TR-294

A Simulation Study of A Knowledge Base Machine Architecture

by
H. Sakai and S. Shibayama (Toshiba)

August, 1987

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 SIMULATION STUDY OF A KNOWLEDGE BASE MACHINE ARCHITECTURE

Hiroshi SAKAI, Shigeki SHIBAYAMA
Toshiba R & D Center, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 210, Japan

Hidetoshi MONOI, Yukihiro MORITA, Hidenori ITOH
ICOT Research Center, Mita-Kokusai-Build., Mita,Minato-ku, 108, Tokyo, Japan

ABSTRACT

As part of Japan's Fifth Generation Computer Project for the development of a knowledge base machine (KBM), an advanced database machine incorporating a new data model and hardware architecture is proposed. For the data model, an extension of the ordinary relational model is used. In this model, a term which may contain variables is allowed as an attribute value and operations over relations concerning unifiability between terms are defined. For the hardware architecture, unification engines handle unification operations over relations in parallel, and a multiport page-memory reduces the bottleneck between primary and secondary storage.

An estimation of the system performance is made by simulating the system. A precise model of the hardware architecture and control strategies were implemented. A control strategy in which a join operation is divided into small ones according to the sizes of the relations and the number of engines is proposed and found to be useful for this hardware architecture in that it saves processing time and memory usage.

FRAMEWORK OF OUR KBM RESEARCH PROJECT

Research Objectives

We have conducted research on constructing a knowledge base machine (KBM) within Japan's Fifth Generation Computer Project. The term KBM means a database machine with advanced functions for knowledge. In an intermediate project lasting four years, we aim to develop a prototype KBM which provides a common knowledge base for inference machines.

Design Philosophy of the KBM Functions

We developed the relational database machine, Delta, as the first step of our research [Kakuta 85], [Sakai 86]. Through the experience, we have arrived at the following conclusion: while a database system is needed, an ordinary relational model is not sufficient; data structures such as list and term which appear in Prolog programs cannot be represented efficiently. So we adopted an extension of the ordinary relational model [Morita 86], [Yokota 86]. In this model, a term having variables is allowed as an attribute

value. The scope of a variable is within the tuple, as in a Horn clause. Operations over relations concerning unifiability and/or generality of terms are defined. Figure 1 shows sample relations and operation. The operation is called a unification join (U-join). This model allows not only nested relations but also a kind of inference. A similar model and a knowledge base mechanism different from ours are reported by Ohmori [Ohmori 86].

R	R_a	R_b
	X	f(X, a)
	h(X, X)	g(a, Y)
	f(a, b)	g(b, c)

S	S_a	S_b
	g(X, b)	f(X, b)
	g(X, c)	g(X, d)

$$T \leftarrow R \underset{R_b}{\bowtie} \underset{S_a}{} S$$

T	R_a'	R_b'	S_a'	S_b'
	h(X, X)	g(a, b)	g(a, b)	f(a, b)
	h(X, X)	g(a, c)	g(a, c)	g(a, d)
	f(a, b)	g(b, c)	g(b, c)	g(b, d)

Figure 1 Example of relations and an operation

Design Philosophy of the System Architecture

Through the experience with Delta, we found that it is useful (1) to use multiple special-purpose processors to cope with heavy operations like U-join in parallel, and (2) to use a large disk cache to reduce the bottleneck between primary and secondary storage. To achieve the first goal, a unification engine (UE) was designed [Morita 86]. To achieve the second, a multiport page-memory (MPPM) [Tanaka 84] was adopted. This is a kind of multiport memory without any contention although its accessible unit is limited to a page, not a word.

Figure 2 gives an overview of the system architecture. It contains disk devices which store permanent relations, an MPPM which behaves as a disk cache shared by the disk devices and UEs, UEs which process operations on relations within the MPPM, and a processor which controls the whole system.

Overview of the Simulation Study

Before undertaking the actual implementation, we simulated our KBM architecture to estimate the system performance. A precise model of the UE was required since it significantly affects the execution time, so we made a simulation study on design alternatives of the UE first [Morita 87]. Like other kinds of parallel processing, the question of control was also important. We adopted alternative control strategies and compared them in the simulation study.

