ICOT Technical Report: TR-150

TR-150

KABU-WAKE: A NEW PARALLEL INFERENCE METHOD AND ITS EVALUATION

by

K. kumon, H. Masuzawa, A. Itashiki

K. Satoh and Y. Sohma (Fujitsu Ltd.)

March, 1986

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

KABU-WAKE: A NEW PARALLEL INFERENCE METHOD AND ITS EVALUATION

Kouichi Kumon, Hideo Masuzawa, Akihiro Itashiki

Ken Satoh and Yukio Sohma

FUJITSU LABORATORIES LTD. KAWASAKI

1015, KAMIKODANAKA, NAKAHARA-KU

KAWASAKI 211, JAPAN

ABSTRACT

The Japanese Fifth Generation Computer System (FGCS) project is working to develop a high-speed parallel inference machine. We have already proposed a new OR parallel inference method called the KABU-WAKE method[Kumon 85ab][Itashiki 85]. In the KABU-WAKE method there are a number of processing elements (PEs) each working sequentially, but the total system performs parallel inference. We are currently evaluating the method using a dedicated experimental system. We have found that simple search problems, such as the N-queen problem, are solved efficiently[Kumon 85ab][Sohma 85].

In this paper, we evaluate the KABU-WAKE method for a definite clause grammer (DCG) parser, as a more practical application than the N-queen problem.

This research was sponsored by MITI as a part of the Japanese FGCS project.

1. INTRODUCTION

The goal of the FGCS project is to build a high-speed inference machine that uses parallel processing. We have been studying ways of realizing a high-speed parallel inference machine. However, sequential inference machines based on a stack mechanism are currently faster and more efficient than machines using a parallel architecture. Thus, we tried to build an OR parallel machine based on sequential inference using a stack mechanism.

The performance of a high-speed parallel computer depends on the following:

- The performance of each processing element (PE)
- The overhead for parallel processing
- The number of PEs

In the KABU-WAKE method, we reduced the number of subtask transfers, enabling us to build a high-speed parallel inference machine using high-performance PEs.

2. THE KABU-WAKE METHOD

The KABU-WAKE method is based on sequential PROLOG execution. In conventional sequential PROLOG processing, such as in DEC-10 PROLOG, execution involves selecting an applicable rule from a PROLOG database, applying it to the current goal, and making a new goal. And if there is more than one applicable rule in the database, the system selects one and pushes the other on a stack to be executed later. If no rule can be applied, the system backtracks to the point at which it pushed alternatives on the stack, pulls the the alternatives from the stack, and resumes processing.

In the KABU-WAKE method, if one PE is performing sequential processing and another processor is idle, the idle processor signals to the busy processor. The busy processor finds its oldest unprocessed alternatives on the stack and tries to divide the search tree at that point. It then sends a portion of the task to the requesting PE and deletes the alternatives from its own stack. This is like splitting a living tree at a node, we call this divided subtask "KABU" in Japanese.

To create a subtask, the sending PE must backtrack temporarily to the node at which the alternatives were created, because binding of variables after that node may change the values of variables in the divided subtask. To void such bindings, the system records the variable binding time in each variable cell, and voids according to the time each variable was bound.

The KABU-WAKE method reduces the amount of communication between PEs, because there is no communication if there are no idle processors in the system. It also reduces the overhead within each PE, because PEs execute ordinary sequential processing, except for splitting subtasks.

The KABU-WAKE method also enlarges the subtask granularity, because a PE sends an entire subtask, rather than a single predicate or unit clause, which requires a lot of inference.

3. EXPERIMENTAL SYSTEM

The experimental system for the KABU-WAKE method consists of 16 PEs connected by 2 dedicated networks called the TASK-NETWORK and CONTROL-NETWORK.

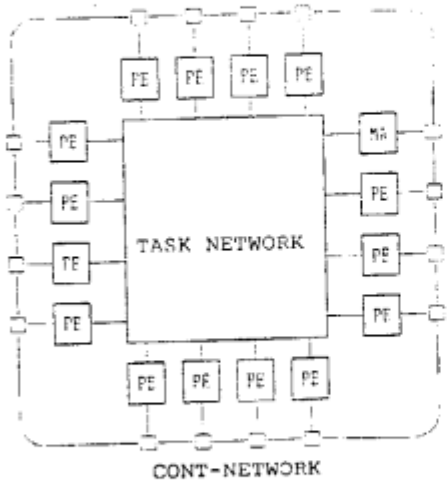

Fig. 1 Hardware Comfiguration of The Experimental Machine

The hardware configuration is shown in Fig 1. One of the PEs is called the manager and the others are called children. The manager is used as the console of the machine; a user inputs a query via the manager, and it is given to one of the children. The manager also collects the results from the child.

