PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT.

ARCHITECTURES AND HARDWARE SYSTEMS: PARALLEL INFERENCE MACHINE AND KNOWLEDGE BASE MACHINE

Kunio Murakami, Takeo Kakuta, and Rikio Onai

Research Center
Institute for New Generation Computer Technology
Tokyo, Japan

ABSTRACT

The parallel inference machine and knowledge base machine are components forming the core of the Fifth Generation Computer Systems (FGCS) hardware. Of the ten-years research program of the FGCS Project being carried out at ICOT, the first three years (initial stage) have been assigned to basic research on individual FGCS components. This paper describes the current research and development status of the parallel inference machine and knowledge base machine.

1 INTRODUCTION

The FGCS research and development aim is to build a prototype of a knowledge information processing system capable of efficiently performing knowledge-based problem solving and inference. Toward this end, a ten-year period has been assigned to the FGCS Project, and this period has been further divided into three stages. The goal of the initial three-year stage is to conduct basic research on individual system components in order to establish basic configuration technology for subsystems which are to be realized in the intermediate four-year stage.

The parallel inference machine and knowledge base machine are the most important hardware components of the FGCS. In the FGCS prototype to be completed as the final product of the project, the two machines will be integrated through a close link. In the initial stage, however, research and development are proceeding separately for each machine with research themes separately determined, since the initial stage mainly aims to conduct research and development of individual component technologies to establish the basic technology for the hardware, called the inference subsystem and knowledge base subsystem to be built in the intermediate stage.

The research and development of the parallel inference machine in the initial stage consist of these three items:

(1) A parallel-type inference basic mechanism to manage the parallel execution of inference operations.

(2) A dataflow mechanism to rapidly execute inference operations on the basic of the dataflow concept.

(3) An abstract data-type mechanism to support various language functions based on the abstract data-type concept.

The current status of the parallel inference machine research and development is briefly summarized below.

(1) For the parallel-type inference basic mechanism, three basic mechanisms—reduction, clause-unit processing, and complete-copying—were chosen; each mechanism went through basic examination and then software simula-

tion for evaluation. Also environments in which the parallel inference machine will operate were analyzed. At present, a prototype of a dedicated system for simulation is being built. In the first half of the intermediate stage, the prototype will collect various data items to evaluate the basic mechanisms by contrast with the dataflow mechanism.

(2) The dataflow mechanism went through basic examination and then software simulation for evaluation. At present, an experimental machine is being constructed. In the first half of the intermediate stage, the experimental machine will collect various data items to evaluate the dataflow mechanism.

(3) The abstract data-type mechanism was investigated, in the process of examining the kernel language version 1, to review and define its concept.

Section 2 outlines the results of the parallel inference machine research, with emphasis on the results of the characteristics analysis of parallel programs and the architectures of the reduction and dataflow mechanisms.

The research and development of the knowledge base machine in the initial stage consist of these three items:

(1) A knowledge base mechanism to provide an overall management of the execution of knowledge base operations.

(2) A parallel-type relational and knowledge operation mechanism to provide speedy knowledge accumulation, retrieval and updating.

(3) A relational database mechanism to provide storage and management of large amount of knowledge.

The current status of the knowledge base machine research and development is briefly summarized below.

(1) Research and development of the knowledge base mechanism will start from the beginning of the intermediate stage, using the results of various experiments performed on Delta, the relational database machine being developed in the initial stage. This is because we have concluded that adequate amounts of information concerning knowledge base operations should be obtained prior to the research of the knowledge base mechanism.

(2) For the parallel-type relational and knowledge operation mechanism, its component, a parallel-type relational operation mechanism is currently under study. Multiple personal sequential inference machines (PSIs) were linked with Delta via a local area network (LAN) to provide an experimental environment. Within Delta, up to four relational database engines can run in parallel. Currently another experimental environment is being implemented in which PSIs and Delta will be connected through a tighter interface rather than LAN. Research on knowledge operation mechanisms and experimental data collection will begin from the intermediate stage using the experimental environment of Delta tightly coupled with PSIs.

(3) For the relational database mechanism, the relational database machine Delta has been under development to provide an experimental environment for basic technology research necessary for realizing the knowledge base machine, as well as for building an ICOT software development system, consisting of Delta, capable of supporting a large-scale database, linked with PSIs via LAN.

Section 3 outlines the results of the knowledge base machine research, with emphasis on the architecture of the relational database machine partially developed in the initial stage.

2 PARALLEL INFERENCE MACHINE

The ultimate aim of this research and development is a machine that will enable the execution of parallel inferences, the central concept of the Fifth Generation computer. The intermediate-stage parallel inference machine[1] will be a milestone on the road to achieving this goal.

In the initial-stage, basic research and development will produce a fully-operational parallel inference machine.

2.1 Research theme in the initial stage

(1) Analysis of kernel language

The project will statically and dynamically analyze Prolog or Concurrent Prolog [Shapiro 83], on both of which the kernel language is based. The results of this analysis will be reflected in the architecture of the parallel inference machine.

(2) Comparing various parallel inference mechanisms

The project will study various mechanisms of parallel inference and the architecture of various parallel-inference machines based on these mechanisms. These approaches will be evaluated using software simulation. Experimental machines with 8 to 16 modules will be tested.

2.2 Analysis of Kernel Language

Programs written in Prolog have been analyzed statically and dynamically, and different kinds of data have been collected [Onai 84b].

2.2.1 Static Analysis

(1) Data collection

By reading programs from the topdown, the following information was collected:

- the number of OR relations (in which clauses have the same head predicate symbol and the same number of arguments)
- the number of AND literals
- the number of arguments in head predicates and the number of their structure data arguments
- the number of arguments in the body and the number of their structure data arguments
- the frequency of evaluable predicates
- the number of cut operations

Clauses are divided into the following two types:

- Inference clause: a clause in the OR relation that includes at least one rule.
- Database clause: a clause in the OR relation that consists of unit clauses only.

(2) Results of Static Analysis

(a) Static analysis of inference clauses

The project statically analyzed 33 Prolog programs (e.g., BUP tracer, Morphological analysis, Predicate logic formula simplification) developed at ICOT and consisting mainly of inference clauses. The data obtained is as follows:

Average number of cuts per clause: 0.65
Average number of references: 3
Average number of arguments in head predicate: 3
Ratio of structure data to head predicate arguments: 0.2(20%)
Average ratio of evaluable predicates to literals: 0.5(50%)
(Ratio of evaluable predicates not requiring unification: 0.25)

(b) Static analysis of database clauses

The project analyzed six ICOT Prolog programs consisting of database clauses (e.g., dictionary programs).

[1] Equipped with an execution control mechanism for parallel inference, consisting of one hundred or more modules, and using some LSIs.

The following data was obtained:

Average number of OR relations per clause: 10 (approximately four times the number of inference clauses)
Average number of head predicate arguments: 2.8

2.2.2 Dynamic Analysis

Sequential Prolog (DEC-10 Prolog) programs were first modified for parallel execution. They were then executed by fixing goals. Static analysis data (except for the number of cuts) and OR parallelism were collected.

Two programs (Morphological analysis and Predicate logic formula simplification) developed at ICOT were changed to parallel Prolog programs for dynamic analysis. The results are shown in Table 2.1.

Table 2.1

	Morphological analysis	Predicate logic formula simplification
Number of dynamic OR relations	7.1	4.6
Ratio of structure data	32	50
Ratio of evaluable predicates	0.4	0.8
Degree of OR parallelism	8.3	3.2

2.2.3 Summary of the Results

(1) Since a sequential Prolog program runs on small memory space at execution time, many cuts are used and the Prolog program becomes deterministic. However, it may be changed into a parallel Prolog program and some parallelism may be obtained.

