ICOT Technical Report: TR-261

TR-261

A Simulation Study of a Knowledge Base Machine Architecture

by
H. Sakai, S. Shibayama (Toshiba), H. Monoi,
Y. Morita and H. Itoh

May, 1987

©1987, ICOT

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
Hidetoshi Monoi, Yukihiro Morita, Hidenori Itoh
ICOT Research Center

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 was implemented. Control strategies for parallel processing were then implemented on the hardware model and evaluated. A control strategy in which the system assigns to a unification engine a request to join multiple pages of two relations is proposed. It was found that the strategy is useful for this hardware architecture in that it optimizes the processing time and memory usage.

Moreover, detailed records of system activities are obtained, which are considered useful for the full-scale implementation of such a system.

1 Framework of Our KBM Research Project

1.1 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.

1.2 Design Philosophy of the KBM Functions

We developed the relational database machine, Delta, as the first step of our research [Kakuta 85], [Sakai 86] and evaluated it. We also made an experiment in which Delta and an inference machine, PSI, communicated through a local area network.

Through the experience, we have arrived at the following conclusion: while a database system is needed, an ordinary relational model is not sufficient; i.e., knowledge engineers prefer more powerful models. Therefore, we decided to adopt 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 also defined. Figure 1 shows sample relations and a sample operation. The operation is called a unification join (U-join). Following this model, one can not only represent complex situations but also realize a kind of inference.

A similar model and a knowledge base mechanism different from ours are reported by Ohmori [Ohmori 86].

1.3 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. The MPPM may be regarded as an alternative to a system bus and shared memory.

1.4 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] and adopted the best model in this simulation study. 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.

This paper shows the result of the simulation of our KBM architecture. Section 2 describes the system architecture and control strategies. Section 3 describes the simulation study. Section 4 gives the results.

2 KBM model

2.1 System Architecture

In the simulation study, we adopted the system architecture illustrated in figure 2 except for disk devices. Therefore, we assumed that all the required and newly generated 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 512 bytes. 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 several pages of a relation within the MPPM and several 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. The main components are two pipeline merge-sorters, a pair generator and a unification unit. The pair generator, receiving arranged sequences of input tuples from the

merge-sorters, produces pairs of tuples which may be unifiable. The unification unit checks the unifiability of each pair and outputs the resulting tuples.

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 from one of the adjacent units (if any), and sends the resultant word at the end of that period.

The 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 number of stages of the pipeline merge-sorter. In the simulation study, however, the latter limitation always follows from the former.

The control processor manages all the resources in the system. The overhead time of the control processor is ignored.

The important specifications are summarized below.

(1) 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

(2) 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

(3) Control Processor

control overhead	: None

2.2 Input Resolution in the System

Although the data model of the system can be applied to various kinds of knowledge information processing, we concentrate on input resolution in the simulation study 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 uses all the functions of the unification engine.
(c) It is a typical operation that an ordinary relational database is not able to handle.

2.2.1 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.

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.

A relation with definite clauses is called a permanent relation (PR) and a relation with a goal clause is called a temporary relation (TR). Figure 4(a) illustrates an example of a PR and TR.

2.2.2 Realizing Input Resolution

In the system, input resolution is realized by repeating the U-join operation. 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. Figure 4(b) shows an example.

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_{n1} 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 to be returned. A restrict operation is used to pick up these tuples.

The input resolution ends when the U-join operation between the PR and TR_n generates no tuples.

Note that the above method, like an ordinary Prolog system, has the disadvantage that it is possible to get into an infinite loop if the definite clauses contain a tautology. However, that can be prevented by checking whether the newly created TR_{n+1} is contained in the union of TR_0, TR_1 and TR_n. Since definite clauses without a tautology were used in the simulation study, the above method was adopted.

2.3 Basic Considerations for Parallel Processing of U-join

This section gives some basic considerations about parallel processing of a U-join operation in order to explain the control strategy in the system.

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 following equation holds:

$$R \overset{\diamond}{\bowtie} S = \sum_{i=1}^{m} \sum_{j=1}^{n} R_i \overset{\diamond}{\bowtie} S_j \qquad (1)$$

Therefore, a U-join operation can be executed in parallel by assigning all the $R_i \overset{\diamond}{\bowtie} S_j$ operations to each UE.

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).

$$F(r,s,t) = a*(r+s) + b*t + c*r*s \qquad (2)$$

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.

$$F(r_1+r_2,s,t_1+t_2) < F(r_1,s,t_1) + F(r_2,s,t_2) \qquad (3)$$

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
s_j = size of S_j
t_{ij} = size of the result of the U-join between R_i and S_j
k = number of UEs

r = summation of r_i
s = summation of s_j
t = summation of t_{ij}

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

The equation holds when every $F(r_i,s_j,t_{ij})$ becomes the same.

$$= (a*(n*r+m*s) + b*t + c*r*s)/k \qquad (4b)$$

$$>= (2a*\sqrt{m*n*r*s} + b*t + c*r*s)/k \qquad (4c)$$

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

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

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 $\sqrt{k*r/s}$ and n be $\sqrt{k*s/r}$.

(b) Since the exact execution time of a U-join operation can not be estimated before its execution, 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 $\sqrt{r*s/k}$.