Each PE contains a general-purpose micro computer and adapters for the dedicated networks. Table 1 shows some characteristics of the PEs.

Table 1. Characteristics of PE

CPU:	68010 (10MHz)
MAIN MEMORY:	2MB
TASK-NETWORK:	2MB/s
CONT-NETWORK:	8μs/round (2MHz clock)

Each PE includes a KABU-WAKE processing system, and all PEs have the same PROLOG database.

The TASK-NETWORK and CONTROL-NETWORK are provided by following reasons:

- To reduce the overhead for subtask transfer
- To reduce the overhead for finding an idle processor

The network specifications are as follows:

* TASK-NETWORK

The TASK-NETWORK is an 8-bit parallel data transfer multi-stage network. It transiers variable-length packets from a sender to a specified receiver. Each packet has a destination address which is added by the sending PE; the multi-stage network sends the packet to the specified destination.

This network also has a broadcast mode, which can be used to deliver a PROLOG database from a manager PE to the children. Data is transferred between TASK-NETWORK and PEs' main memories by the direct memory access (DMA) controller.

The network throughput is about 2MB/s, however, main memories requires some wait cycles to the DMA controller, and data transfer by software between the DMA-accessible buffer and areas used for inference drop the throughput to 50KB/s.

* CONT-NETWORK

The CONT-NETWORK is a 16-bit parallel data transfer ring network. Each PE has a control network adapter (CNA), and all the CNAs are connected in a circle. This network is used to report each PE's status (busy or idle) to all the other PEs. The 16 bits are divided into 4 bits for the packet address, 4 bits for each PE's status, and 8 bits are free for other information.

4. EXPERIMENTS

We previously collected data for the N-queen problem, and found the KABU-WAKE method to perform very well. We are now comparing the performance data for the DCG parser with that for the N-queen problem. We parsed three sentences with a DCG parser using the KABU-WAKE method.

The three sentences and their characteristics are below. Because of ambiguity in the original Japanese-language sentences, the following translated English-language sentences do not have exactly the same meanings.

Sentence 1:

"wakai otoko wa kouen de akai boushi no onna no ko ga hon wo yomu nowo mita".

MEANING: A young man saw a girl with a red hat reading a book at a park.

This sentence takes 18 seconds for one PE to process, and it has 10 parsing trees.

Sentence 2:

"aoi fuku no wakai otoko wa kouen no ki no benchi de akai boushi no onna no ko ga atarashii ryouri no hon wo yomu nowo mita".

MEANING: A man wearing blue clothes saw a girl with a red hat reading a new cookbook on a wood bench at a park.

This sentence takes 167 seconds for one PE to process, and it has 80 parsing trees.

Sentence 3:

"akai fuku no wakai otoko wa watashi no machi no kouen no ki no benchi de akai boushi no onna no ko ga atarashii ryouri no hon wo yomu nowo mita".

MEANING: A young man wearing red clothes saw a girl with a red hat reading a new cookbook on a wood bench at a park in my town.

The sentence takes 2141 seconds for one PE to process, and it has 560 parsing trees.

Since DCG parsing problems have no natural parameter of size, such as N in the N-queen problem, We discuss the size in terms of the cpu time to solve the problem by a single processor.

We obtained execution times for different numbers of PEs, from 1 to 12. We also checked the number of subtask transfers, and dynamic status changes during execution.

5. EVALUATION

1)Performance improvement ratio

Fig 2 shows the performance improvement ratio due to parallel processing for some DCG parsing and N-queen problems. For Sentence 3 with 12 PEs, the performance improvement ratio is 91 % of the ideal performance.

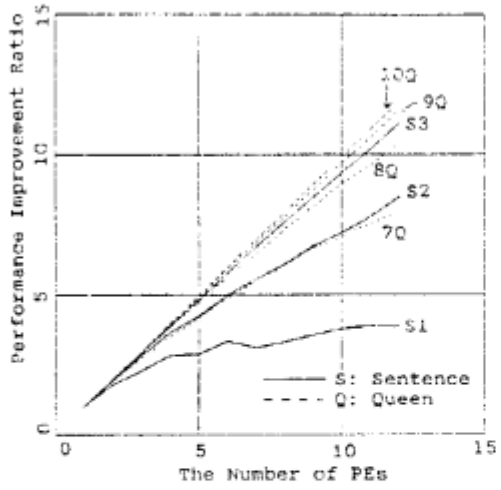

Fig. 2 Performance Improvement Ratio due to Parallel Inference

2)Number of subtask transfers.