(2) Since about half of the AND literals are evaluable predicates, the speed of evaluable predicate execution affects the overall execution of a program.

(3) The number of OR relations in a database clause is approximately four times the number in an inference clause. This ratio increases as the database clause becomes larger. In a program including a large database, increasing unification speed greatly speeds up program execution.

2.3 Machine Architecture

At present, four mechanisms are being studied [ICOT 84].

(1) Reduction Mechanism [Onai 84b]

When executed, a Prolog or Concurrent Prolog program generates resolvents from a goal and a clause. This can be regarded as a process in which a goal modifies itself using a clause as a rule. The reduction mechanism can also be viewed as a kind of self-modification. Thus, there is a close similarity between the execution of Prolog or Concurrent Prolog programs and the reduction mechanism. Accordingly, the reduction mechanism was selected for a machine architecture that executes Prolog programs in OR parallel, and Concurrent Prolog programs in AND parallel.

(2) Dataflow Mechanism[Ito 83][Ito 84]

In the dataflow concept, execution starts when data necessary for the execution arrived. This concept can result in parallelism regardless of whether it is explicitly indicated in the program. This mechanism executes kernel language programs in parallel based on the dataflow concept.

(3) Complete-copying Mechanism

Complete-copying is type of reduction mechanism. Even if a process includes several literals (subgoals) and only one literal (subgoal) is reducible, the whole process is copied and transferred to a unit that executes the unification process. This increases the number of copies and the length of a packet in the network, while enhancing the independence of each process.

(4) Clause-Unit-Processing Mechanism

In response to a request from an idle processing unit, a busy processing unit sends a process. Thus, this mechanism can avoid an explosion of resource requests. However, it takes time for all the processing units to become busy.

2.3.1 Parallel Inference Machine based on Reduction Mechanism (PIM-R)

(1) Overall Structure

As shown in Figure 2.1, PIM-R consists of Inference Modules, Structure Memory Modules and Networks.

(a) Inference Module

This module consists of a Process Pool Unit and a Unification Unit.

- Process Pool Unit

The Process Pool Unit stores processes and the relations among them. Reducible processes are selected by the Controller and sent to the Unification Unit. Whether a process is reducible is indicated through the sequential AND operator, the parallel AND operator, the commit operator and messages from the producer process.

The Message Board stores values sent through channels and suspend processes. When the consumer process is in suspension, the Message Board is checked to see if it is receiving a value from the producer process. If no value has been sent, the consumer process is connected to the suspend process list in the Message Board. If a value is bound to a channel by the producer process, the value is written into a specified cell of the Message Board and sent to a suspend process waiting for the value.

- Unification Unit

In this unit, the matcher selects unifiable clauses using the following items sent from the Process Pool Unit:

•• the predicate name of a goal

•• the number of its arguments

•• the data type of the first argument.

Then, the goal and clause are unified in the Unifier. The unified result is returned to the Process Pool Unit. A clause and related information are stored in the Clause Pool.

(b) Structure Memory Module

This module stores structure data. There are two special memories for list data and vector data.

(2) Features of PIM-R

(a) A process in the Process Pool consists of more than one subgoal (body literal). In PIM-R, however, only a reducible subgoal is copied and sent to the Unification Unit (partial copy mechanism). Thus, the number of copies is smaller and the network packet length is shorter compared with the mechanism in which all the subgoals are copied.

(b) PIM-R has a function that executes Prolog in OR-parallel and Concurrent Prolog in AND-parallel. Both Prolog and Concurrent Prolog are base languages of Kernel Language version 1 (KL1). Therefore, PIM-R can process KL1 programs in parallel.

2.3.2 Parallel Inference Machine based on Dataflow Mechanism (PIM-D)

(1) Overall structure

As shown in Figyre 2.2, PIM-D is com-

posed of a group of Processing Element Modules (PEMs), which perform basic inference processing, a group of Structure Memory Modules (SMMs), which store and manage structure data, and the network connecting them.

(a) Processing Element Modules

PEM interprets the unification procedure and controls instruction execution, procedure calling, and the basic pattern matching operation. It also executes several built-in predicates. It is further divided into the following two units:

- Instruction Control Unit (ICU)

The ICU is activated when a result packet representing a token arrives at its input port. A result packet consists of a process identifier, which specifies a process, the destination address of an instruction to which the result is transferred, and a result value, which is an operand of the instruction. Using the process identifier and the destination address of the result packet, the ICU determines whether all operands for the instruction are now present (i.e. whether the instruction is executable). If the instruction is executable, the ICU generates an instruction packet and sends it to the Execution Unit.

- EXecution Unit (EXU)

The EXU receives an instruction packet from the ICU, decodes the operation code, controls its execution, and sends result packets to the next destination.

(b) Structure Memory Module

The SMM is initiated when an SMM instruction packet is sent from the EXU via the network. It executes reference count control instructions, data read-write instructions and free-cell control instructions.

(2) Features of PIM-D

Three types of parallel processing are implemented: OR parallel, AND parallel, and parallel unification.

Also, structure data is shared among the processes for unification. Structure data is distributed and stored in the group of SMMs (i.e. the address (pointer) is transferred to the group of SMMs). When it is necessary to access the structure (on demand), PEM accesses SMM. This can avoid the overhead of copying structure data in applications where complicated structure data must be processed.

2.3.3 Evaluation through Software Simulation

Software simulators have been developed for the dataflow and reduction mechanisms. Figure 2.3 shows the results of a simulated OR-parallel execution for the four Queens program (effect of the number of units).

Saturation occurs at about six or seven units, as shown in the figure, and the degree of OR-parallelism for the program is 6.2. Therefore it has been confirmed that either mechanism can take full advantage of parallelism in this program.

3 KNOWLEDGE BASE MACHINE

This section describes research and development of the knowledge base machine, particularly its architecture.

3.1 Research theme in the initial stage

It was decided to implement a relational database machine in order to develop the basic techniques necessary for developing a knowledge base machine in the intermediate stage of the project and for investigating a prototype database machine capable of parallel relational and knowledge operations [Murakami 83].

Development of the ICOT relational database machine (called RDBM Delta) has

two concrete purposes. One is to create an experimental environment in which various knowledge base functions and their implementation can be investigated. The other is to connect the machine via a local area network (LAN) [Taguchi 84] with the personal sequential inference machine (called PSI) being developed separately [Uchida 82]. PSI is a software development tool that makes use of Kernel Language Version-0 (KL0).

3.2 Delta Architecture

3.2.1 Processing Flow of Query from the Host

Figure 3.1 shows the processing flow of a query made from the host to Delta. A user on the host describes a program in Prolog and makes a query to the external database. The PSI software generates Delta query plans according to the contents of the query made by the user and the Prolog program. It then converts the query plans to Delta commands in the form of packets, making a query to Delta via the LAN interface. Delta extracts Delta commands from this packet information. After analyzing these retrieval commands, Delta retrieves the necessary data from the database, and transmits it to the host via the LAN interface.

Figure 3.2 shows a concrete example of processing by the PSI and Delta. The Prolog phrases and relations shown in Figure 3.2 (1) are stored in PSI and Delta. Figure 3.2 (2) shows the processing flow for the query, "`?-son(S,mary)`", made from the PSI user. The PSI software first generates plans 1 and 2, converts them to Delta commands 1 to 4, then transmits them to Delta. Delta retrieves relation data from the database and transfers the results to PSI.

