ICOT Technical Report: TR-234

TR-234

Parallel Inference Machine and Knowledge Base Machine

by

H. Itoh and S. Uchida

February, 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

Parallel Inference Machine and Knowledge Base Machine

Hidenori Itoh and Shun-ichi Uchida

ICOT Research Center

Institute for New Generation Computer Technology

Mita Kokusai Building, 21F,

1-4-28 Mita, Minato-ku, Tokyo 108 Japan

03-456-2514

Abstract

This paper describes ICOT's parallel inference machine and knowledge base machine. A parallel logic programming language has been developed as a kernel language of the Fifth Generation Computer System (FGCS). Under the specification of the kernel language, hardware systems and software systems have been researched and developed. The personal sequential inference machines (PSI) has been developed as a hardware system; it is the tool for researching and developing the parallel inference machine (PIM) and knowledge base machine (KBM). The PIM operating system (PIMOS) and knowledge base management software system have been developed as basic software systems.

1. Introduction

Logic programming has begun to be used to formalize computer architecture, new programming styles, programming language semantics, and data base semantics. Logic programming will also play an important role in such fields as linguistics, artificial intelligence, and their parallel processing.

With this background, the FGCS project has been started with the conjecture that logic programming is the bridge connecting knowledge information processing and parallel computer architecture. Prolog was selected as the logic programming language to be used as a research tool [Fuchi 84 and 86]. From Prolog, a kernel languagse [Ueda 85] was developed corresponding to a conventional machine language. The kernel language is the nucleus of hardware and software systems for the Fifth Generation Computer System.

At present, a kernel language (KL) is being developed, which includes a parallel processing feature added to the logic language. Under the KL specifications, the parallel inference machine (PIM), the knowledge base machine (KBM) and their operating system (PIMOS) are being developed to establish parallel inference technology. Moreover the knowledge base management software system is being developed on the PIMOS.

2. Kernel Language

This section outlines some features of the kernel language. Various studies have been carried out to realize parallel programming in logic programming. Guarded Horn Clauses (GHC) [Ueda 85] was developed as the kernel language of the FGCS using the following approach:

Imposition of guards and restriction of nondeterminism where the clause which has passed the guard first is used for subsequent computation.

The pioneering works of GHC are Relational Language [Clark 81], Concurrent Prolog [Shapiro 83] and Parlog [Clark 84]. All of these languages are very similar and their common feature is that each

clause has a guard part and a body part, separated by commit operator (denoted by |). In these language GHC has the simplest syntax. Each clause written in GHC is in the following form:

H :- G1, ... , Gm | B1, ... ,Bn.

where connectives :- and , are common to ordinary Horn clauses. The part of a clause before | is called the guard, and the part after it is called the body. The calculation results of the guard are only effective in the guard. A guarded clause with no head is a goal clause, as in Prolog.

The semantics of GHC are also quite simple. The execution of a GHC program proceeds by reducing a given goal clause to the empty clause under the following rules:

Rule 1: Any piece of unification in the guard of a clause cannot instantiate a variable in the caller.

Rule 2: Any piece of unification in the body of a clause cannot instantiate a variable in the guard until that clause is selected for commitment.

Rule 3: When there are several clauses of the same head predicate (candidate clauses), the clause whose guard first succeeds is selected for commitment.

Rule 1 is used for synchronization, Rule 2 guarantees selection of one body for one invocation, and Rule 3 can be regarded as a sequencing rule for Rule 2. Under the above rules, each goal in a given goal clause is reduced to new goals (or null) in parallel.

3. KL Language System and PIMOS

KL is a hierarchical language system: KL-U (user), KL-C (core), KL-P (pragma), or KL-B (base) as shown in Figure 1. In an actual programming environment, where most system programmers are exected to

develop using KL-U, these four compiler systems are very important.

KL-C is a core language system, and is based on Flat-GHC with built-in predicates and meta-calls. Flat-GHC is a subset of GHC, whose guard goals are all built-in predicates. This restriction makes its language implementation more efficient in keeping its description power. Meta-calls are special functions enabling programmers to handle the logical values of goals.

KL-P is the attached language functions to KL-C such as job allocation and goal priority control. These functions exceed the logical framework of GHC, but are necessary to describe the operating system (PIMOS). PIMOS dynamically re-allocates processes considering their localities and load balance. Programmers should only assume the abstract image of PIM without a real processor network, which is a kind of homogeous processing power plane with logical distance of communications. These functions have been designed in KL-P for about 100 processing elements, and the control methods in KL-P for about 1000 processing elements have being studied with a software simulator.