Fig 3 shows the number of subtask transfers versus the number of PEs for the DCG parser. The relationship is approximately linear for all the DCG and N-queen problems.

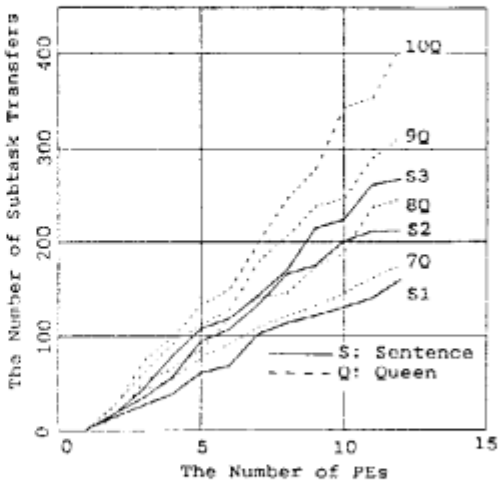

Fig. 3 The Number of Subtask Transfers vs. The Number of PEs

3)Dynamic change of PE statuses.

Fig 4 shows the dynamic status changes of PEs during parsing of Sentence 3.

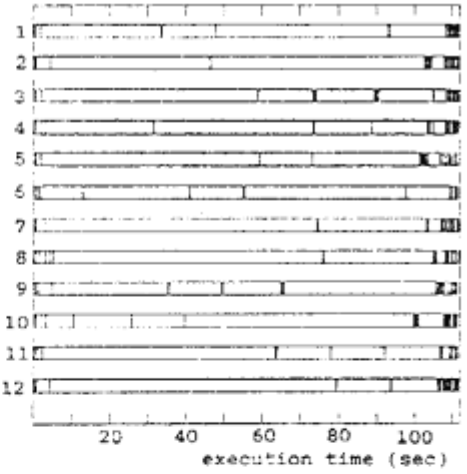

Fig. 4 Dynamic Status Changes of PEs during Parsing of Sentence 3 with 12 PEs

At the beginning of task execution, there are not many large subtasks near the root of the search tree, so subtasks are small and their execution times are short.

As task execution proceeds, larger subtasks are produced and all the PEs are busy processing large branches, so there is no communication in this stage.

As task execution ends, the PEs are either idle or are working on small subtasks; all the subtasks held by busy PEs are small or not splittable.

From these figures, we can draw the following conclusions about the KABU-WAKE method.

Larger tasks are executed more efficiently.

With 12 PEs, Sentence 3 is 11.1 times faster than with a single PE, but sentence 1 is only 3.9 times faster.

This means that sentence 3 is executed more efficiently by the KABU-WAKE method than Sentence 1 is.

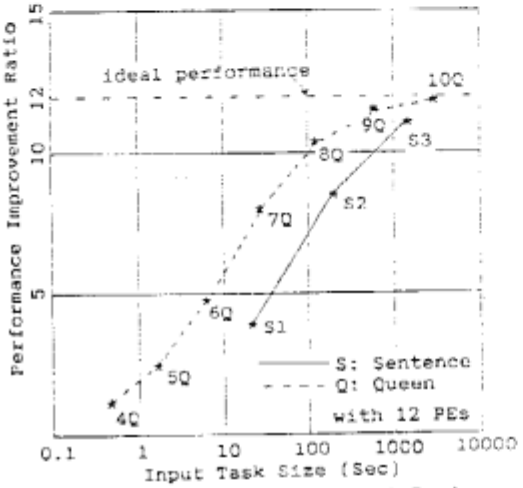

Fig. 5 Performance Improvement Ratio vs. Input Task Size

Fig 5 shows performance improvement ratio versus task size. Larger tasks such as the 10-queen problem or Sentence 3, are executed more efficiently, but even for an N-queen problem and DCG problem with the same task size, the N-queen problem is processed more efficiently. For Sentence 2 and the 7-queen problem, the performance improvement ratios are nearly the same, even though the task size of Sentence 2 is about 7 times larger than that of the 7-queen problem. This is because, the number of transferred data bytes during execution is different for DCG problems and the N-queen problem. For the N-queen problem, the number of transferred data bytes is about 1/4 to 1/5 that for the DCG problems.