As indicated by this example, access to the external database by Prolog program is characterized by the following:

(1) Equal frequency of access to all attributes.

(2) Queries to the relation are converted into relational algebra operations, such as selection, join, and union. There are a number of operations that require intensive processing, such as join and union.

(3) Relatively few attributes in typical relations.

3.2.2 Characteristics of the architecture

Various studies have been made of relational database machines [Bancilhon 82], [DeWitt 79], [Kitsuregawa 83], [Schweppe 82], [Tanaka 82]. Delta's design incorporates the following architectural characteristics in consideration of the external database access characteristics described in section 3.2.1 [Kakuta 83], [Shibayama 82], [Shibayama 83], [Shibayama 84a], [Shibayama 84c].

(1) Functionally distributed multiprocessor configuration

Functions are distributed among five processors to lighten the load on any one processor's control program, to extend the functions of the prototype machine and to improve database processing performance.

(2) Relational Algebra command interface with host

Relational algebra commands (called Delta commands) have been selected as the logical command interface with the host. Also, a tuple-based format has been adopted for data transfer with the host. Delta commands and tuple-based data are transferred between the host and Delta in packets in accordance with LAN protocol.

(3) Attribute-based internal schema

An attribute-based internal schema and a two-level clustering method have been adopted. Details are given in section 3.4.

(4) Dedicated hardware for relational algebra operations

The relational database engine (RE) consists of a 12-stage pipeline sorter using the pipelined two-way merge-sort algorithm and a merger unit for processing relational algebra operations. Up to four REs may operate in parallel under the Control Processor (CP).

(5) Interface for two-channel stream data transfer

Stream data is independently transferred between the RE and the Hierarchical Memory (HM) via one input channel and one output channel per RE. The channel data transfer rate is 3M bytes per second.

(6) Application of the Hierarchical Memory (HM)

The HM has a two-level structure consisting of a Moving Head Disk (MHD) unit and a Database Memory Unit (DMU). DMU data is non-volatile; the entire Delta system has fail-safe power supply unit in case of power failure.

(7) Statistical information collection function

Various experiments will use Delta to establish basic techniques for the development of a future knowledge base machine. Delta has functions for collecting performance information and various types of statistical information during such experiments.

3.2.3 Overall Configuration and Specification

Delta consists of an RDBM Supervisory and Processing (RSP) subsystem, which takes charge of overall control/monitor and operation processing, and a Hierarchical Memory (HM) subsystem, which takes charge of storage, retrieval, and updating of relation data. Figure 3.3 shows the Delta configuration.

The HM and the RSP are connected via a total of 11 channel interfaces.

Figure 3.4 shows the specification for the Delta system.

3.2.4 RSP Subsystem Configuration

The RSP subsystem consists of the CP (Control Processor), IP (Interface Processor), MP (Maintenance Processor), and RE (Relational database Engine), each of which consists of hardware and software.

(1) RSP hardware configuration

Each unit basically consists of a processor and either 512K bytes or 1M bytes of main memory. The RE is provided with dedicated hardware for sorting and merging, and the CP is provided with a 15M-byte semiconductor disk storage for increasing memory capacity. The MP is connected to the Delta system monitor display, the desk console, and the MTU for collecting RSP log information. Each RSP unit is connected to the others via three IEEE 488 buses. Each CP, IP, and MP unit is provided with one HM adapter(HMA), which serves as interface hardware for the HM. The RE is also provided with an HMA for input and another for output.

(2) RSP software configuration

The following is a list of software functions divided among the RSP units.

(a) CP

(i) Transaction management

Transaction execution management, Delta command analysis, generation and execution of subcommands.

(ii) Management of relational data management information

(iii) Communication control for the IP,RE, and MP

(iv) Recovery management [Kakuta 84]

Data recovery is performed in units of transactions.

(b) IP

(i) LAN interface control

(ii) Delta command and data extraction control

(iii) Command tree parallel reception control

A command tree consists of two or more Delta commands, and is the unit in which any sequence of meaningful processing is carried out. This software identifies and controls a command tree for each transaction in the received packed array.

(iv) Data format conversion

When data is input from the host, it is provided with a tuple identifier (TID). The TID is removed when data is output.

(v) Data transfer with the HM

After the CP completes an instruction that reserves a buffer for data transfer between the IP and the HM, the IP performs data transfer with the HM.

(c) RE

(i) RE control

Analysis of subcommands from the CP, sequence control of RE operations, and control of I/O operations with the HM.

(ii) Relational operation support

Performs relational algebra operations and mathematical operations that cannot be processed by the merger hardware.

(d) MP

(i) Delta system status monitoring, and configuration

(ii) Delta system start-up and shut-down

(iii) Database loading and dumping

(iv) Statistical information collection

(v) Operator command and message management

3.2.5 HM Subsystem Configuration

The HM subsystem consists of HM hardware and software designed to efficiently manage the writing of Delta data to and the reading of Delta data from the storage area, as instructed by the RSP subsystem.

(1) HM hardware configuration

In the final configuration, the HM hardware comprises of a non-volatile high-speed DMU having a memory capacity of 128M bytes, magnetic disk units having a maximum capacity of 20G bytes, disk controllers, and an HM controller, which controls the HM execution.

(2) HM software configuration

HM software functions are listed below.

(a) HM subcommand processing

The following processing is performed by HM subcommands specified by the RSP.

(i) Attribute definition and operation

Generates and deletes attribute definitions; transposes tuples to attributes, and vice versa.

(ii) Update processing

Inserts, deletes, and updates attributes.

(iii) Clustering operation

(iv) Data transfer management

Transfers stream data between the HM and the RE, and packed data between the HM and the IP or CP.

(v) RSP buffer management

Reserves and releases RSP buffers inside the HM.

(b) Memory resource and MHD space management

Controls the transfer of attribute and

directory information between secondary storage and the DMU, and manages memory resources for storing such information.

(c) Data recovery processing

Recovers attribute information by recovery subcommands from the CP.

3.3 Relational Database Engine Processing

The RE consists of a sorter and a merger, which perform relational operations for sorted results. A two-way merge-sort algorithm [Knuth 73] has been adopted for the sorter, a major component of the RE.

(1) Two-way merge-sort algorithm

This algorithm rearranges input record values in either ascending or descending order. An array of input records is regarded as a collection of sorted arrays, each of which has a length of 1. Sorted arrays are merged in pairs so that the length of each sorted array is doubled at each sort step. When this operation is carried out in the pipeline processing configuration, the use of the $log_2 n$-stage sorter enables a sort output with a cycle time of $2n + (log_2 n) - 1$ [Todd 78] (n = number of records).

(2) Configuration of the RE [Oka 84]

The RE consists of an IN module, a sorter consisting of 12-stage sorting cells, a merger, two HM adapters, and the CPU and RE control programs, which control the above units. Figure 3.5 shows the configuration of the RE.

RE operations, based on relational algebra, sort item groups consisting of attribute values and TIDs while transferring them as a stream, performs relational operations from the head of sort output, and outputs combinations of items based upon the results of the operation [Iwata 84],[Sakai 84].

The IN module transforms input stream data item into an internal format suitable for the sorter and merger: field ordering, which replaces the head of each item to be sorted, and data type transformation. The sorting cell is made of two FIFO buffers, a comparator, and a multiplexer for merging. Each FIFO buffer has a capacity of 16 bytes at the first stage and 32K bytes at the twelfth stage.