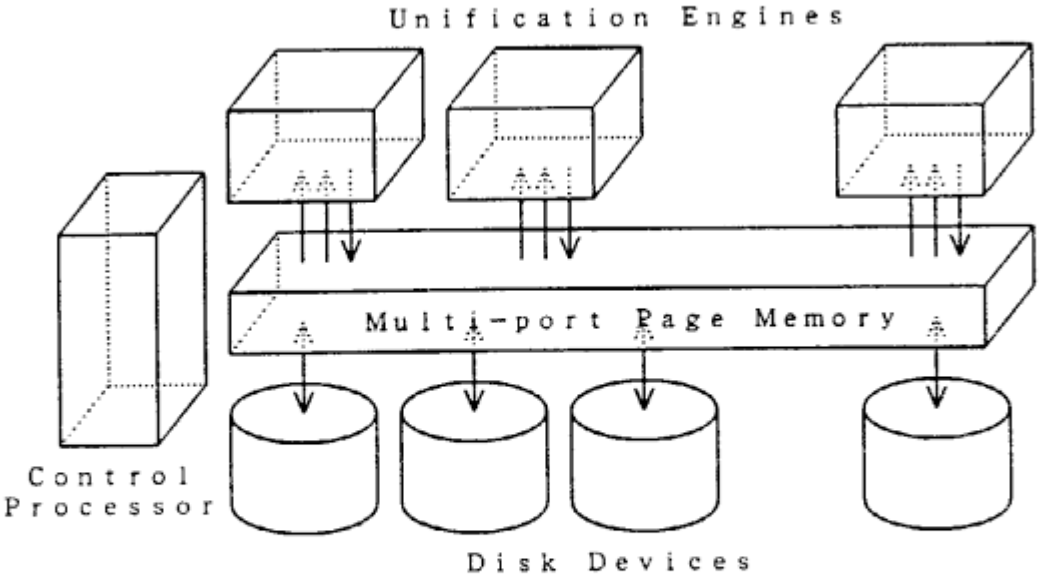

Figure 2 System configuration of a KBM

KBM MODEL

System Architecture

In the simulation study, we adopted the system architecture illustrated in figure 2 except for disk devices. Therefore, we assumed that relevant relations are always staged in the MPPM; it is the best case of system performance.

The MPPM is a shared memory with multiple ports. It allows constant data transfer rate for each port. The rate is assumed to be 20 Mbytes/sec so that it is equal to the processing speed of the UE. The MPPM stores relations on a page basis. The term "page" has two meanings. One is a unit of size of data access which causes a time delay. It depends on the hardware implementation and is called the track size. It is assumed to be 512bytes. The other is a unit of size of storage that must be a multiple of the track size. It is determined by the management software of the system. We compare the cases of 0.5, 1, 2, 4, 8, 16, 32, 64 Kbyte pages in the simulation study.

The UE can process a U-join operation between pages of a relation within the MPPM and pages of another, and output resultant tuples into the MPPM in a page scheme. Each UE is connected to the MPPM through three ports, two of which are used to input tuples of two relations simultaneously and the other to output the resultant tuples. Figure 3 shows its configuration. Here, the pair generator, receiving arranged sequences of input tuples from the sorters, produces pairs of tuples which is possibly unifiable. The unification unit checks the unifiability of each pair and outputs the resulting tuples. The

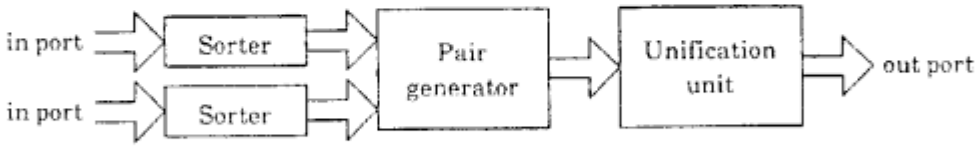

Figure 3 Internal configuration of UE

amount of pages that can be processed by a UE at one time is limited to the size of its buffer memory. The number of tuples that can be processed is also limited by the sorter.

A tuple is represented as a sequence of four-byte words, each of which stands for a component of a term (atom, functor, variable). Every unit of a UE is designed to process the word within a certain period, assumed to be 200 nsec. At the beginning of a period, a unit receives a word (if any) from the adjacent unit, and sends the resultant word at the end of that period.

The control processor manages all the resources in the system. The overhead time is ignored. The important specifications are summarized below.

MPPM	Track Size	0.5 Kbytes (access delay 0.0256 msec)
	Logical Page Size	0.5, 1, 2, 4, 8, 16, 32, 64 Kbytes
	Data Transfer Rate	20 Mbyte/sec for each port
UE	Number of UEs	1 to 32
	Processing Speed	200 nsec per four-byte word
	Number of I/O Ports	3 (2 for input and 1 for output)
	Buffer Size	4, 8, 16, 32, 64 Kbytes
Control Processor	control overhead	None

Input Resolution in the System

In the simulation study, we focus on input resolution for the following reasons.

(a) It is suitable for studying the system behavior since input resolution is realized by the repetition of U-joins.

(b) It is a typical operation that an ordinary relational database is not able to handle.

Representation of Definite Clauses and a Goal Clause. Definite clauses are stored in a relation with two attributes. One stores the head of a definite clause and the other stores its body. Both attributes are stored in list form, with the same variable attached as the last component. The relation is called a permanent relation (PR)

A goal clause is stored in another relation with two attributes. One stores the expected form of the goal and the other stores the original goal clause in list form, with '*nil*' attached as the last component. The relation with a goal clause is called a temporary relation (TR). Figure 4 illustrates an example of a PR and TRs.

PR	Head	Body
	[ancestor(X, Y)|L]	[father(X, Z),ancestor(Z,Y))|L]

TR_0	Answer	Resolvent
	X	[ancestor(tom, X)]

TR_1	Answer	Resolvent
	X	[father(tom, Z), ancestor(Z, X)]

Figure 4 Knowledge representation for input resolution

Realizing Input Resolution. In the system, a U-join operation between the first attribute of the PR and the second attribute of the TR makes one step of input resolution and generates tuples which contain resolvents in the second attribute.

