IXM2: A Parallel Associative Processor for Knowledge Processing

Tetsuya Higuchi[1], Hiroaki Kitano[2], Tatsumi Furuya[1],
Ken-ichi Handa[1], Naoto Takahashi[3], Akio Kokubu[1]

Electrotechnical Laboratory[1]
1-1-4 Umezono, Tsukuba,
Ibaraki, Japan 305

Carnegie Mellon University[2]
Center for Machine Translation
Pittsburgh, PA 15213
Email: higuchi@etl.go.jp

University of Tsukuba[3]
1-1-1 Tennodai, Tsukuba
Ibaraki, Japan 305

Abstract

This paper describes a parallel associative processor, IXM2, developed mainly for semantic network processing. IXM2 consists of 64 associative processors and 9 network processors, having a total of 256K words of associative memory. The large associative memory enables 65,536 semantic network nodes to be processed in parallel and reduces the order of algorithmic complexity to O(1) in basic semantic net operations. We claim that intensive use of associative memory provides far superior performance in carrying out the basic operations necessary for semantic network processing: intersection, marker-propagation, and arithmetic operations.

1 Introduction

In this paper, we propose a parallel associative memory processing architecture, and examine its performance superiorities over existing architectures for massively parallel machines. The parallel associative memory processing architecture is characterized by its intensive use of associative memory to obtain massive parallelism. The architecture is ideal for processing very large knowledge bases often represented by semantic networks. We have implemented the IXM2 associative memory processor based on our architecture in order to validate benefits of our architecture.

Several efforts are underway to develop a very large knowledge base (VLKB) which contains over a million concepts. MCC's CYC [Lenart and Guha, 1989] and EDR's electric dictionaries [EDR, 1990] are such examples. The basic framework of these knowledge-bases can be represented by semantic networks [Quillian, 1967]. While notable effort has been made to develop a sound theory on how to represent and develop VLKB, no significant investigation has been made on how to process VLKBs. The obvious problem of processing VLKB, as opposed to a small or medium size knowledge-base, is its computational cost. Even a simple operation to propagate markers through a certain link would require increasing computing time on serial machines as the size of the network grows. This also applies to three basic operations for processing semantic networks: (1) intersection search, (2) marker-propagation, and (3) arithmatic operations.

One obvious way out from this problem is the development of massively parallel machines. There are several massively parallel machines already developed, or currently being developed (SNAP [Moldovan, 1990], the Connection Machine [Hillis, 1985]). In general, these machines assume one node per processor type mapping of semantic network onto the hardware. The underlying assumption is that significant speed up can be obtained due to parallel computing by each processor in SIMD manner.

However, the pitfalls of this approach are that (1) processing within each processor is performed in a bit-serial manner, and (2) all marker-propagation must be done through communication links which is very slow. This implies that current architectures exhibit serious degradation of performance regardless of the fact that these operations look highly parallel for the user who observes the phenoema from outside of the processors. In scientific computings, especially in matrix computing, all PEs are always active and communictions are limited to neighbor PEs, thus it takes full advatange of SIMD parallelism. However, in the semantic network, although most processing can be carried out in a SIMD manner, not all PEs are activated all the time. Number of PEs active at a time vary during processing and it could range from a few PEs to thousands of PEs. Communication often need to be performed between distant PEs. Thus, a processing and communication capability of each PE significantly affects overall performance of the system. Unfortunately, for a machine with 1-bit PEs, bit-serial operations and communication hampers high performance processing.

In this paper, we propose a new approach to massively parallel computing in order to avoid the problem described above. Our approach is based on intensive use of large associative memories. The IXM2 is a machine built based on this paradigm. The IXM2 consists of 64 associative processors and 9 network processors. These are interconnected based on a complete connection scheme to improve marker propagation, and provides a 256K words of large associative memory. Using an associative memory of this size, IXM2 can perform the parallel processing of 65,536 semantic network nodes with a processing time of O(1) in basic operations, and only a minimum communication will be carried out between processors.