The merger consists of an operation section, and an output control section. The operation section comprises of a comparator and two 64 Kbyte memories having FIFO function, and performs relational algebra operations by comparing two sorting stream data. The output control section comprises of two 16 Kbyte buffers, two field-ordering, field-selection and data-type-transformation circuits, a selector and an output sequence controller. This section performs field reordering and selecting of an output data item, and adding a new TID to it, and then transferring output data items to the HM via an HMA.

3.4 Internal Schema and Storage Allocation

3.4.1 Internal Schema

The internal schema, which greatly affects the performance of the database machine, falls into two categories: tuple-based schema and attribute-based schema. The attribute-based schema, in which tuples constituting a relation are divided into attributes and stored in units of attributes, has been adopted in Delta for the following reasons [Miyazaki 83]:

(1) Since Delta constitutes a logic programming environment, all attributes are very likely to be accessed on a symmetrical basis.

(2) In the attribute-based schema, only those attributes that are necessary are accessed for each query. In the tuple-based schema, on the other hand, has the disadvantage that unnecessary at-

tributes must also be accessed.

(3) The attribute-based schema provides storage suited to the RE sort/merge algorithm, with respect to its processing efficiency and performance.

The application of the attribute-based schema, however, has the following disadvantages.

(a) Tuple reconstruction and tuple-to-attribute transposition are necessary.

(b) Because tuples are identified by attaching a tuple identifier (TID) to each attribute, memory space cannot be utilized as efficiently.

In Delta, the two-stage clustering method, in which the attribute identifier, attribute value range, and TID value range are used as search keys has been applied to reduce the search space for attribute information. Figure 3.6 shows the internal schema of Delta. Attribute information consisting of a TID and a value is distinguished from other attributes by attribute IDs, sorted first by attribute values divided into value ranges, then temporarily stored in a work space. Information in each work space is then sorted by TIDs and stored in page space groups consisting of the TID-attribute value pairs, as shown in the right column in Figure 3.6. Next, an attribute value range table and a TID value range table are generated. The attribute value range table contains pointers to the TID value range table. The TID value range table contains pointers to the TID-attribute value page space.

3.4.2 Storage allocation

Data is stored in three HM data sets, depending on its contents.

(1) RSP data set

Database management information, called a directory. Data contents are managed by the CP.

(2) Attribute data set

Space in which attribute information is stored.

(3) RSP page data set

3.5 Performance Estimation

To estimate the performance of Delta, the characteristics of tuple selectivity and processing time were estimated for a selection query put to Delta under the following conditions. The performance characteristics are illustrated in Figure 3.7 [Shibayama 84a], [Shibayama 84b].

Conditions

(1) A relation consists of 10,000 tuples, each of which consists of 10 attributes.

(2) Each attribute is 10 bytes long.

4 CONCLUSION

The research and development of the FGCS hardware in the initial stage is being carried out individually for the parallel inference machine and knowledge base machine.

For the parallel inference machine, the environmental conditions in which it will operate were analyzed and an architecture was established for each of the four selected mechanisms: dataflow, reduction, clause-unit processing, and complete-copying. At present, a software simulator is being developed or an experimental machine is being constructed for each method.

Research items to be carried out hereafter for the parallel inference machine are listed below.

(1) More detailed software simulation will be performed for the four parallel-inference mechanisms.

(2) Parallel programs will be investigated to evaluate the four mechanisms.

(3) Various experiments will be conducted on the experimental machines, each consisting of 8 to 16 modules, to validate each mechanism.

For the knowledge base machine, the relational database machine Delta is under development; in late May, 1984, two subsystems forming Delta, the RDBM supervisory and processing (RSP) subsystems and hierarchical memory (HM) subsystem, were successfully linked and tested as prototype hardware. At present the software for the two subsystems is being tested for integration. Meanwhile, the investigations are progressing to link Prolog with the relational database machine, and the inference mechanism with the relational database machine [Yokota 83], [Yokota 84a], [Yokota 84b].

Research items to be carried out hereafter for the knowledge base machine are listed below.

(1) The entire system of the Delta hardware will be built and functional enhancements will be added to the Delta software.

(2) Real data will be collected in the operational environment consisting of multiple PSIs linked via LAN and the validity of architecture will be checked.

(3) Algorithms for parallel processing of queries streams will be experimented using up to four relational database engines.

(4) Experiments will be conducted using the environment of Delta tightly-coupled with PSI.

Research and development of hardware in the intermediate stage will be carried out as follows:

First of all, the research items for the initial stage thus far described will be further pursued in a more active manner and various data will be collected to quantitatively evaluate the validity of each mechanism.

Then, on the basis of the evaluation results, the architectures of the parallel inference machine and knowledge base machine will be investigated and the hardware technology to implement these architectures will be developed.

In addition, the inference subsystem and knowledge base subsystem on which several application programs will run will be built based on the parallel inference machine and knowledge base machine, respectively.

The method to connect the inference subsystem with knowledge base subsystem will be studied independently from each subsystem's point of view. Then, the best integrating method of both subsystem will be set up based on these studies.

ACKNOWLEDGMENTS

This research was carried out by the first laboratory of ICOT Research Center in very tight cooperation with manufactures. Many fruitful discussions were done at the meeting of Working Group 1.

REFERENCES

[Bancilhon 82] Bancilhon. F. et al. VERSO: A Relational Back-End Data Base Machine, *Proc. of Int'l Workshop on Database Machines*, Aug., 1982.

[DeWitt 79] DeWitt, D. DIRECT—A Multiprocessor Organization for Supporting Relational Database Management Systems, *IEEE Trans. on Computers*, C-28, No.6, June, 1979.

[ICOT 84] *ICOT Annual Report* (Inference Subsystem), ICOT, 1984.

[Ito 83] Ito, N. Onai, R. Masuda, K. and Shimizu, H. Prolog Machine Based on the Data Flow Mechanism, *ICOT Technical Memorandum TM-0007*, 1983.

[Ito 84] Ito, N. and Masuda, K. Parallel Inference Machine Based on the Data Flow Model, *Proc. of Int'l Workshop on High-Level Computer Architecture 84*, 1984.

[Iwata 84] Iwata, K. et al. Design and Implementation of a Two-Way Merge-Sorter and its Application to Relational Database Processing, *ICOT Technical Report TR-066*, May, 1984.

[Kakuta 83] Kakuta, T. et al. RDBM Delta (I), (II) and (III), *Proc. of 26th National Conf. of IPSJ*, Mar., 1983 (in Japanese), and also *ICOT Technical Memorandum TM-0008*. (in English)

[Kakuta 84] Kakuta, T. et al. Recovery Method in RDBM Delta, *Proc. of 29th National Conf. of IPSJ*, Sep., 1984. (in Japanese)

[Kitsuregawa 83] Kitsuregawa, M. et al. Application of Hash to Data Base Machine and Its Architecture, *New Generation Computing*, Vol.1, No.1, 1983.

[Knuth 73] Knuth, D. E. et al. Sorting and Searching, *The Art of Computer Programming*, vol.3, Addison-Wesley Publishing Co., 1973.

[Miyazaki 82] Miyazaki, N. et al. On Data Storage Schemas in Database Machine, *Proc. of 27th National Conf. of IPSJ*, Oct., 1982.

[Murakami 83] Murakami, K. et al. A Relational Database Machine : First step to Knowledge Base Machine, *Proc. of 10th Symposium on Computer Architecture*, Stockholm, Sweden, June, 1983, also as *ICOT TR-012*.