(d) The execution time is not inversely proportional to k because of the first term in the formula (4d).

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.

However, there are database machines which hold relations on a page basis and assign to an engine a join request between one page of a relation and one page of another. We call this the SP (Single Page at a time) method. section 4 will show that the MP method is almost always better than the SP method.

2.4 Control Model for Parallel Execution of Input Resolution

2.4.1 Management of Temporary Relations

There are two alternatives concerning the management of multiple versions of temporary relations (TR_n). One is to keep them separately according to subscript n, and the other is to ignore the subscript and treat them as one relation. The first has the advantage of controlling the depth of inference, while in the second, the performance improves because U-join operations can be made over two large relations (c.f. inequation (3)). In the simulation study, we aim at better performance and select the second alternative.

2.4.2 Parallel Execution of an Input Resolution

Section 2.3 discussed the parallel processing of a U-join operation in a static environment. This section discusses the parallel processing of input resolution, involving time-dependent factors.

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 constituent pages of the TR vary as input resolution progresses.

The control model of 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 of the MP method contains alternatives since the request generation depends on two time-dependent factors: the size of 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.

2.4.3 Number of U-join Requests to be Generated

In the course of input resolution, the activation of UEs is not completely synchronized; they are not necessarily invoked at the same time and they do not stop at the same time. 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 to be generated from UEfree to UEtotal. Therefore, a parameter 'p' is introduced so that the number of requests is calculated by the formula, max(UEfree, p*UEtotal). This parameter is called the partitioning factor.

2.4.4 Timing of Request Generation

In the MP method, the timing of generating U-join requests also affects the execution time. If the system postpones generating new U-join requests between the PR and the contents in the page pool, the page pool may have more pages when new requests are generated. This leads to higher performance unless the working-ratio of UEs decreases. Note that the accumulated contents of the TR in the course of input resolution do not vary whatever control strategy is adopted. Therefore, to achieve high performance, the system must not generate new requests until the

requests generated before and stored in the request queue are exhausted. 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 all the following conditions hold:

(1) UEfree becomes greater than or equal to wt*UEtotal.
(2) There is no request in the request queue.
(3) There is at least one page in the page pool.

This parameter is called the waiting-ratio.

3 Simulation Methodology

3.1 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 by UEs based on the control model described in 2.4.

The simulator is executed every time the parameters within the KBM model are modified, since the order of operation terminations of UEs significantly affects the generation of subsequent requests and their assignments to UEs. This gives reliable results except that the KBM model does not account for overhead time due to the control processor.

3.2 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 the pages are loaded with resultant tuples. There is an alternative measure: the maximum number of pages used during input resolution. However, this number depends on how pages are allocated to UEs. That is, in the system where a UE is allocated pages for output at every invocation, the higher the number of UEs, the more pages the system consumes even if the control model does not vary.

(c) Performance Stability (Ps)
This shows to what extent the system exhibits stable performance over variance of the tuple size or other system parameters.

3.3 Sample Problems Used in the Simulation Study

3.3.1 Ancestor Problem

Finding all the ancestors of a certain person in a collection of parent-child relationships (ancestor problem) is a typical inference problem. A computer-generated family tree based on a simple model of human life cycle is used. As shown in figure 7, the PR consists of 1800 relationships among persons and several related rules. This problem has the following characteristics:

(a) The tuple size of the TR is small (44 or 60 bytes).

(b) The size of the PR (64 Kbytes) is large compared with that of the TR (36 Kbytes).

(c) Since the pair generator of the UE generates only pairs of tuples which are exactly unifiable, the antiselectivity ratio in formula (2) is 0.

3.3.2 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 on the left side. The problem has the following characteristics:

(a) The tuple size of the TR is large (from 188 to 220 bytes).
(b) The size of the PR (2 Kbytes) is very small compared with that of the TR (386 Kbytes).
(c) Since each tuple has a similar form, the pair generator of the UE generates almost all tuple combinations. Less than one seventh of these are exactly unifiable. Therefore, the anti-selectivity ratio is large.

The differences between the sample problems are listed below.

	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 8 Comparison of the Sample Problems

4 Results and Discussion

4.1 SP versus MP

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, which was adequate for our measures.

Figure 9 shows the relationship between UEtotal and the time to execute the ancestor problem. 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 small.

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 10 shows the relationship between UEtotal and the page loading factor in executing the ancestor problem.

For the SP method, it shows that:

(s3) The page loading factor does not vary when the number of UEs increases.

(s4) The page loading factor for a 64 Kbyte page is smaller than those for 4 and 16 Kbyte pages. This is because the probability of generating resultant tuples varies according to the parts of the relations.

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 the 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).

Figure 11 and 12 show the results of the eight-queen problem. 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 12 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.

4.2 Details of the MP Method

4.2.1 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. The reason is suggested by figure 13, which shows a sample history of activities of UEs in that case. It sometimes occurres 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.

4.2.2 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.

4.2.3 Relationship between UEtotal and Performance

Figure 14 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 1 Kbyte. The figure shows 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.

4.3 Discussions on KBM Architecture

4.3.1 Use of MPPM

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

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

Figure 15 summarizes the port utilization ratio in both problems.