KL-U is a high-level system programming language for system programmers. KL-C and KL-P specify the overall language functions of KL. Such functions are included in KL with some modular programming concepts. KL-U has been extended to be a parallel object oriented.

KL-B is a virtual machine code interfacing the PIM and PIMOS. It can be regarded as a compiler target language of KL-U, KL-C, and KL-P. KL-B also includes some special functions to control and maintain the PIM hardware directly.

4. Parallel Inference Machine Architecture

Figures 2 and 3 show the two overall construction images of PIM. Both machines have cluster concepts. There are about 10 processing elements in each cluster. A physical shared memory form a cluster in Figure 2, and a global address space is distributed in each processing element memory to form a cluster in Figure 3.

There are two types of clusters: logical and physical. The logical cluster is a group of processing elements that has one address space. These processing elements share data space such as parallel goal environments. Thus, some address transformation tables and their management are necessary for communication between logical clusters. Garbage can be collected in each logical cluster. The physical cluster is the group of processing elements which are physically connected closely. Each processor can communicate faster with other processors in its own cluster than with processors in another physical cluster.

In Figure 2, a cluster is a group of processors connected with the same shared memory, and the inter-processor network is a two-level network. These processors communicate with each other using both an intra-cluster network (i.e., a lower-level network) and their shared memory. The network response is more important than throughput for intra-cluster network communication. These clusters are connected with each other by an inter-cluster network (an upper-level network). High-throughput networks such as cross-bar networks are suitable and available because there are about 10 clusters in the PIM.

In Figure 3, a cluster is a group of processors whose global memories (GM) have the same address space. So each GM(i) in cluster(i) forms one global address space. The inter-processor network can be designed in a singl-level network. However, a two-level network is better to make the best use of the locality in application programs.

Other issues in designing processing elements are tag architecture for unification, efficient context-switching mechanisms, and interrupt handlers for inter-processor communication. Context-switching and interrupt handling are key issues in enabling communication between parallel goals. Context-switching occurs both in goal suspension and in unification with remote data. The concept of concurrent virtual machines has been introduced to realize efficient context-switching. The concurrent virtual machine mechanism can be regarded as a logical cache of goal contexts.

5. Knowledge Base Machine Architecture

The shared knowledge represented in the logic programming language is stored in the shared memory hardware system. It is also managed by a knowledge base management software system and retrieved by retrieve commands. This section describes the shared memory hardware system, pipelined-data stream type dedicated hardware engines, and their parallel control software system. The next section describes the knowledge base management software system and the retrieve commands.

For retrieval of the grand clauses that seem to be relational data, the relational engines (RE) and their parallel control methods have been developed and evaluated in the initial stage [Itoh 87].

For retrieval of the general clauses, the unification engine (UE) [Yokota 86] & [Morita 86] has been designed by enhancing the RE.

A generality on the clauses is defined in unification operation. Two sets of clauses are sorted in the generality stream in the UE as inputs. The UE manipulates them under relational-unification operation [Yokota 86] that is defined in sets of clauses. Figure 4 shoes the UE, it is composed of two sort units, a pair generation unit, and a unification unit.

This group is developing the multi-access mechanism that makes a number of UEs retrieve the same logical domain [Tanaka 84] simultaneously. This mechanism is composed of several ports and a switching network. The experimental machine, ports, and multi-UE parallel control software written in KL were developed as reported in [Itoh and Takewaki 86]. This experimental system has evaluated the relation of degree of parallelism, granularity of data, and availability ratio of UE. In the knowledge base the processing is assumed to be or-parallelism, allowing each UE to process without communication between common variables. Figure 5 shows the hardware configuration of the knowledge base machine. This knowledge base system is also available to the shared memory system in the parallel inference machine.

6. Knowledge Base Management Software System

The knowledge base management software system has such mechanisms as knowledge acqusition composed of knowledge assimilation, knowledge accomnodation, and knowledge equilibration [Kitakami and Kunifuji 84]. These mechanisms are supported by the basic inference functions: induction, reduction, and abduction realized with meta-programming technique.

For integrity-constraint checks, the knowledge base management software system needs to treat excepitonal relations or exceptional inheritance of knowledge under the closed world assumption. For efficiency of retrieval processing, rerieval command compiling and optimizing methods [Yokota 86] & [Miyazaki 86] were developed using the partial evaluation technique of logic programming [Takeuchi 85].

The problem-oriented knowledge representing languages used in frame, production, and other systems should be compiled in KL-C. The knowledge compiler (KC) was investigated for this pupose..