[Oka 84] Oka, T. et al. Development of a Relational Database Engine, *Proc. of 29th National Conf. of IPSJ*, Sep., 1984. (in Japanese)

[Onai 84a] Onai, R. Shimizu, H. Masuda, K. and Asou, M. Analysis of Sequential Prolog Programs, *Proc. of Logic Programming Conference'84*, 1984

[Onai 84b] Onai, R. and Asou, M. Parallel Inference Machine Based on Reduction Mechanisms — its architecture and software simulation —, *ICOT Technical Report TR-077*, 1984

[Sakai 84] Sakai, H. et al. Design and Implementation of the Relational Database Engine, *Proc. of Int'l Conf. of Fifth Generation Computer Systems* 1984, Nov., 1984, and also *ICOT TR-063*.

[Schweppe 82] Schweppe, H. et al. RDBM—A Dedicated Multiprocessor System for Data Base Management, *Proc. of Int'l Workshop on Database Machines*, Aug., 1982.

[Shapiro 83] Shapiro, E.Y. (Weizmann Institute) A Subset of Concurrent Prolog and its Interpreter, *ICOT Technical Report TR-003*, 1983.

[Shibayama 82] Shibayama, S. et al. A Relational Database Machine "Delta", *ICOT TM-0003*, Nov., 1982.

[Shibayama 83] Shibayama, S. et al. On RDBM Delta's Relational Algebra Processing Algorithm, *Proc. of 27th National Conf. of IPSJ*, Oct., 1983, and also ICOT TM-0023.

[Shibayama 84a] Shibayama, S. et al. A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor, *New Generation Computing*, Vol.2, No.2, 1984.

[Shibayama 84b] Shibayama, S. et al. Relational Database Processing on an Attribute-based Schema, *Proc. of 29th National Conf. of IPSJ*, Sep., 1984.

[Shibayama 84c] Shibayama, S. et al. Query Processing Flow on RDBM Delta's Functionally-Distributed Architecture, *Proc. of Int'l Conf. of Fifth Generation Computer Systems 1984*, Nov., 1984, and also *ICOT TR-064*.

[Taguchi 84] Taguchi, A. et al. INI : Internal Network in ICOT and its Future, *Proc. of 7th Int'l Conf. on Computer Communications*, Oct., 1984.

[Tanaka 82] Tanaka, Y. A Data Stream Database Machine with Large Capacity, *Proc. of Int'l Workshop on Database Machines*, Aug., 1982.

[Todd 78] Todd, S. Algorithm and Hardware for a merge Sort Using Multiple Processors, *IBM J. of Res. Develop.*, Vol.22, No.5, Sep., 1978.

[Uchida 82] Uchida, S. et al. The Personal Sequential Inference Machine Outline of Its Architecture and Hardware System, *ICOT Technical Memorandum TM-0001*, Nov., 1982.

[Yokota 83] Yokota, H. et al. An Investigation for Building a Knowledge Base Machine, *Proc. of 27th National Conf. of IPSJ*, Oct., 1983 (in Japanese), and also ICOT TM-0019. (in English)

[Yokota 84a] Yokota, H. et al. An Enhanced Inference Mechanism for Generating Relational Algebra Queries, *Proc. of 3rd ACM SIGACT-SIGMOD Symposium on Principles of database systems*, Apr., 1984, and also ICOT TR-026.

[Yokota 84b] Yokota, H. et al. Unification in a Knowledge Base Machine, *Proc. of 29th National Conf. of IPSJ*, Sep., 1984.