	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%

Figure 15 Port Utilization Ratio of the MPPM

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 40% 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 20% smaller, assuming that UEtotalis equal and that the MPPM has only one third of the ports as in the current architecture model.

4.3.2 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 processing algorithm of UE. We are currently

conducting research into an efficient U-join algorithm based on the clustering of terms.

5 Conclusions

A simulation study of a knowledge base machine architecture and its control strategies wss reported. For the control strategy, the MP method was proposed and found to be superior to a conventional 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 an efficient unification join algorithm based on the clustering of terms.

ACKNOWLEDGEMENTS

We express thanks to Mr. Kazuhide Iwata for fruitful suggestions and to Mr. Shoju Takahashi and Mr. Toshimasa Uehara for the development of the simulation software.

REFERENCES

[Kakuta 85] Kakuta, T., et al.: The Design and Implementation of Relational Database Machine Delta, Proc. IWDM'85, 1985.

[Monoi 86] Monoi, h., et al.:A Large-Scale Knowledge Base Machine Control Technique Using Multi-Port Page-Memory, ICOT TR-156, 1986.

[Morita 86] Morita, H., et al.: Retrieval by Unification Operation on a Relational Knowledge Base, Proc. VLDB'86, 1986.

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

[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, Proc. IFIP TC-10 Working Conference on Fifth Generation Computer Architectures 1985.

[Ohmori 86] Ohmori, T., et al.: An Approach to a Relational Database System Integrated with the Inference Power, AI86-21, Technical Report of IECE, AI86-21, pp.30-40, 1986(In Japanese).

[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, Proc. International Symposium on Computer Architecture, 1986.

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)