2 Problems of Current Massively Parallel Machines

The central problem which prevent current massively parallel machines from further performance improvement in AI applications is that all PEs are not always active. Number of active PEs at a time range from one to few thousands. Thus, performance bottleneck emerge in case when relatively small number of PEs are active and non-local message passings in irregular communication patterns are required. In such cases, the two characterestics of current massively parallel machines, (1) a bit-serial operation in each processor, and (2) bit-serial communication between processors, cause the degradation of performance. This is because current massively parallel machines assume tasks where most of 1 bit-PEs are highly utilized during the execution and the local and simultaneous communications among PEs are performed. In VLKB processing, marker propagation is especially tough in this respect. In addition, since one node in the semantic network is mapped to a single PE, any propagation of a marker must go through communication links which results in so called *hillis bottleneck*. This section reviews problems in current architectures in basic operations for processing semantic networks: (1) set intersection, (2) marker-propagation.

The set intersection is a very important operation in AI and is frequently performed to find a set of PEs with two properties. Although set intersection contains SIMD parallelism, there is a room for further improvement because a current architecture carrys out the intersection in a bit serial manner by each 1-bit PE.

Marker propagation, however, presents more serious problems. First, propagation of markers from a base node to N descendant nodes requires a sequential marker propagation operation to be carried out N times at the 1-bit PE of the base node. In addition, the serial link is very slow. Thus, as average fan-out increases, hence parallelism increases, current architecture suffers from severe degradation. Second, marker propagations are very slow because all the propagations are performed through the message passing communications among PEs. Message passing communications between PEs are slow due to the limited bandwidth of communication lines, the delays caused by intervening PEs in message passing, message collisions and so on. For these reasons, the marker propagation in Connection Machine is two orders of magnitude slower than SUN-4, Cray, and IXM2.

3 IXM2 Architecture

3.1 Design Philosophies behind IXM2

The use of associative memory is the key feature of the IXM2 architecture. Needless to say, IXM2 is a parallel machine which is constituted from a number of processors. However, the IXM2 attains massive parallelism by associative memory, not by processor itself. IXM2 has only 64 T800 transputers, each of which has a large (4K) associative memory [Ogura, 1989]. Instead of having thousands of 1-bit PE, IXM2 stores nodes of the semantic network in each associative memory. Because each associative memory can store up to 1,024 nodes, IXM2 can store 65,536 nodes in total.

Four major issues have been addressed in the design decision to use associative memory in the IXM2: (1) attainment of parallelism in operations on each node, (2) minimization of the communication bottleneck, (3) powerful computing capability in each node, and (4) parallel marker propagation. The term *node* refers to a node in the semantic network, not necessary that of a single PE.

By allocating nodes of semantic networks on associative memories, association and set intersections can be performed in O(1). Because of the bit-parallel processing on each word of associative memory, these operations can be performed faster than 1-bit PE.

Multiple nodes allocation on a PE offers a significant advantage in minimizing communication bottleneck. Because a large number of nodes can be loaded in a PE, propagation of markers from a node to other node on the same PE can be done not by communication between PEs, but by memory reference. In addition, to deal with marker propagation between PEs, IXM2 employs a full-connection in which all processors in a cluster are directly connected. Thus, it decreases the time required for communication between PEs more than other connection models such as N-cube or torus. Specifically, the full-connection minimizes the chance of collision and message-interleaving PEs.

Associative memory has extremely powerful logical and arithmetic computing power. When an operand is stored in each word of the associative memory, a bit-serial operation can be executed to all words in parallel. Massive parallelism offered by this mechanism provides nano seconds order execution time per datum.

By use of the parallel write and the search functions of associative memory, multiple marker propagation from a node can be done in parallel, independent of the number of fan-outs (O(1)). We call this powerful feature *parallel marker propagation*.

3.2 Overall structure

IXM2 contains 64 associative processors (AP) and 9 network processors (NP) for communication. Figure 1 shows an external view of the IXM2 (left), and its structure (right). Eight APs and one NP form a processing module (PM), where eight APs are completely interconnected. In recursive fashion, eight processing modules are also interconnected each other and are connected to one NP which has the connection with the host SUN-3. IXM2 works as an AI co-processor. The technical summary of IXM2 is described in the appendix.