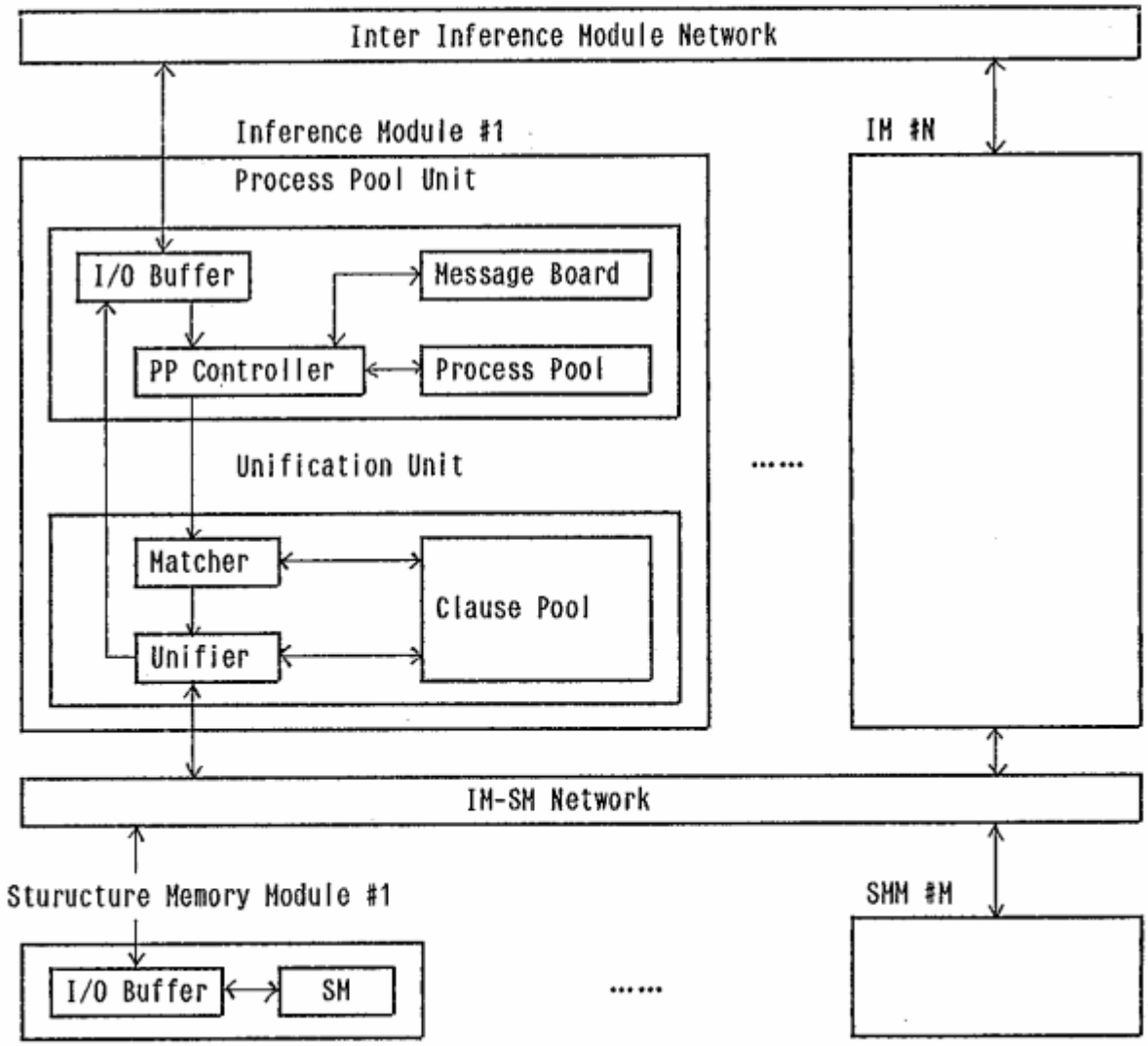


Figure 2.1 Conceptual Configuration of PIM-R

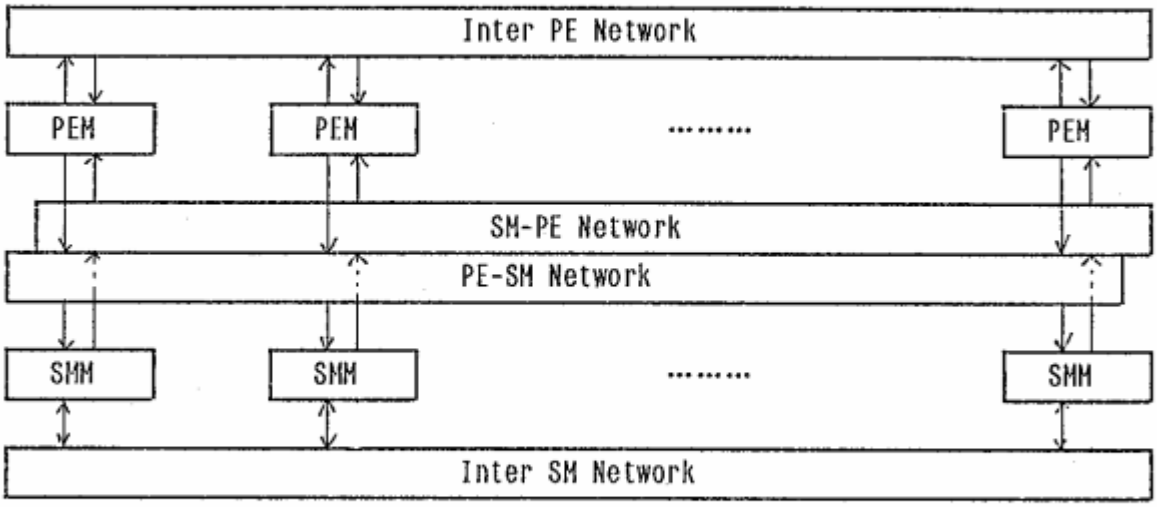


Fig. 2.2 Conceptual Configuration of PIM-D

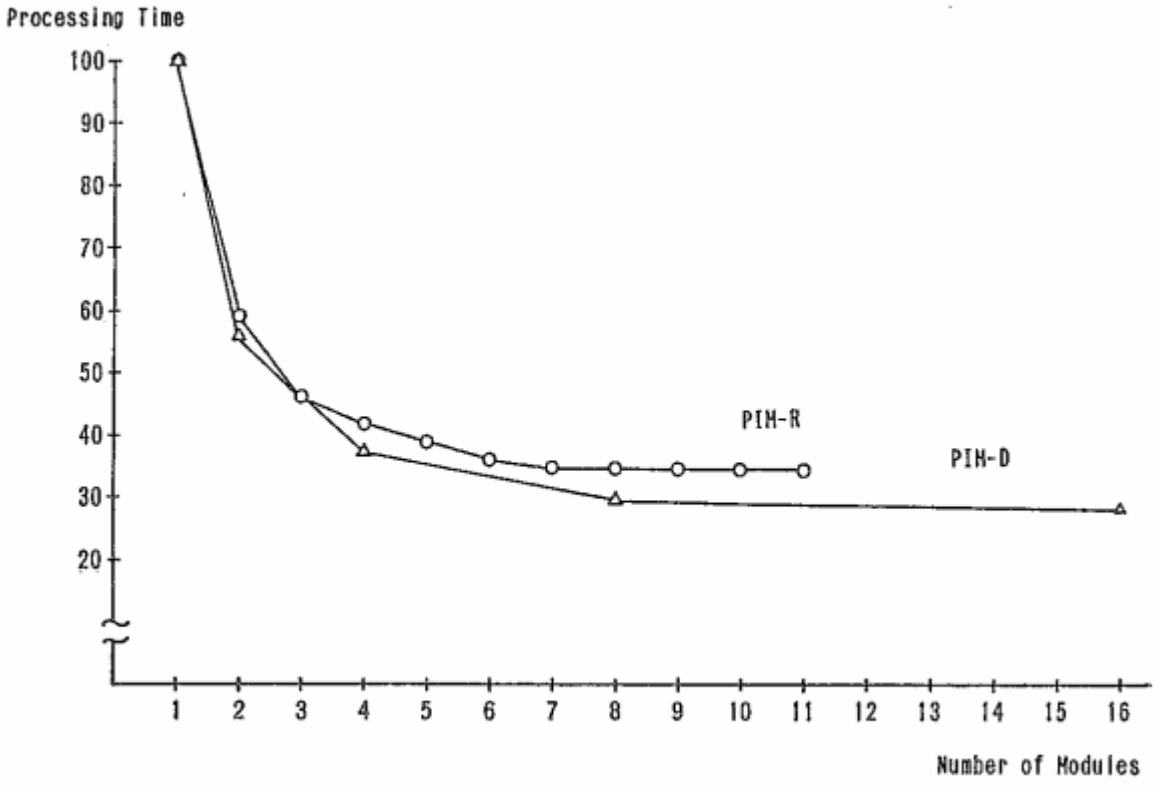


Fig. 2.3 Simulation Results of Four Queen Program

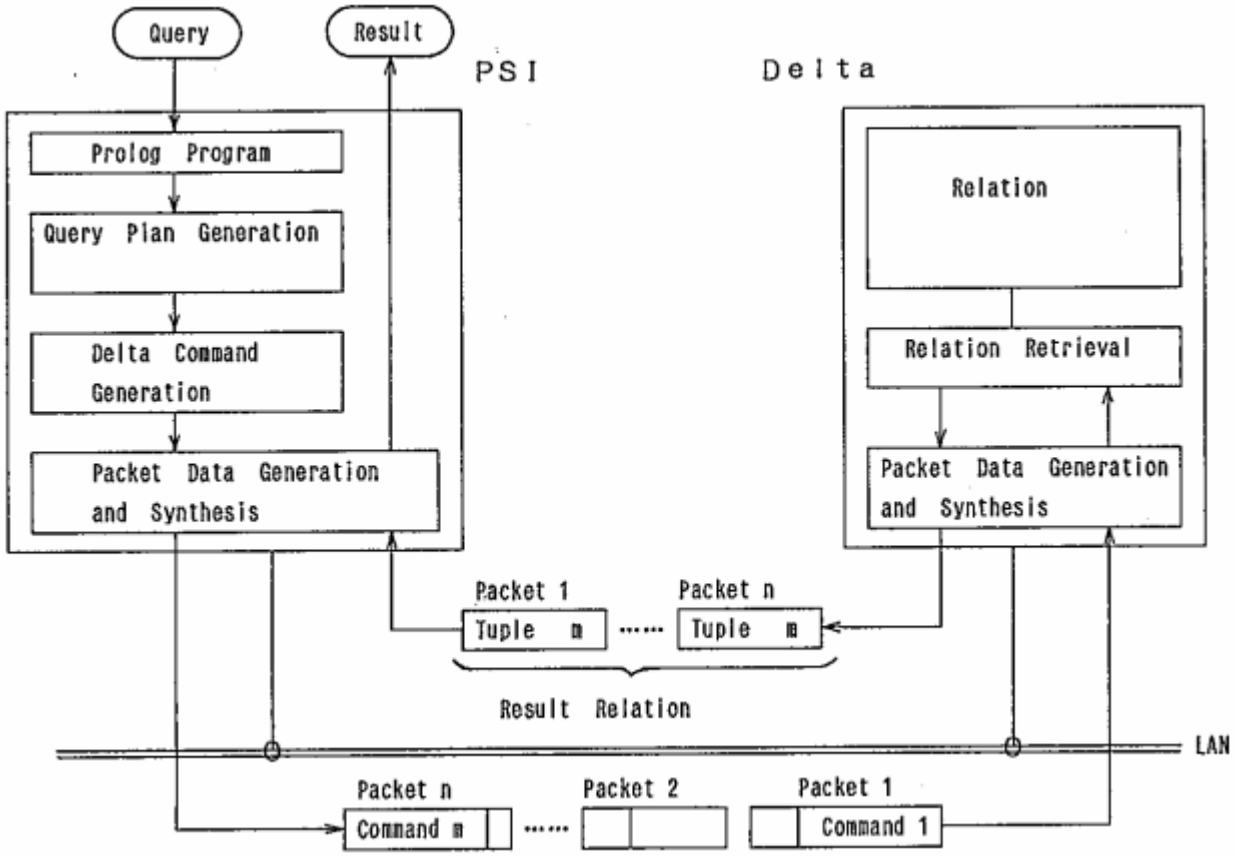


Fig. 3.1 Query Processing Flow

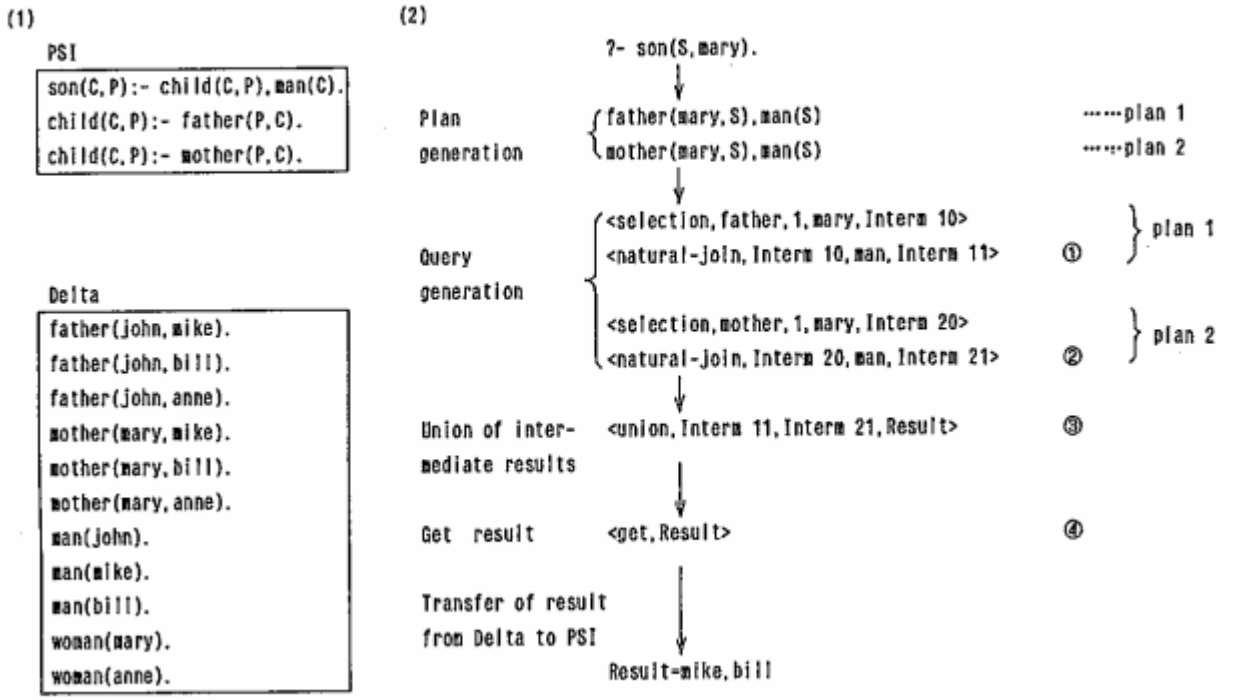
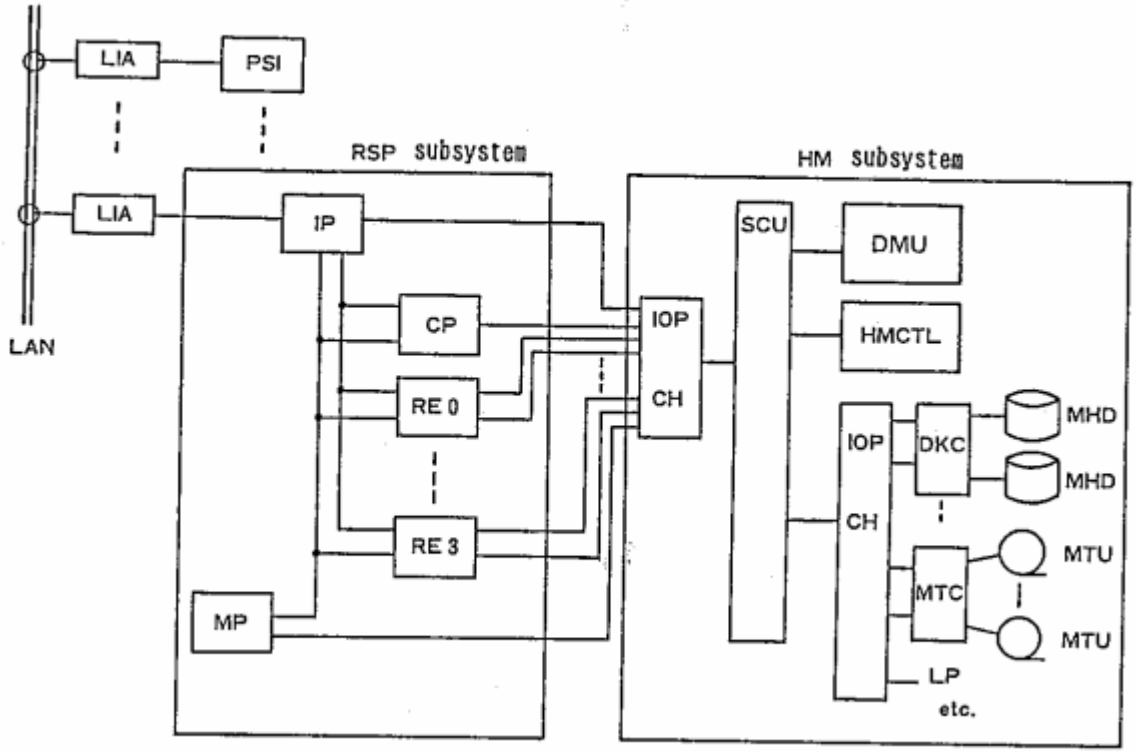


Fig. 3.2 Example of Query Processing



PSI: personal sequential inference machine, LIA: LAN interface adapter, RSP: RDBM supervisory and processing subsystem, IP: interface processor, CP: control processor, RE: relational database engine, MP: maintenance processor, HM:hierarchical memory subsystem, HMCTL: HM controller, DMU: database memory unit, IOP: I/O processor, SCU: storage control unit, MMD: moving head disk, DKC: disk controller, MTC: MT controller

Fig. 3.3 Delta Configuration

	Configuration of May,1984	Final configuration
(1) RSP subsystem		
Control Processor	1	1
Relational DB Engine	1	4
Interface Processor	1	1
Maintenance Processor	1	1
(2) HM subsystem		
HM Controller	1	1
DB Memory Unit	16 MB	128 MB
Moving Head Disks	2.5 GB ×2units	2.5 GB ×8units
Magnetic Tape Unit	2	4