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

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

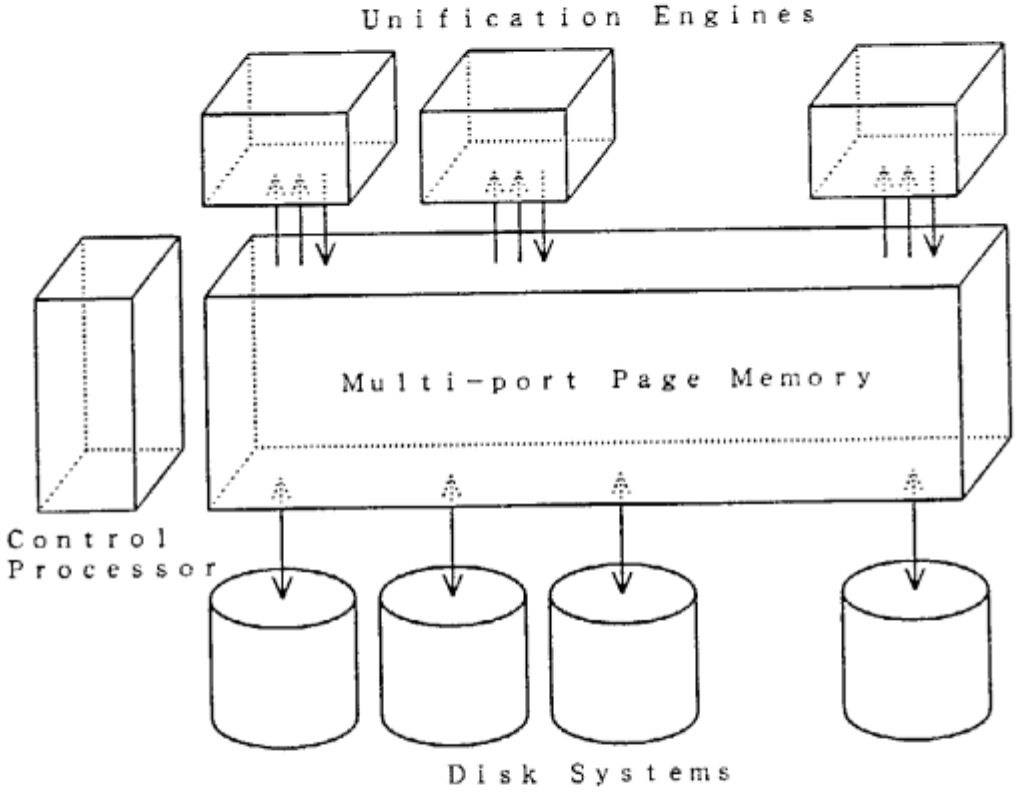


Figure 2 System Configuration of a KBM

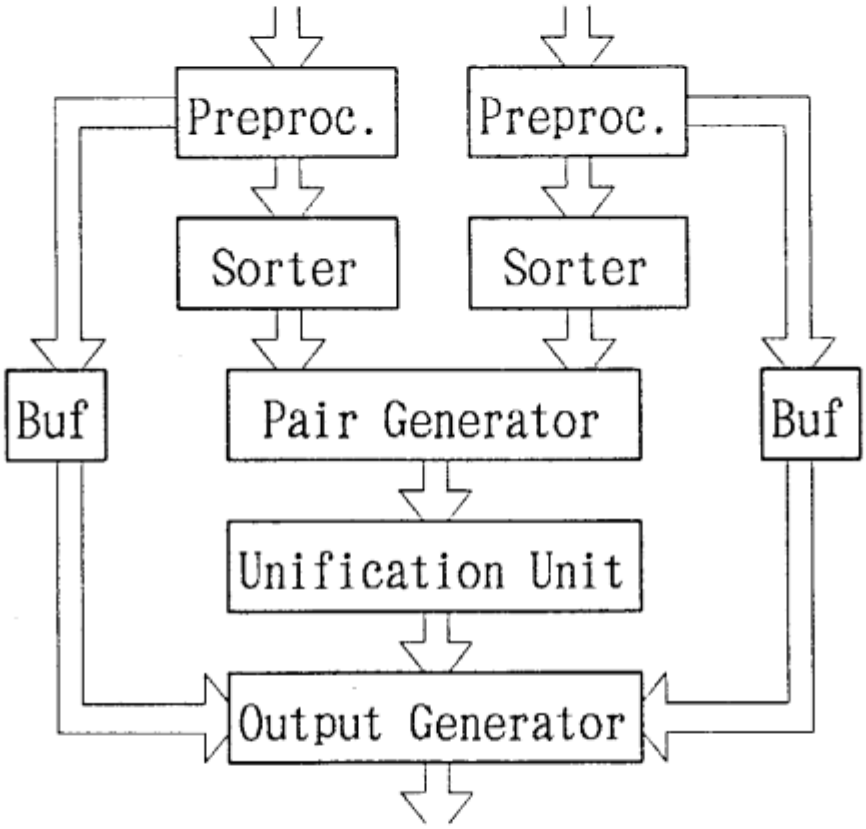
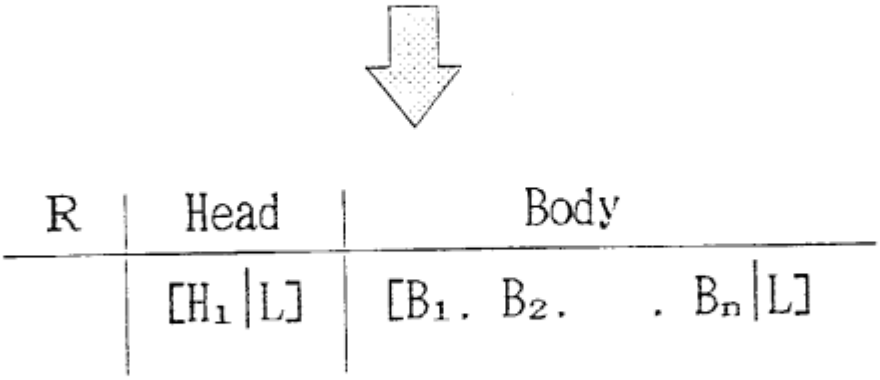
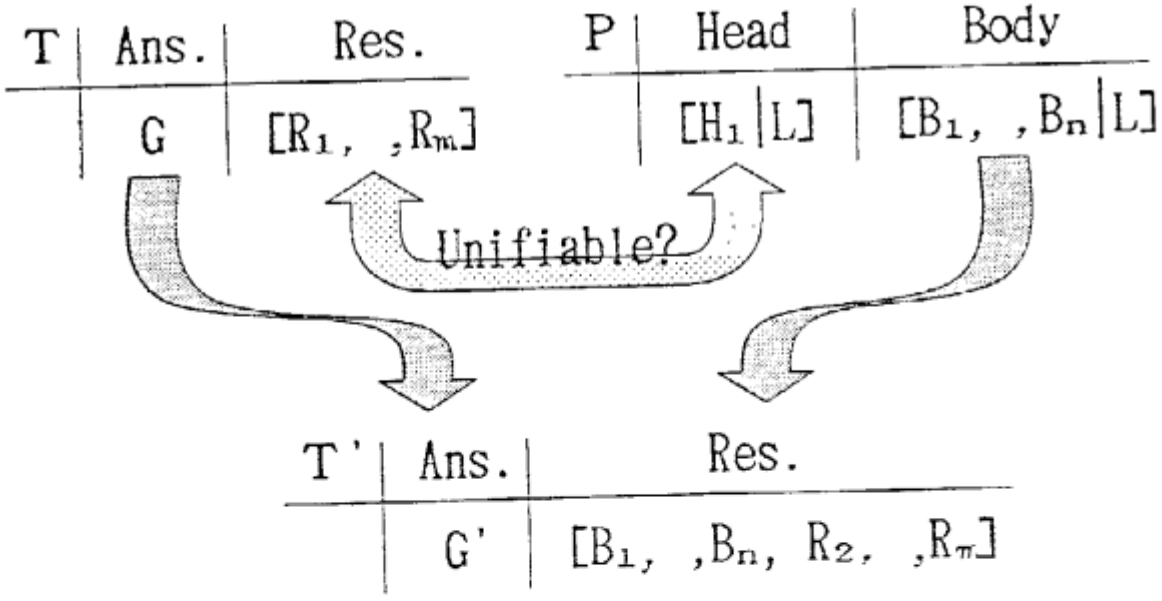


