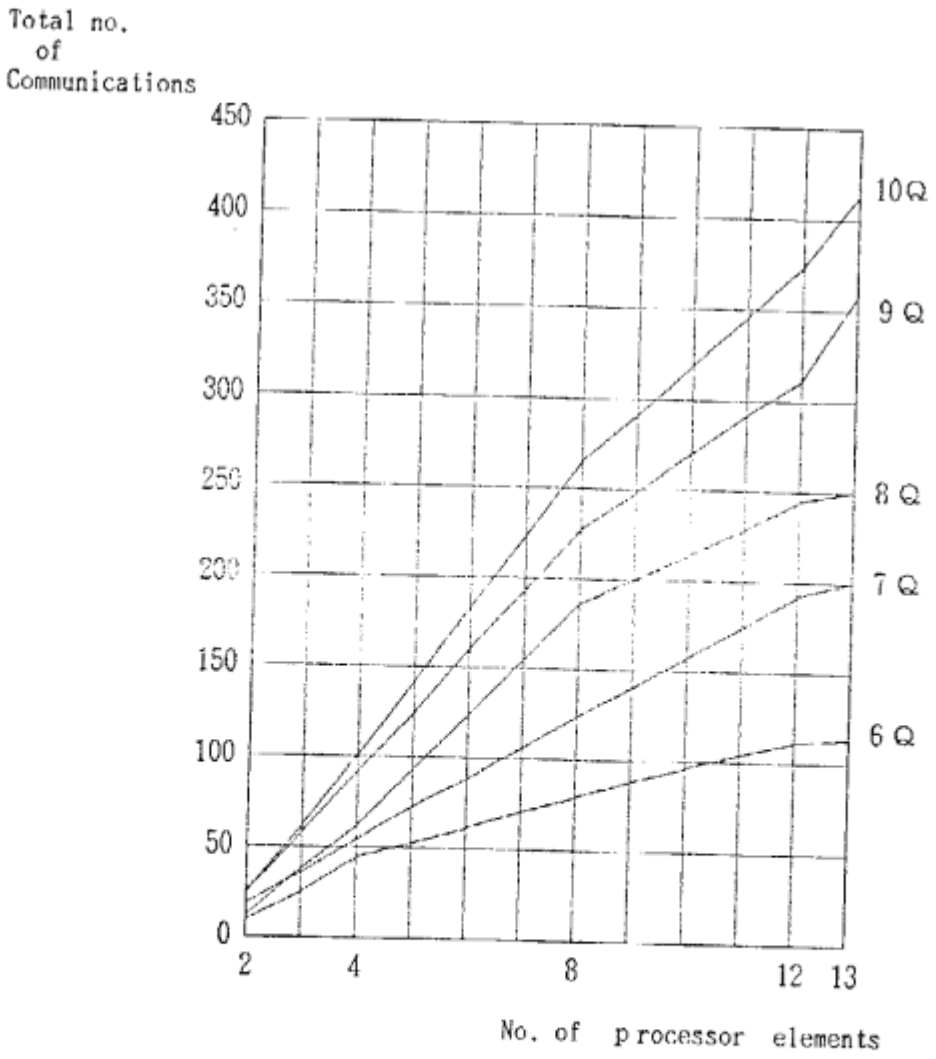

Fig.6 Total number of communications

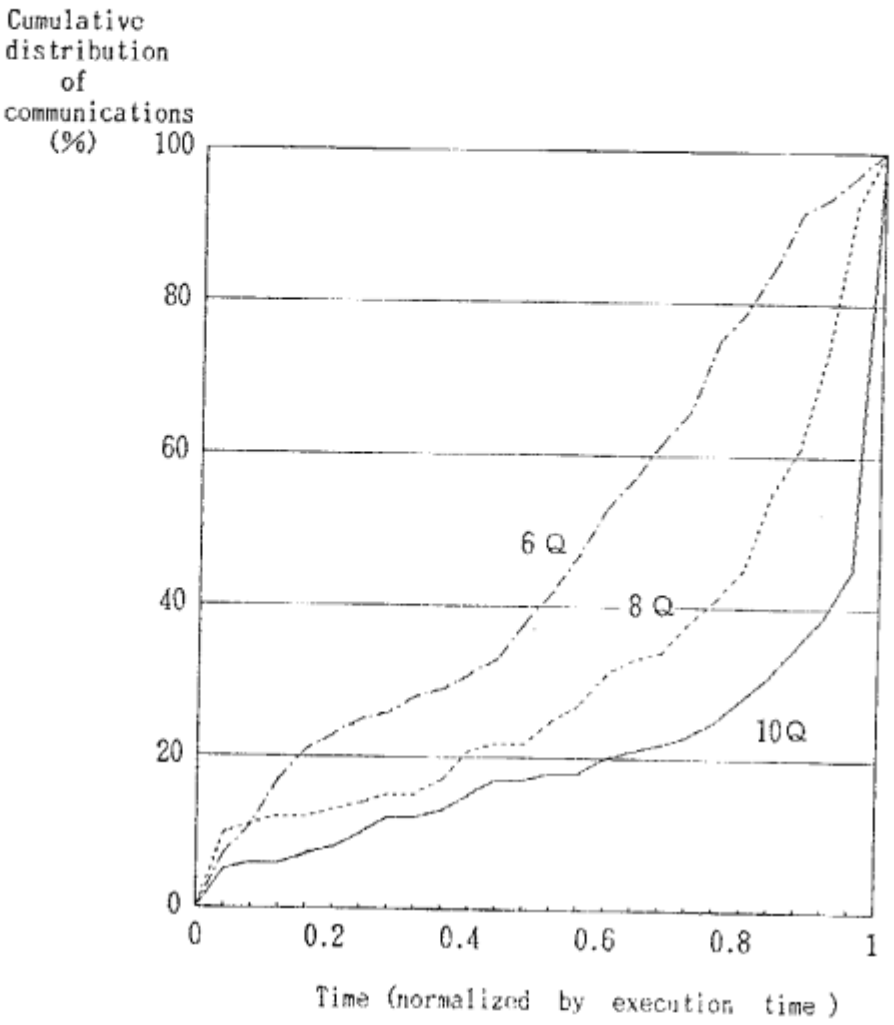

Fig.7 Dynamic change of the number of communications