IXM2 employs complete connections to speed-up marker propagation among APs. Marker propagations between two APs have to be done as much as possible between APs which are directly connected; message path distance in marker propagation must be kept as close to

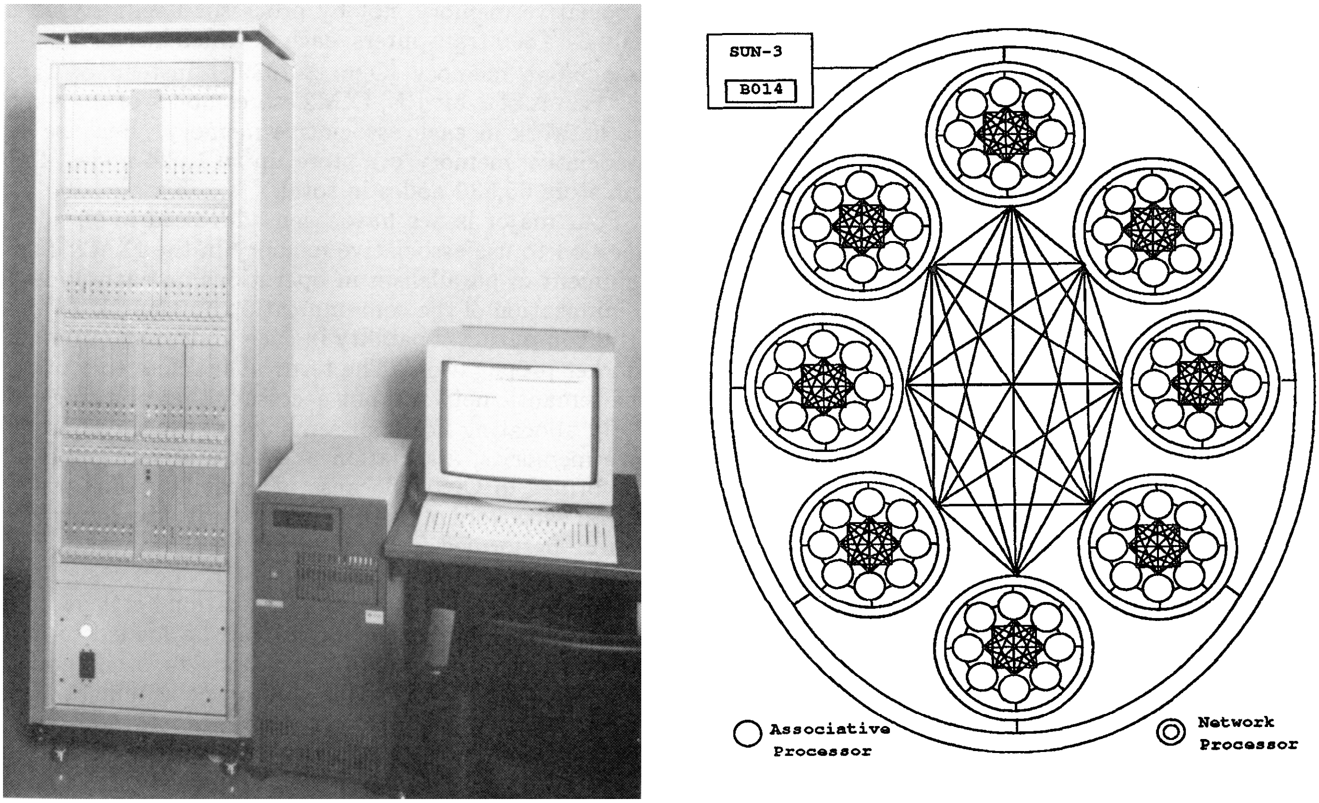


Figure 1: The IXM2: An External View and its Structure

Table 1: Average message path distance for IXM2 and other interconnections

No. of PE	IXM2	hypercube	torus
64	2.77	3.04	4.03

1 as possible. Although it is almost impossible to establish a complete connection among 64 APs, complete connection among a smaller number of APs is possible. Eight APs are selected as a unit of complete connection due to the implementation requirement. Furthermore, it is possible to keep the communication locality in marker propagation within these 8 APs. It is known that a large semantic network can be divided into sub semantic networks, with dense connectivity among the nodes of a given sub-semantic network, but with relatively few connections to the outside sub-semantic network [Fahlman, 1979].