Fig 6 shows a schematic time chart of task division, and subtask transfer and reception. The overhead for communication between PEs for one subtask transfer depends mainly on the amount of data transferred.

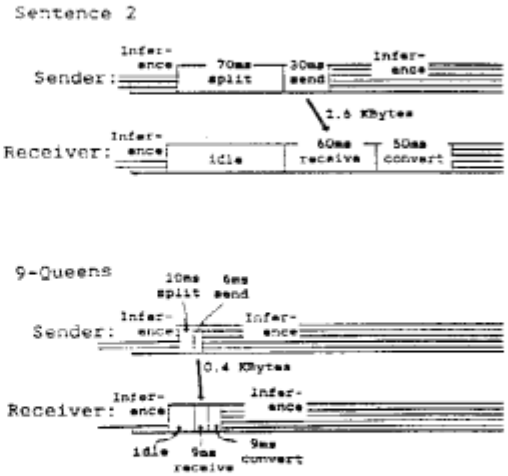

Fig 6. Schematic Time Chart of KABU-WAKE process

During subtask splitting and transfer to an idle PE, the sending PE must stop its own inference. The receiving PE must convert the transferred subtask into an internal form suitable for inference. This conversion takes about as same time as transferring a subtask. Currently, the data transfer rate of TASK-NETWORK is not very high, so data transfer significantly reduces system performance.

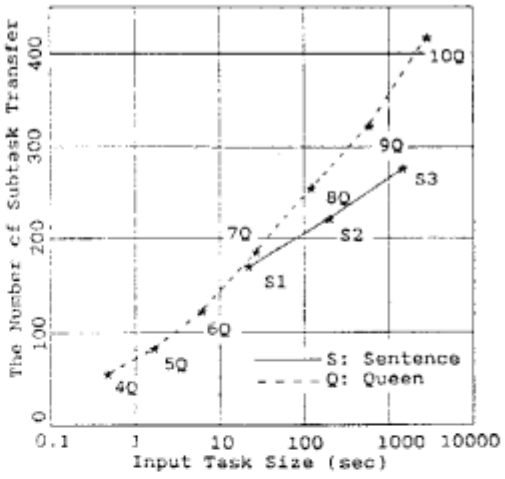

Fig. 7
Relation ship Between Input Task Sizes and the Number of Subtask Transfers

The subtask granularity increases as the task size increases.

Fig 7 shows the relationship between the input task size and the number of subtask transfers. We can see the

number of transfers is not proportional to the task size, but to the logarithm of the task size.

It means that the average subtask granularity increases as the size of the initial task increases. Parallel inference efficiency is the ratio of inference execution time to overhead for parallel execution. Thus, the KABU-WAKE method is suited to large tasks because the amount of communication between PEs is kept relatively low, and the efficiency increases.

6.CONCLUSION

We have experimented and evaluated the KABU-WAKE method. We have tested a DCG parser on our experimental machine as a practical application.

We have found that the KABU-WAKE method can solve larger tasks efficiently because the subtask granularity becomes large as the input task becomes large and the number of subtask transfers is kept relatively low even if the input task becomes large.

So the KABU-WAKE method is suitable for a large problems which has a lot of parallelism.

In the future, we will work on the followings:

- Speeding up PEs by using compilation techniques
- Reducing the amount of overhead for subtask splitting
- Reducing the amount of overhead for transferred data conversion

7.Acknowledgement

The authors would like to thank Manager Tanahashi and General Manager Sato for their unfailing encouragement in our research, and Managing Director Yamada for giving us the opportunity to conduct this research.

References

[Itashiki 85] Itashiki, et al., "PARALLEL INFERENCE PROCESSING SYSTEM -- IMPROVED CLAUSE UNIT PROCESSING METHOD", 30th meeting of Information Processing Society, 7c-7, March 1985. Japanese.

[Kumon 85a] Kumon, et al., "PARALLEL INFERENCE PROCESSING SYSTEM -- IMPROVED CLAUSE UNIT PROCESSING METHOD", 30th meeting of Information Processing Society, 7c-8, March 1985. Japanese.

[Kumon 85b] Kumon, et al., "EVALUATION OF KABU-WAKE PROCESSING", 31th meeting of Information Processing Society, 2c-5, Oct. 1985. Japanese.

[Sohma 85] Sohma, et al., "A NEW PARALLEL INFERENCE MECHANISM BASED ON SEQUENTIAL PROCESSING", IFIP TC-10 Working Conference On Fifth Generation Computer Architecture, July, 1985.