To construct these mechanisms on the PIMOS, techniques evaluated using the experimental PIM and KBM should be integrated into KL-U, KL-C, and KL-B.

7. Conclusion

This paper described the present status of research and development at ICOT on parallel inference machines and knowledge base machines.

Based on the technologies developed through this research, activities in this paper, this group will develop a prototype of a highly parallel computer for knowledge information processing, the ultimate aim of the project.

Acknowledgements

We would like to thank Dr. Fuchi, the director of ICOT, who contributed many useful suggestions for this research and development, and Dr. K. Furukawa and Mr. T. Yokoi for their valuable comments on this paper.

References

[Fuchi 84] K. Fuchi, "Revisiting the Original Philosophy of Fifth Generation Computer Systems Project"' Int. Conf. on Fifth Generation Computer Systems. Conf. Report pp. 18-25, 1984.

[Fuchi and Furukawa 86] K. Fuchi, K. Furukawa, "The Role of Logic Programming in the Fifth Generation Computer Project", International Logic Programming Conference, July 1986.

[Ueda 85] K. Ueda, "Guarded Horn Clauses", ICOT TR-103, 1985; Lecture Notes in Computer Science 221, Springer-Verlag 1986.

[Clark 81] K. L. Clark and S. Gregory, "A Relational Language for Parallel Programming", In Proc. on Functional Programming Languages and Computer Architecture, pp.171-178, ACM (1981).

[Clark 84] K. L. Clark and S. Gregory, "PARLOG: Parallel Programming in Logic", Research Report DOC 84/4, Dept. of Computing, Imperial College of Science and Technology, London (1984).

[Shapiro 83] E. Y. Shapiro, "A Subset of Concurrent Prolog and Its Interpreter, ICOT Technical Report TR-003, 1983.

[Goto and Uchida 86] A. Goto, S. Uchida, "Toward A High Performance Parallel Inference Machine -- The Intermediate Stage Plan of PIM --" ICOT Technical Report TR-201.

[Itoh 87] H.Itoh, et al., "Parallel Control Techniques for Dedicated Relational Database Engines", The Third International Conference on Data Engineering, IEEE, Feb. 1987.

[Yokota 86] H. Yokota, H. Itoh, "A Model and an Architecture for a Relational Knowledge Base", The 13th Annual International Symposium on Computer Architecture, June 1986.

[Morita 86] Y. Morita, et al., "Retrieval-by-Unification on a Relational Knowledge Base Model", Proceedings of the 12th International Conference on Very Large Data Base. Aug. 1986.

[Itoh and Takewaki 86] H. Itoh, T. Takewaki et al., "A Deductive Database System Written in GHC", ICOT TR-214 (1986).

[Tanaka 84] Y. Tanaka,"MPDC: Massive Parallel Architecture for Very Large Databases", Proc. of the Int. Conference on 5th Generation Computer Systems, pp. 113-137, Nov. 1984.

[Kitakami and Kunifuji 84] H. Kitakami, S. Kunifuji et al., "A Methodology for Implementation of Knowledge Acquisition System", Proceedings of the 1984 International Symposium on Logic Programming, pp. 131-142, Feb. 1984.

[Yokota 86] H. Yokota et al., "Deductive Database System on Unit Resolution", The Second Data Engineering Computer Conference, pp.228-235. Feb. 1986.

[Miyazaki 86] N. Miyazaki et al. "Compiling Horn Clause Queries in Deductive Database", ICOT Technical Report TR-183 (1986).

[Takeuchi 85] A. Takeuchi et al., "Partial Evaluation of Prolog Programs and Its Application to Meta Programming", ICOT Technical Report TR-126 (1985).