If TR_0 is the relation which contains the original goal clause, then a U-join between the PR and TR_0 generates a relation with new resolvents, TR_1. In general, a U-join between the PR and TR_n generates a relation, TR_{n+1}.

If the value of the second attribute of a tuple in TR_n is equal to *'nil'*, this indicates that the original goal clause becomes a sequence of ground clauses and that the value of the first attribute is the answer. A restrict operation is used to pick up these tuples. The input resolution ends when the U-join operation generates no more tuples.

Basic Considerations for Parallel Processing of U-join

Let us consider a U-join operation between relations R and S. If R_1 to R_m are the partitions of relation R, and S_1 to S_n are those of relation S, then the equation (1) holds. A U-join operation can be executed in parallel by assigning the indivisual $R_i \bowtie S_j$ to UEs.

$$R \bowtie S = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} R_i \bowtie S_j \tag{1}$$

Let F(r, s, t) be the execution time for a UE to execute a U-join operation between a part of relation R and that of relation S. Here, parameters r and s are the size of the portions and parameter t is the size of the result. Then as a first approximation, F(r, s, t) can be given by formula (2).

$$F(r, s, t) = a^*(r+s) + b^*t + c^*r^*s \tag{2}$$

Here, parameters a and b are almost independent from the characteristics of the relations. However, parameter c depends on how many tuple pairs generated by the pair generator of the UE are actually unifiable. Parameter c is called the antiselectivity ratio.

Basic inequation (3) holds, which shows that partitioning a U-join operation takes time unless the number of UEs increases.

$$F(r_1+r_2, s, t_1+t_2) < F(r_1, s, t_1) + F(r_2, s, t_2) \tag{3}$$

Now let us consider the execution time of the U-join operation between R and S relations in parallel. This is given by E in the inequations in (4), where

r_i = size of R_i — r = summation of r_i
s_j = size of S_j — s = summation of si
t_{ij} = size of the result of R_i S_j — t = summation of t_{ij}
k = number of UEs

$$E \geqq \sum_{i=1}^{m} \sum_{j=1}^{n} F(r_i, s, t_{ij}) / k \tag{4a}$$

The equation holds when all UEs work continuously.

$$= (a^*(m^*r + n^*s) + b^*t + c^*r^*s) / k \tag{4b}$$

$$\geqq (2a\sqrt{m^*n^*r^*s} + b^*t + c^*r^*s) / k \tag{4c}$$

The equation holds when $m^*r = n^*s$.

$$\geq 2a\sqrt{r^*s} / \sqrt{k} + (b^*t + c^*r^*s) / k \qquad (4d)$$

The equation holds when $m^*n = k$.

For inequation (4d), note that the product of m and n must be greater than k in order to use all the UEs. These inequations tell us the following facts.

(a) To minimize the execution time, m should be $\sqrt{k^*s / r}$ and n be $\sqrt{k^*r / s}$.
(b) To make all UEs work continuously, it seems reasonable to make all r_i equal and all s_j equal.
(c) (a) and (b) suggest that all r_i and s_j should be made equal to $\sqrt{r^*s / k}$.

Fact (c) suggests that relations should be partitioned according to their size and the number of UEs. Therefore, we adopt what we call the MP (Multiple Pages at a time) method. In the MP method, the page size is set relatively small and a U-join request between multiple pages of a relation and multiple pages of another relation is assigned to a UE. In the simulation study, we compare it with what we call the SP (Single Page at a time) method. Following the SP method, each join request is limited to between single page of a relation and single page of another. The simulation study found that the MP method is almost always better than the SP method.

Control Model for Parallel Execution of Input Resolution

Parallel Execution of an Input Resolution. During input resolution, the control processor of the system does the following jobs many times: generating U-join requests between parts of the PR and TR, assigning these requests to UEs, attaching the resulting pages to the TR, releasing a page of the TR when all the requests concerning it are completed. Therefore, the pages within the TR vary as input resolution progresses.

The control model based on the SP method, as illustrated in figure 5, does not contain significant alternative choices because the generation of U-join requests does not contain any time-dependent factors. On the other hand, the control model based on the MP method contains alternatives since the request generation depends on two time-dependent factors: the size of the TR and the number of UEs. In the control model illustrated in figure 6, output pages are first stored in a page pool. The control processor is able to generate U-join requests between the pages within the pool and PR whenever it wants to. The requests are stored in the request queue before being assigned to a UE. The remainder of this section introduces two parameters which describe the freedom within the MP method.

Partitioning Factor. In the course of input resolution, the activation of UEs is not necessarily synchronized. Therefore, in generating U-join requests between the PR and TR, there is room to consider how many UEs are available at the moment, in other words, how many requests should be generated.

If *UEfree* is the number of currently free UEs and *UEtotal* is the number of UEs in the system, it seems reasonable to choose the number of requests from *UEfree* to *UEtotal*. Therefore, a parameter '*p*' is introduced so that the number of requests is calculated by the formula, $\max(UEfree, p \times UEtotal)$. This parameter is called the partitioning factor.