Figure 3 Internal Configuration of UE

H_1 :- B_1. B_2. . B_n



(a) Representaion of Definite Clauses For Input Resolution



(b) U-join in Input Resolution

Figure 4 Knowledge Representation for Input Resolution

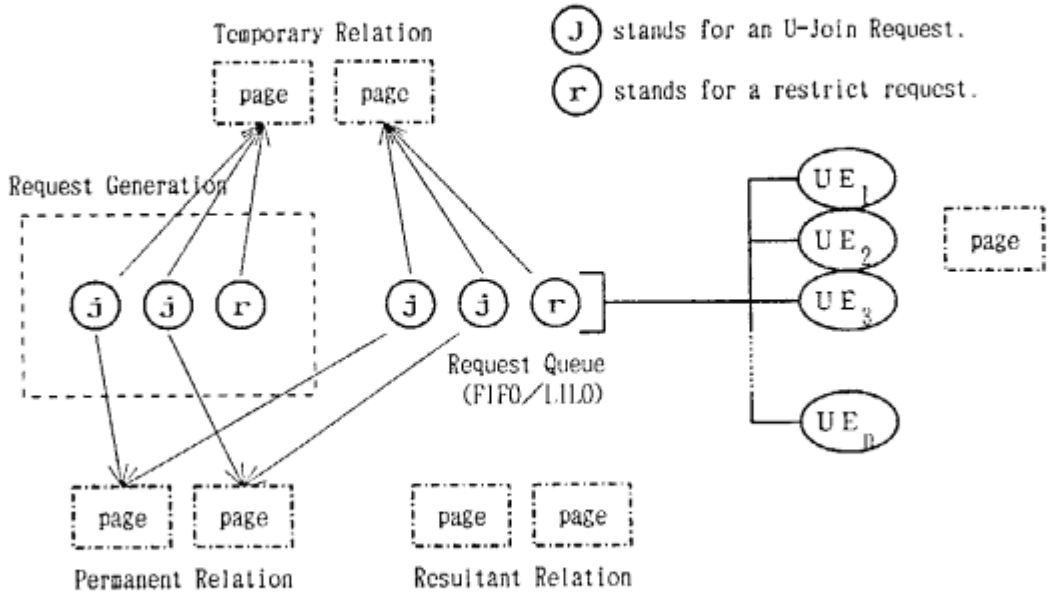


Figure 5 Control Model based on the SP Method

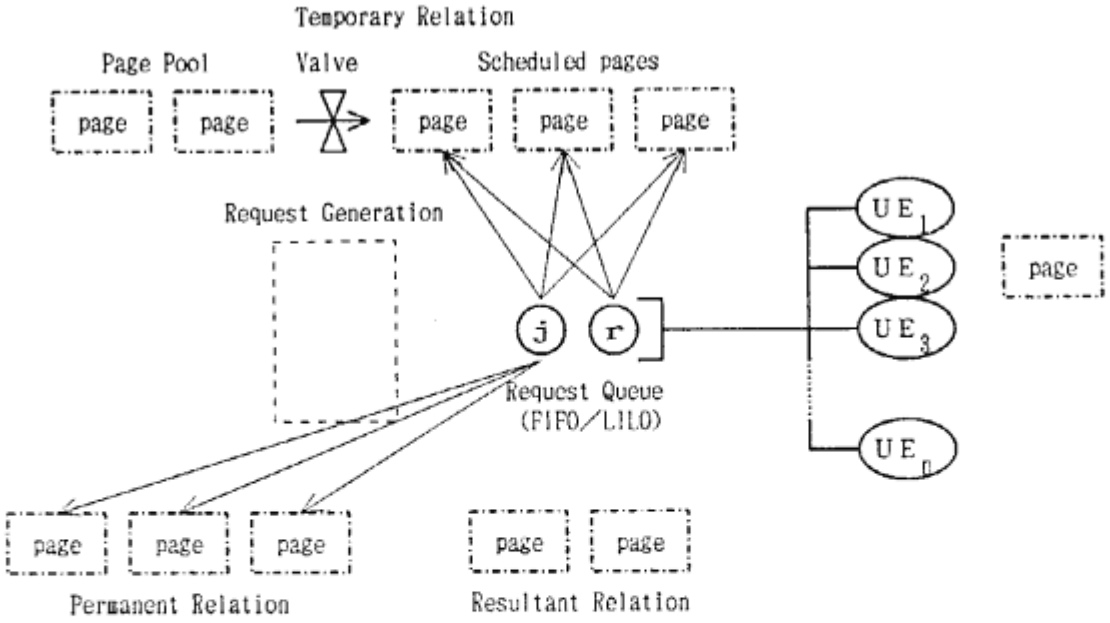


Figure 6 Control Model based on the MP Method

Permanent Relation

```
ancester(A.B)  :- father(A.B).
ancester(A.B)  :- mother(A.B).
ancester(A.B)  :-
       father(A.C).ancestor(C.B).
ancester(A.B)  :-
       mother(A.C).ancestor(C.B).

unrelated eight rules

father(*. *).   × 900
mother(*. *).   × 900
```