Fig. 3.4 Delta System Specifications

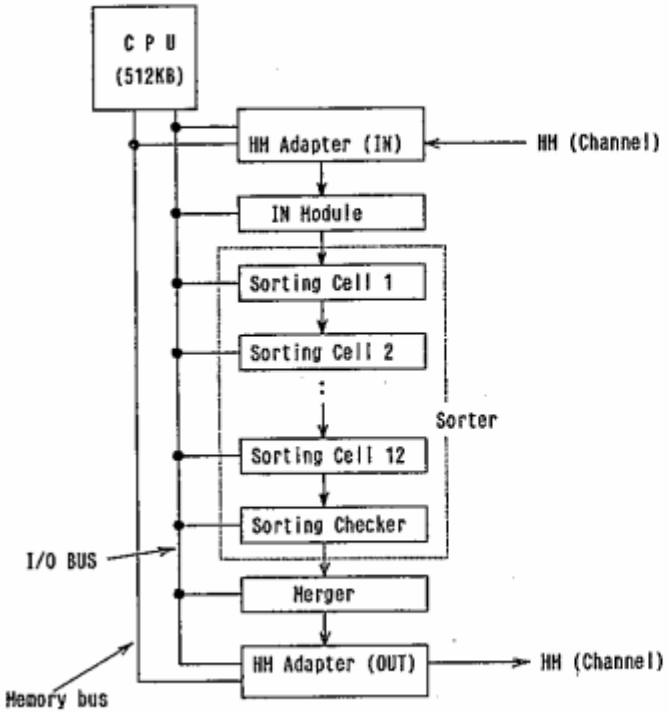
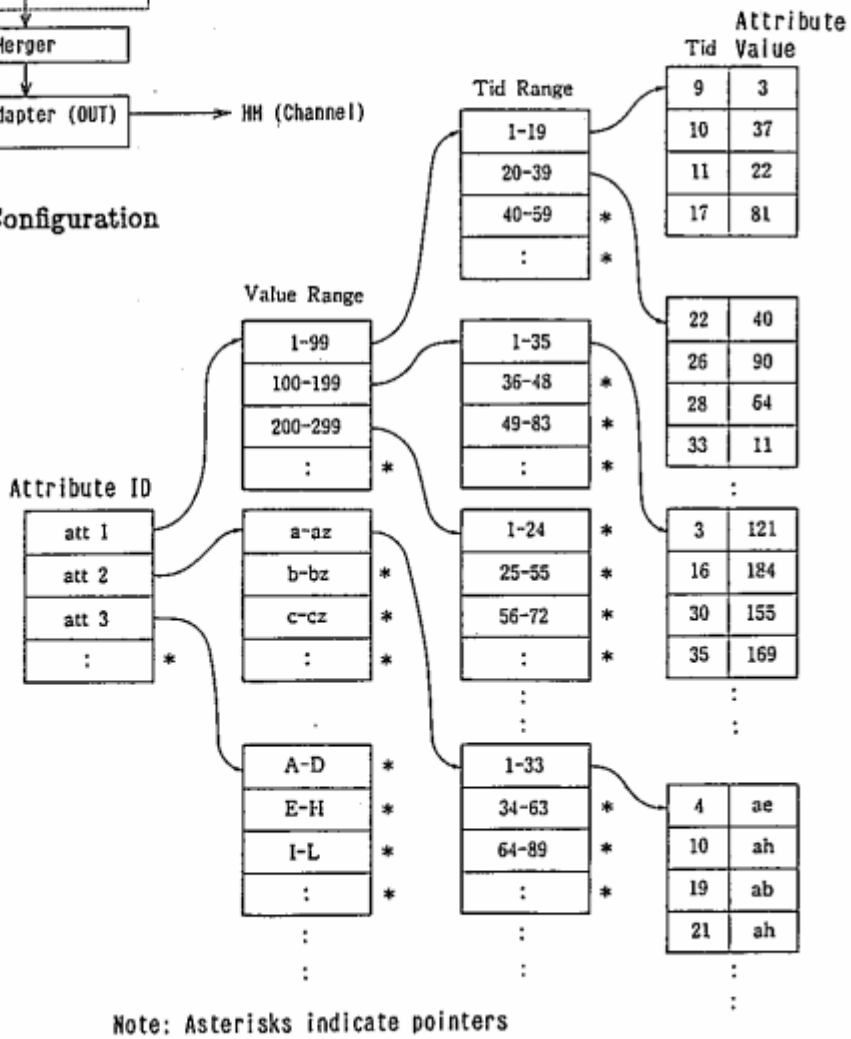


Fig. 3.5 RE Configuration



Note: Asterisks indicate pointers

Fig. 3.6 Internal Schema of Delta

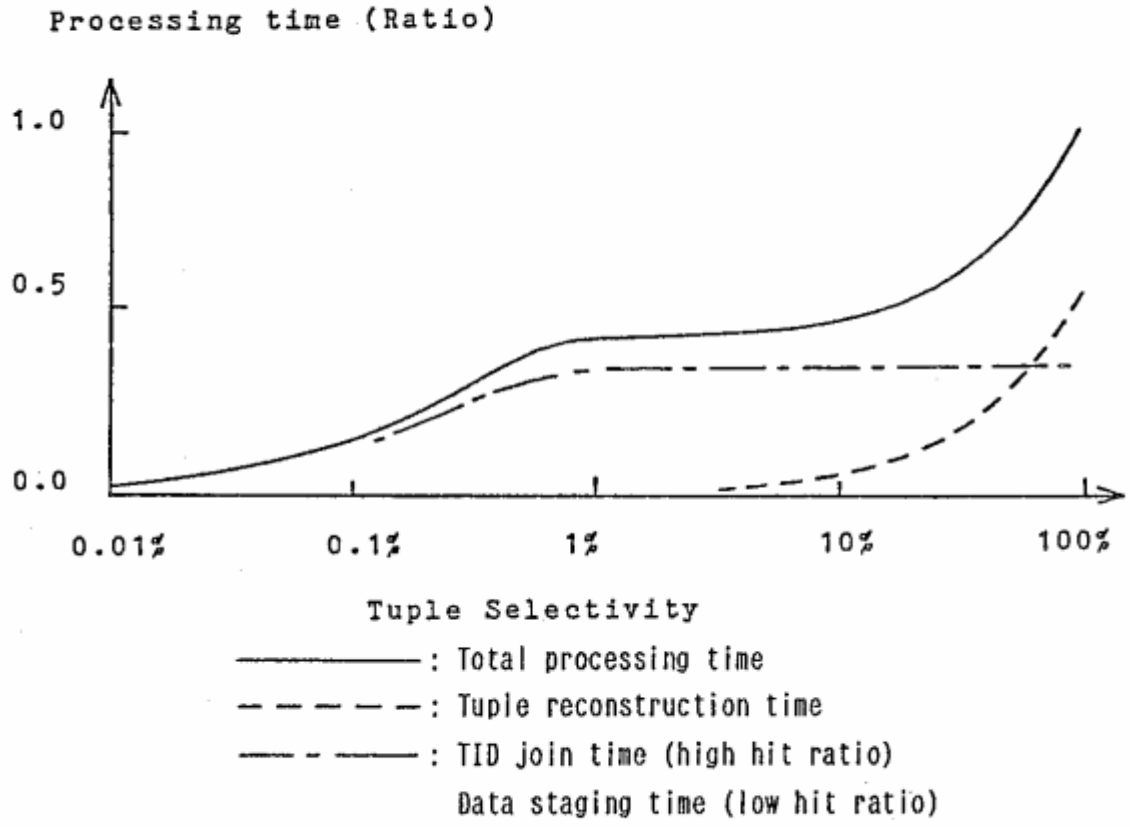


Fig. 3.7 Estimated Delta Performance Characteristics