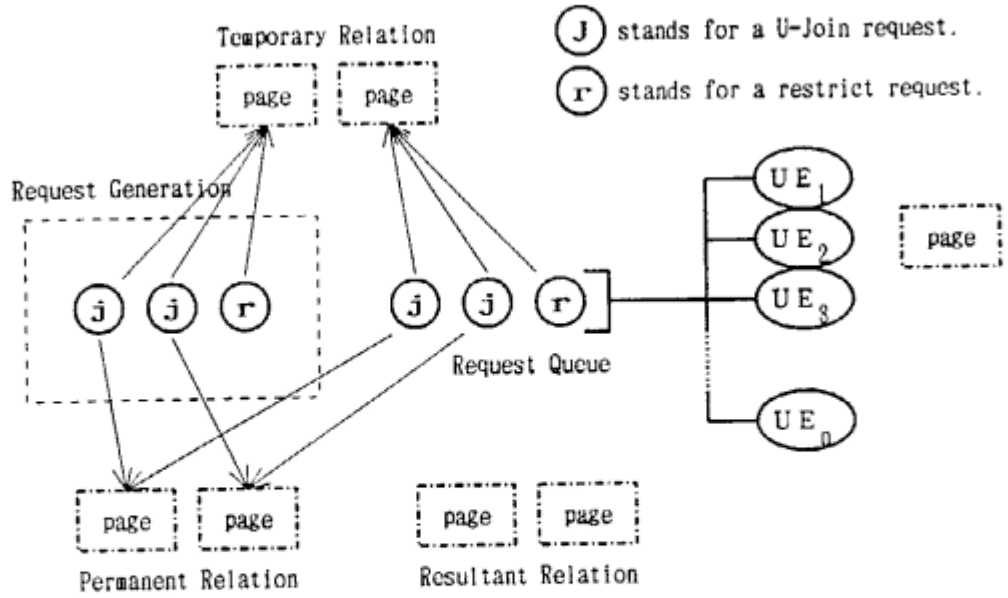

Figure 5 Control model based on the SP method

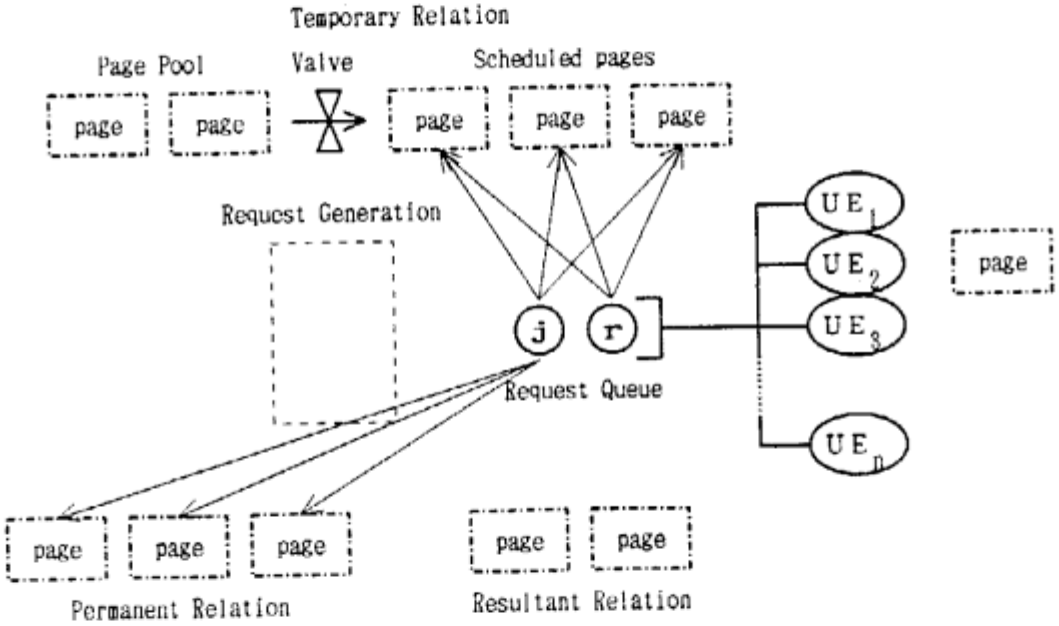

Figure 6 Control model based on the MP method

Waiting Ratio. If the system postpones generating U-join requests, the page pool may contain more pages. This leads to higher performance unless the working-ratio of UEs decreases, because the accumulated contents of the TR during input resolution do not vary whatever control strategy is adopted. It is possibly better that the system postpones generating requests until a certain number of UEs become free. In the simulation study, a parameter '*wt*' is introduced so that the system postpones the request generation until *UEfree* becomes greater than or equal to *wt*UEtotal*.This parameter is called the waiting ratio.

SIMULATION METHODOLOGY

Simulator Overview

The simulator is an ordinary single task program which runs on a conventional computer. It has a module which simulates the activity of a UE at the register transfer level and a scheduling module which reflects parallel processing based on the control model described above. The simulator is executed every time the parameters within the KBM model are modified. This gives reliable results except that the KBM model does not account for the overhead time due to the control processor.

Measures for Evaluation

To compare alternatives within the KBM model, the measures listed below are selected:

(a) Execution Time (*Et*)

This is the time elapsed in executing an input resolution in parallel.

(b) Page Loading Factor (*Ld*)

This is an average over output pages which shows how much the pages are loaded with resultant tuples.