ave. tuple size	36.16 byte
total data size	65520 byte

Goal Relation

```
? ancester(m0999. X)
```

Figure 7 Ancestor Finding Problem

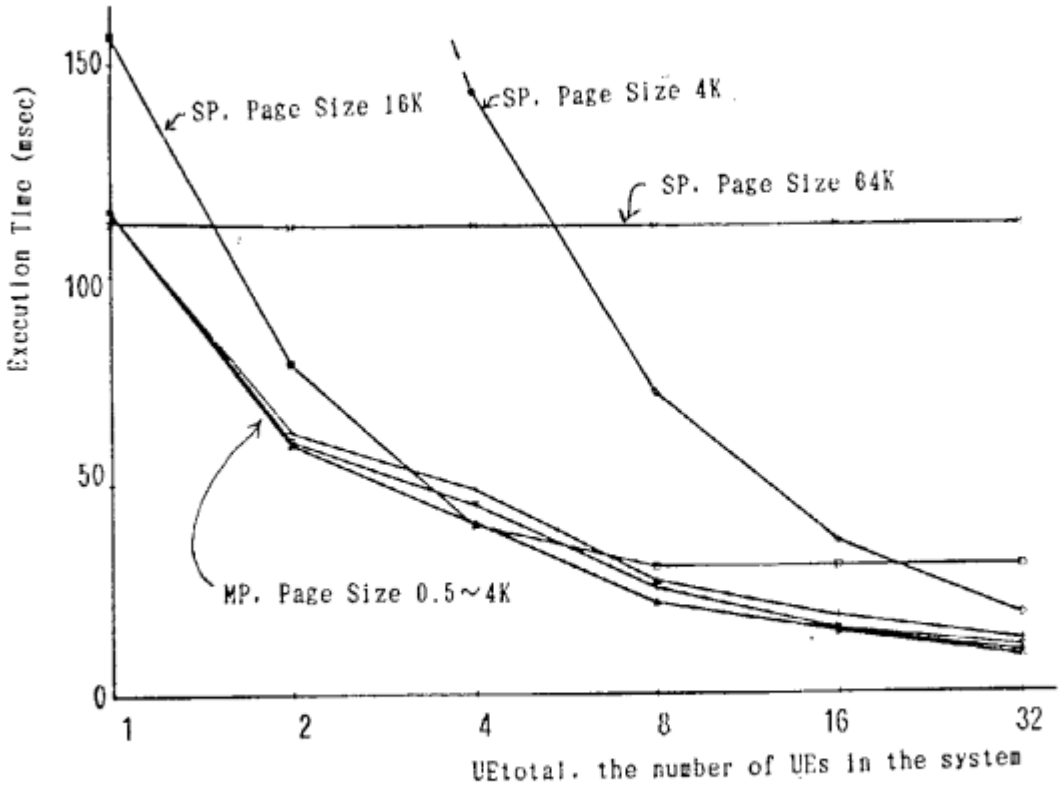


Figure 9 Relationship between UEtotal and Execution Time (Finding Ancestors)

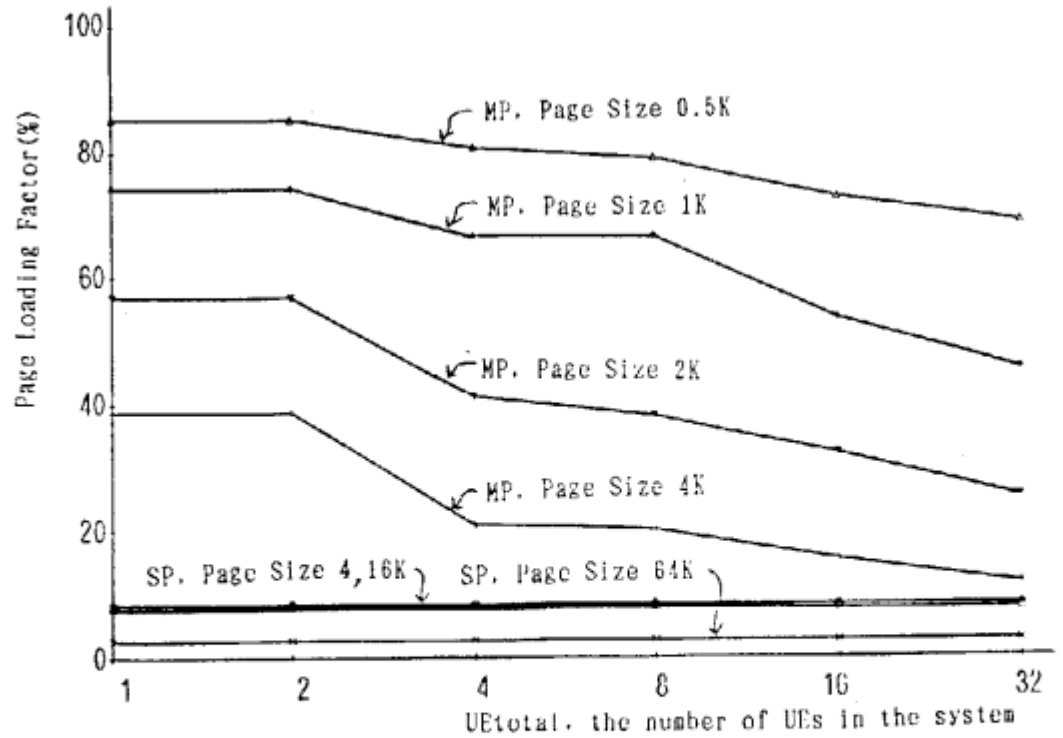


Figure 10 Relationship between UEtotal and Page Loading Factor (Finding Ancestors)