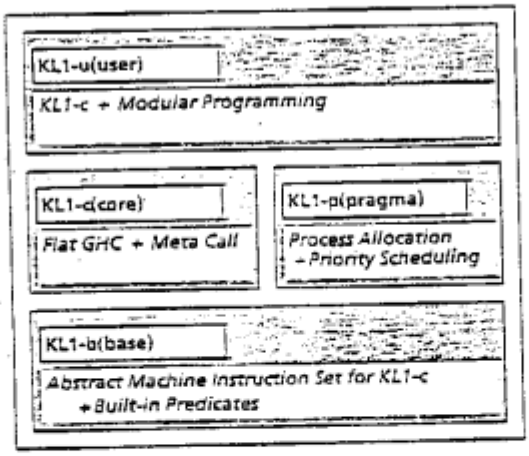


Figure 1 KL Language Systems

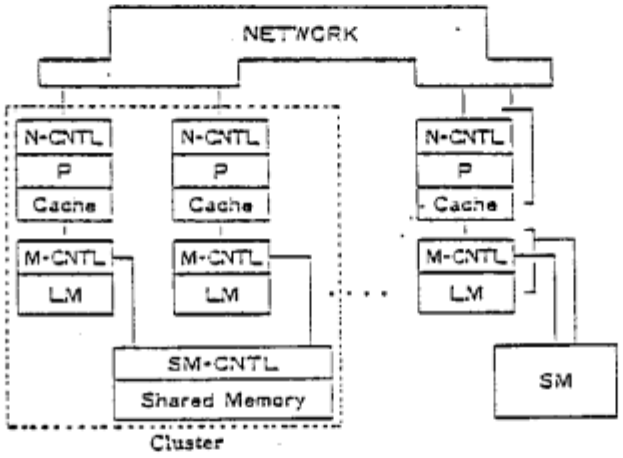


Figure 2 Hardware Configuration of PIM (1)

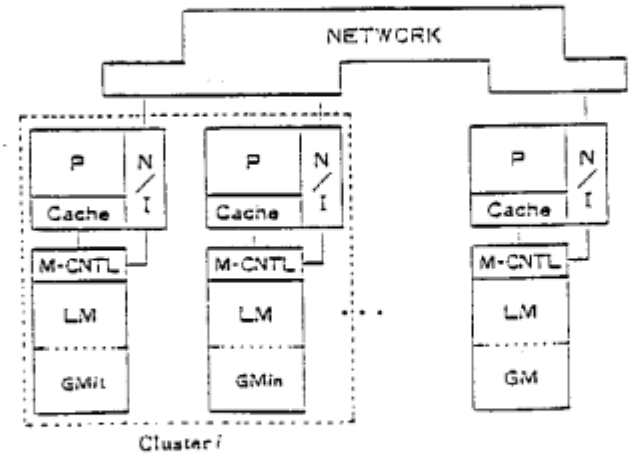


Figure 3 Hardware Configuration of PIM (2)

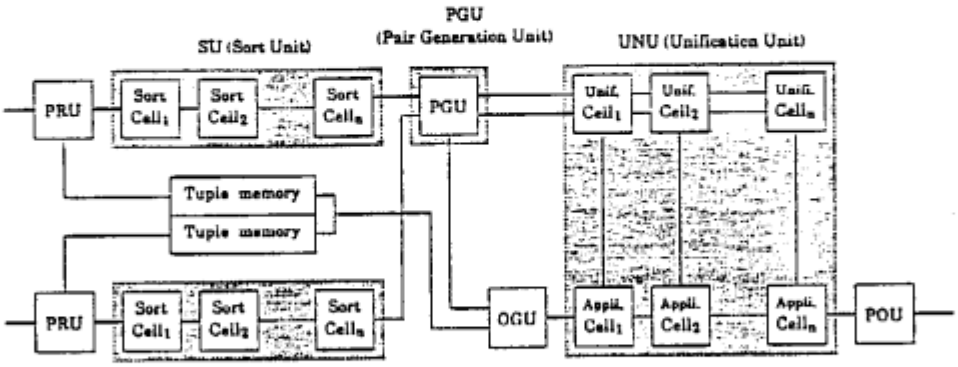
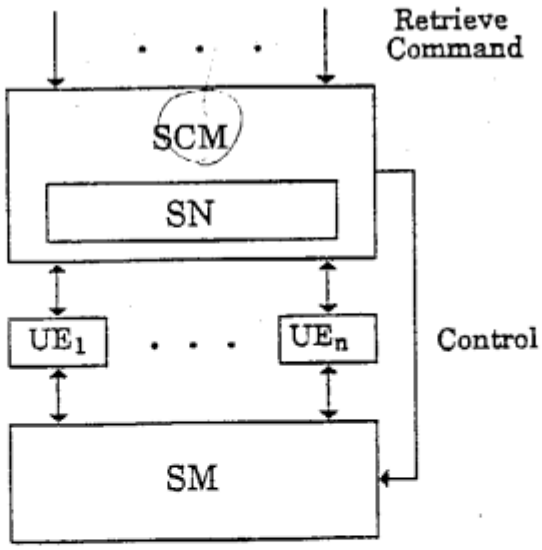


Figure 4 UE configuration



UE : Unification Engine
SN : Switching Network
SM : Shared Memory
SMC : Shared Memory Controller

Figure 5 Hardware Configuration of KBM