(c) Performance Stability (*Ps*)

This shows to what extent the system exhibits stable performance over variance of the tuple size or other system parameters.

Sample Problems Used in the Simulation Study

Ancestor Problem. Finding all the ancestors of a certain person in a collection of parent-child relationships is a typical problem for deductive databases. A computer-generated family tree based on a simple model of human life cycle is used. The PR consists of 1800 relationships among persons and several related rules. Since the pair generator does not generate any non-unifiable pair, the antiselectivity ratio is 0.

Eight-queen Problem. The eight-queen problem is also frequently used to evaluate the performance of inference machines. In the simulation study, we use not an ordinary set of rules but a set of nine complicated rules, each of which has twenty-five variables and does not require any arithmetic operations.Since each tuple has a similar form, the pair generator generates almost all tuple combinations. Less than one seventh of these are exactly unifiable. Therefore, the antiselectivity ratio is large.

The differences between the sample problems are listed in figure 7.

	Ancestor	8-queen
Tuple Size of TR	44 - 60 bytes	188 - 220 bytes
PR size	64 Kbytes	2 Kbytes
TR size (accumulated)	36 Kbytes	386 Kbytes
No. of inference steps	16	10
Antiselectivity ratio	0	Large

Figure 7 Comparison of the sample problem

RESULTS AND DISCUSSION

SP method versus MP method

Here, the SP and MP method are compared. For the parameters of the MP method, the partitioning factor is 1 and the waiting-ratio is 1 / *UEtotal*.

Ancestor Problem. Figure 8 shows the relationship between *UEtotal* and the execution time. For the SP method it shows that:

(s1) If the page size is small, the execution time becomes large when *UEtotal* is small. This is because too many small U-join requests are generated.

(s2) If the page size is large, the execution time does not decrease when *UEtotal* increases. This is because the number of requests generated during the input resolution remains so small that the working-ratio of UEs becomes low.

For the MP method, it shows that:

(m1) The page size does not affect the execution time much.

As a whole, the figure shows that the MP method is superior to the SP method for measures (*Et*) and (*Ps*).

Figure 9 shows the relationship between *UEtotal* and the page loading factor. For the SP method, it shows that:

(s3) The page loading factor does not depend on the number of UEs.

For the MP method, it shows that:

(m3) The page loading factor becomes lower when the number of UEs increases, because each U-join request, partitioned into smaller pieces, generates smaller number of resultant tuples.

(m4) The page loading factor at a certain number of UEs decreases, when the page size grows, because a small page size means that the unit of space for holding resultant tuples is also small.

As a whole, the figure shows that the MP method is superior to the SP method for measures (*Ld*) and (*Ps*).

Eight-queen Problem. Figure 10 and 11 show the results. They also show that the MP method is superior to the SP method, but the degree of superiority is small.

(s5) Unlike the ancestor problem, even when the number of UEs is small, the execution time in the SP method does not increase much. This is chiefly because both sides of inequation (3) become almost the same value since the antiselectivity ratio is large.

(s6) The page loading factor in the SP method becomes larger than in the ancestor problem, because the eight-queen problem generates a large number of resultant tuples compared with the ancestor problem.

For the MP method, figure 11 shows that:

(m6) Unlike the ancestor problem, a small page size does not always mean a high page loading factor. This is because the tuple size is close to the page size; i.e., 512 byte page can hold only two tuples and about 20 percent of the page size remains unused.

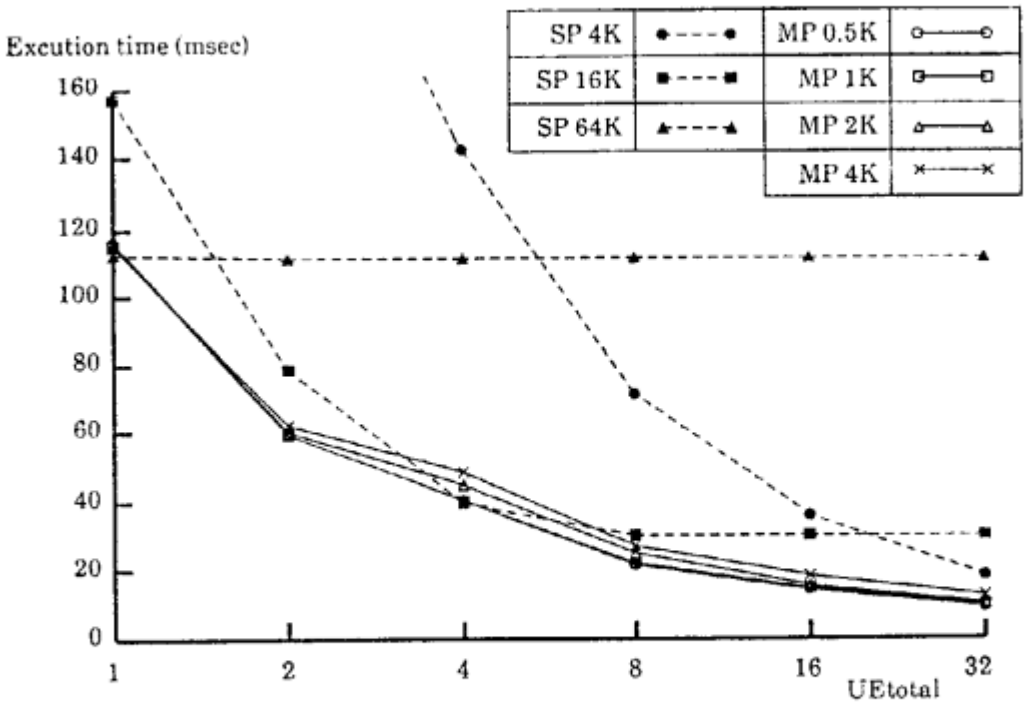

Figure 8 Relationship between *UEtotal* and execution time (Ancestor problem)