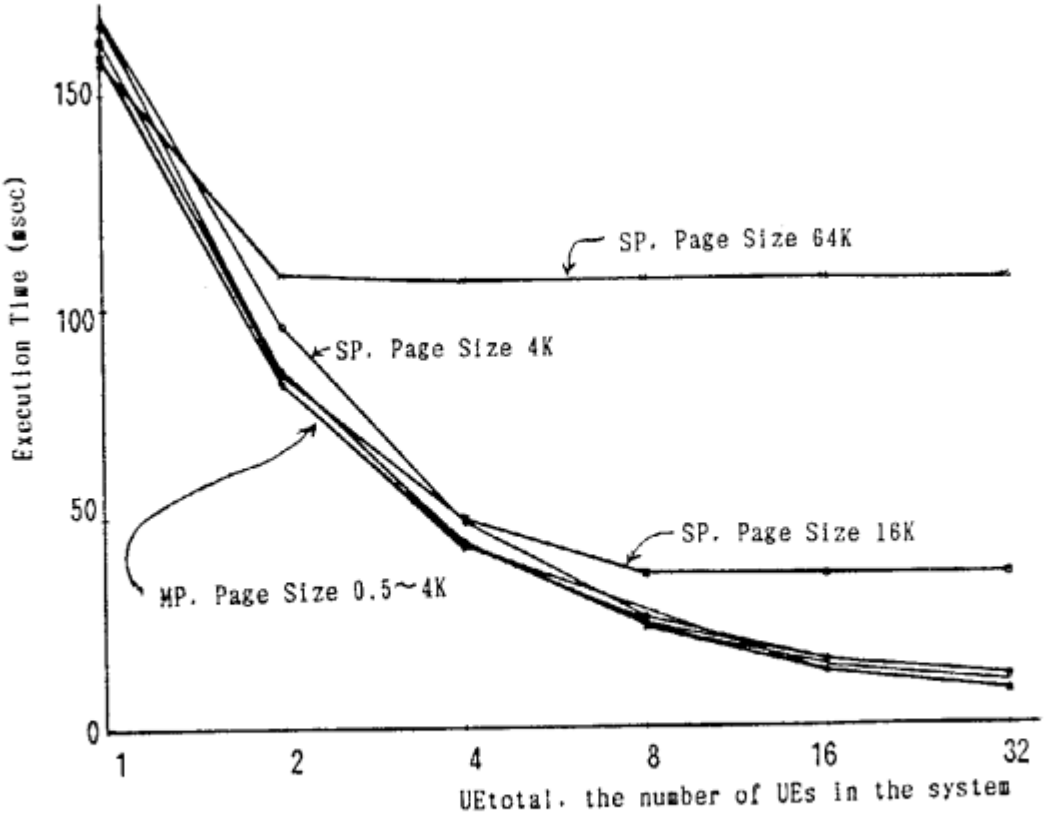


Figure 11 Relationship between UEtotal and Execution Time (8 Queen)

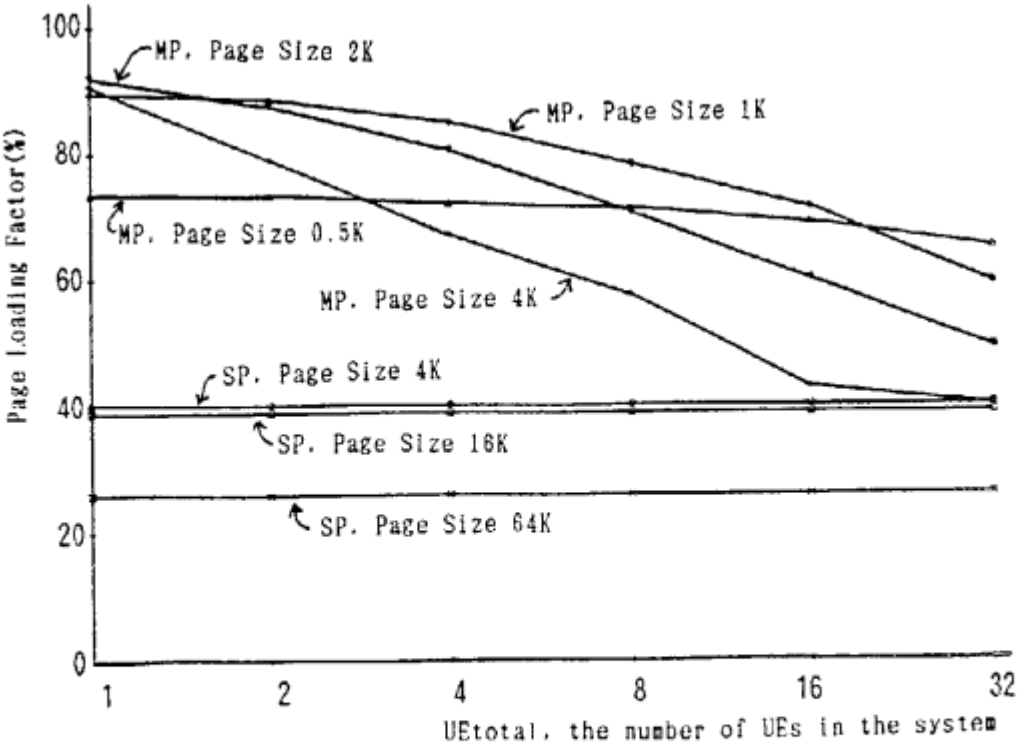


Figure 12 Relationship between UEtotal and Page Loading Factor (8 Queen)