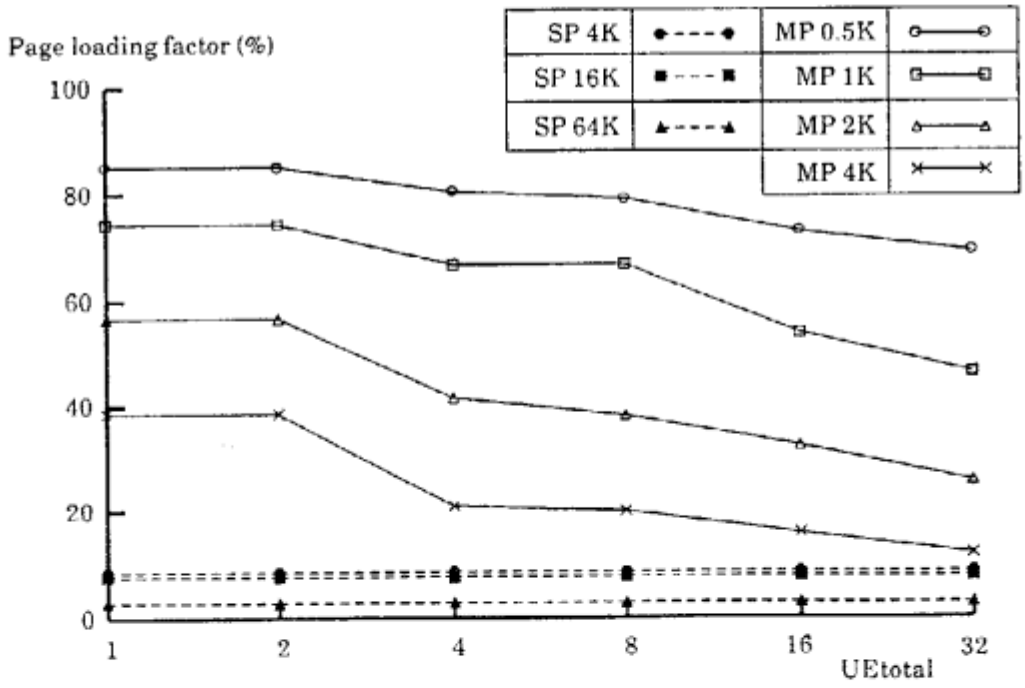

Figure 9 Relationship between *UEtotal* and page loading factor (Ancestor problem)

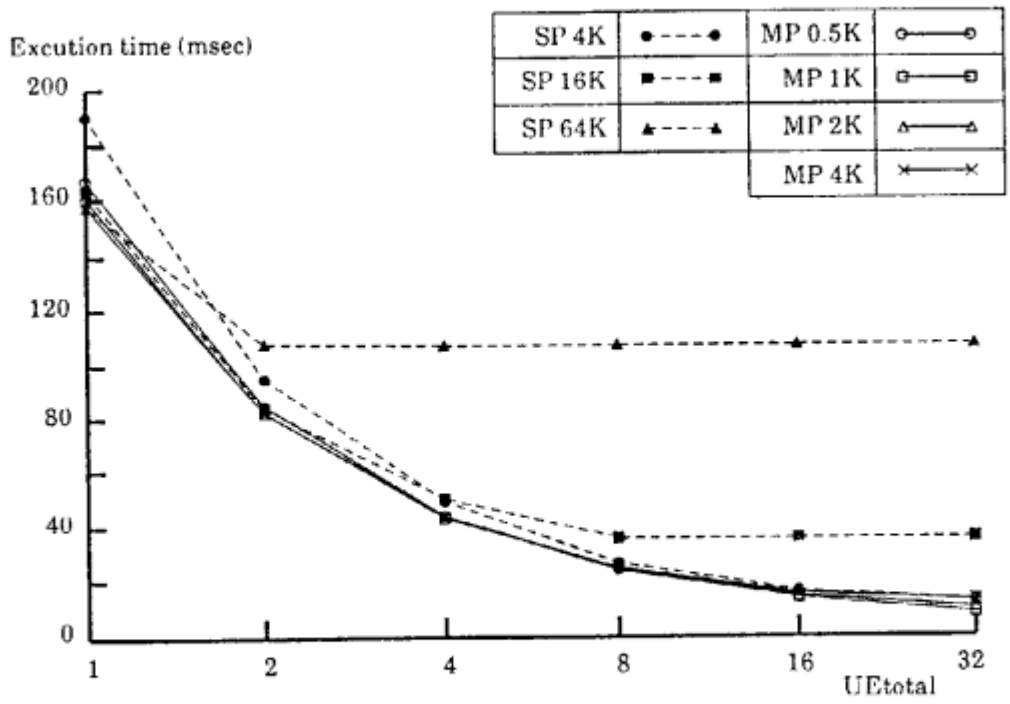

Figure 10 Relationship between *UEtotal* and execution time (8-queen problem)

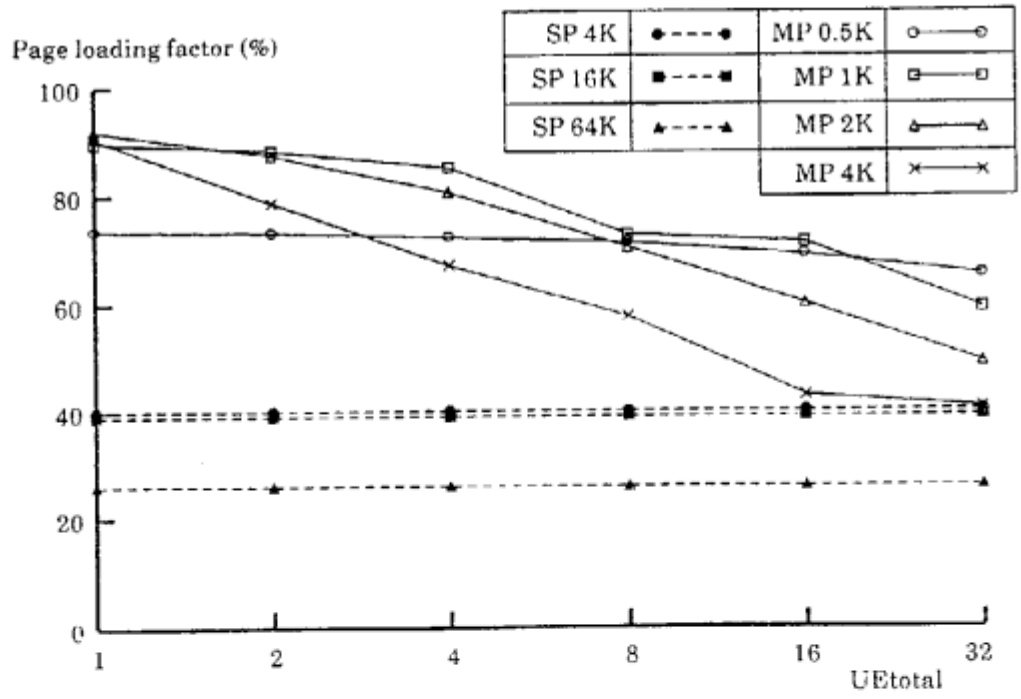

Figure 11 Relationship between *UEtotal* and page loading factor (8-queen problem)

Details of the MP Method

Effects of the Partitioning Factor. The partitioning factor determines the number of U-join requests generated at one time. If the parameter is 0, the system tries to generate as many requests as *UEfree*. If it is 1, the system tries to generate as many requests as *UEtotal*.

When the parameter is 0, the system performance is not stable. This is because it sometimes occurs that only a small number of UEs are processing U-join requests for a long time and the other UEs are just idling.

Further simulation study found that the system shows good and stable performance for both problems when the parameter is between 0.8 and 1.0.

Effects of the Waiting-ratio. The waiting-ratio is used to trigger the generation of new U-join requests. This section discusses the effects of this parameter when the partitioning factor is equal to 1.

If the waiting-ratio is low, the system generates new U-join requests as soon as a small number of UEs become free. Therefore, the working-ratio of UEs becomes larger than that when the waiting-ratio is high. However, the system has to generate U-join requests more times since each request generation consumes a smaller part of the TR.

For the eight-queen problem, the performance when the waiting-ratio is 1/*UEtotal* is 10% to 20% better than that when the working-ratio is 1. This is because the PR size is very small and the antiselectivity ratio is large. Therefore, increase in the frequency of request generation is of little importance.

For the ancestor problem, if *UEtotal* is less than a certain threshold, the waiting-ratio should be 1. This is because the PR size is large and the antiselectivity ratio is 0. However, if the *UEtotal* exceeds the threshold, the waiting-ratio should be 1/*UEtotal*, because the high working-ratio of UEs becomes more important.

Relationship between UEtotal and System Performance. Figure 12 shows the relationship between *UEtotal* and performance of both problems. Here the partitioning factor is 1, the waiting-ratio is 1/*UEtotal*, and the page size is 0.5Kbyte. The figure shows