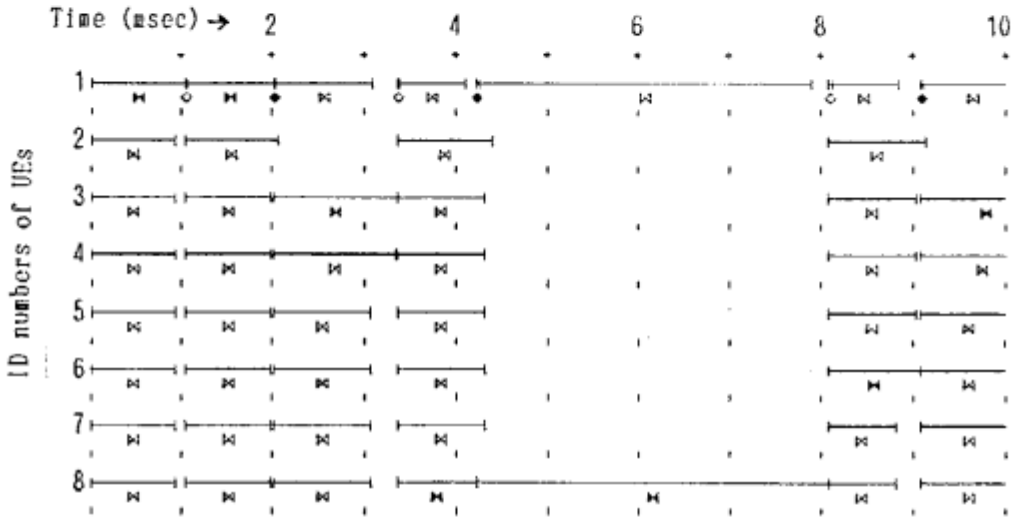


Figure 13 A History of Unification Engines' Activity

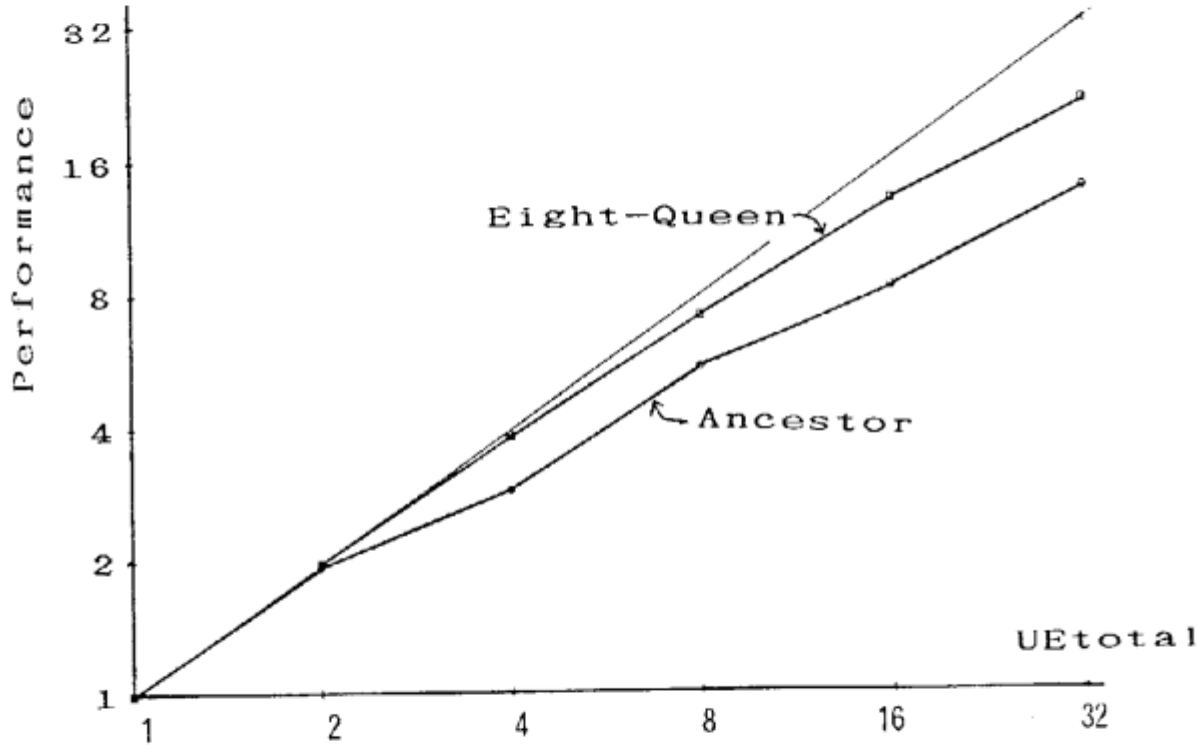


Figure 14 Relashonship between UEtotal and Performance