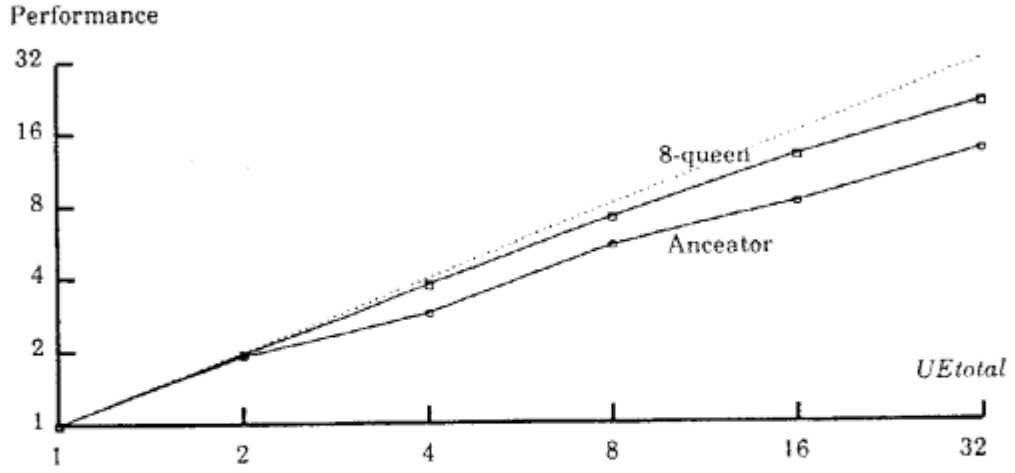

Figure 12 Relationship between UEtotal and Performance

that the performance improvement of the ancestor problem is smaller than that of the eight-queen problem.

This is because:

(1) In the ancestor problem, the low antiselectivity ratio (= 0) causes higher costs of partitioning of a U-join operation than in the eight-queen problem.
(2) In the ancestor problem, the PR size is large. Therefore, the increase in the frequency of request generation obstructs the performance improvement more than in the eight-queen problem.

Discussions on KBM Architecture

Use of MPPM. This section discusses how much of the potential data transfer capability of the MPPM is used during input resolution. Anew measure, port utilization ratio, is introduced and defined as follows:

$$\frac{\textit{the actual size of transferred data through a port}}{\textit{the execution time} * \textit{data transfer capability of a port}}$$

Figure 13 summarizes the port utilization ratio in both problems.

The figure shows that although the port utilization ratio of a certain port varies widely, the average over three ports of a UE is stable. The port utilization ratio seems rather low. However, simulation shows that the port utilization ratio becomes about 45% in the ancestor problem and about 30% in the eight-queen problem by assigning only one port to each UE. In this case, the performance will become 5% to 30% smaller, assuming that *UEtotal* is equal and that the MPPM has only one third of the ports as in the current architecture model.

Towards Processing a Large Number of Definite Clauses. Since a knowledge base machine is expected to have enough power for problems with a large number of definite clauses, it is hoped that the system performance does not depend much on the number of definite clauses. From this point of view, though such a system with the MPPM shared among processors and disk devices seems to have a potential capability, it has a problem in that the execution time is proportional to the number of definite clauses. It comes from the fact that each U-join operation causes the transfer of all the relevant definite clauses from the MPPM to the UE. We think that a clustering method on terms may reduce the amount of data transfer.

	Ancestor	8-queen
Port for PR	4 % - 18 %	16 % - 23 %
Port for TR	36 % - 45 %	1 % - 14 %
Port for output	0.7 % - 2 %	7 % - 12 %
Average	16 % - 18 %	11 % - 13 %
When only one port per a UE	36% - 49 %	25 % - 35 %

Figure 13 Port Utilization Ratio of the MPPM

CONCLUSIONS

A simulation study of a knowledge base machine architecture and its control strategies was reported. For the control strategy, the MP method was proposed and found to be superior to the SP method for both the execution time and the page loading factor.

For hardware architecture, although the potential data transfer capability of the MPPM is not fully used, it can be resolved by assigning only one port to each UE. We are currently conducting research into clustering methods on terms in order to improve the performance when the amount of definite clauses becomes large.

ACKNOWLEDGEMENT

We express thanks to Mr. Kazuhide Iwata for fruitful discussions.

REFERENCES

[Boral 82] Boral, H., et al.., Implementation of the Database Machine DIRECT, *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 6, 1982.

[Kakuta 85] Kakuta, T., et al.: The Design and Implementation of Relational Database Machine Delta, *Proceedings of the Fourth International Workshop on Database Machines*, pp.13-34, 1985.

[Morita 86] Morita, H., et al.: Retrieval by Unification Operation on a Relational Knowledge Base, *Proceedings of the 12th International Conference on VLDB*, 1986.

[Morita 87] Morita, H., et al.: Performance Evaluation of a Unification Engine for a Knowledge Base Machine, *ICOT TR-204*, 1987.

[Sakai 84] Sakai, H., et al.: Design and Implementation of the Relational Database Engine, *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp.419-427, 1986.

[Sakai 86] Sakai, H., et al.: Development of Delta as a First Step to a Knowledge Base Machine, *Database Machines Modern Trends and Applications, NATO ASI Series* F, Vol 24, 1986.

[Shibayama 85] Shibayama, S., et al.: A Knowledge Base Architecture and its Experimental Hardware, *Proceedings of IFIP TC-10 Working Conference on Fifth Generation Computer Architectures* 1985.

[Ohmori 86] Ohmori, T., et al.: An Optimization in Unifiable Pattern Search.

[Tanaka 84] Tanaka, Y.: A Multiport Page-Memory Architecture and A Multiport Disk-cache System, *New Generation Computing*, Vol 2,no.3, pp.241-260, 1984.

[Yokota 86] Yokota, H., et al.: A Model and an Architecture for a Relational Knowledge Base, *Proceedings of the International Symposium on Computer Architecture*, 1986.