Even in a worst case scenario, where communication locality within a PM is difficult or impossible to maintain (i.e. marker passings occur between any pair of the 64 APs), the average message path distance in the IXM2 interconnection can be kept smaller than that of hypercube and torus, as shown in Table 1. There, marker passings are assumed to occur between each AP and all other APs in the system and then the average length of each message path distance is calculated.

The programming language for IXM2 is the knowledge representation language IXL [Handa, 1986]. In IXM2, data and programs for semantic net processing are allocated as follows:
(1) A large semantic network to be processed by IXM2 is partitioned into semantic sub-networks and stored in the associative memory of each AP.
(2) Subroutines to execute IXL commands[1] are stored in the local memory of each AP.

The network processors broadcast an IXL command simultaneously to all the APs. NPs also accept results from each AP and pass them back to the host computer for the IXM2 (SUN-3/260).

An IMS B014 transputer board is installed in the SUN-3 to control the IXM2, load occam2 programs into the IXM2, collect answers returned from the IXM2, handle errors, and so on.

4 Parallel Processing Using Associative Memories

This section describes how parallel processing is performed on an associative memory. We begin by describing the data representation of a semantic network. Then parallel marker propagation and set intersection are described.

[1] IXL commands are predicates defined for semantic network processing.

4.1 Representation of Semantic Network

The semantic network representational scheme in IXM2 is strongly node-based. Each node stores information in both associative memory and RAM.

Node information stored in associative memory is intended to be processed with the massive parallelism provided by large associative memory (256K words). By this means, the times for association, set intersection and marker propagation operations can be reduced to O(1).

The node information in associative memory comprises; (1) A marker bit field (28 bits), (2) A link field (8 bits), (3) A parallel marker propagation identifier (abbreviated as PID; 22 bits), and (4) A literal field (16 bits).

The marker bit field stores the results of processing and is used just like a register in microprocessors. There are 28 marker bits in the current implementation.

The link field consists of 8 bits; each bit indicates the existence of a primitive link through which the node is connected to other nodes. The four types of primitive links are defined in IXM2 to support basic inference mechanisms in the knowledge representation language IXL which is an extended Prolog for IXM2. The primitive links are *isa*, *instance-of* (iso), *destination* (des), and *source* (soc) link. Because the direction of a primitive link must be distinguished, there are 8 bits in a link field; from the most significant bit (MSB), they are *risa, isa, riso, iso, rdes, des, rsoc* and *soc.* 'r' signifies an inverse link. If a node is pointed to by an *isa* link, the node has a *risa* link and the MSB of the link field becomes 1.

The literal field is prepared for a node which is itself a value and is processed by algorithms which exploit the massive parallelism of large associative memories.

On the other hand, the following node information is kept in RAM; (1) destination nodes from the node (classified according to the link type), (2) parallel marker propagation identifiers (PID), (3) search masks for parallel marker propagation (PMASK). PID and PMASK are the information for parallel marker propagation and are classified according to the 8 link types.

Figure 2 shows an example of the representation of a semantic network. The node C points to the A node via an *isa* link, and the link field of node C has '01000000'. This is because the position of '1' in the link field represents that the C node has out-going *isa* links. The destination field on RAM area has 'A' in the isa part, because the destination of the C node connected by an *isa* link is the A node.

4.2 Marker Propagation

Marker propagation in IXM2 is performed either by a sequential marker propagation or by a parallel marker propagation.

A sequential marker propagation is performed by message passing either within an associative processor or among associative processors, using the destination information stored in the RAM area.

Parallel marker propagation is performed within an associative processor; it can perform multiple marker propagations from a large fan-out node in parallel. (In addition to this parallelism within one AP, parallelism among 64 APs is available.) The rest of this section describes how the parallel marker propagation is performed.

In the network example in Figure 2, marker propagation from node A to C, D and E can be performed using parallel marker propagation. We call the node A a base node and nodes C, D and E descendant nodes.

The basic idea of parallel marker propagation is to search descendant nodes and write a particular marker bit into them by use of associative memory; the search and parallel write functions are used. Specifically to, (1) assign an identifier to all the descendent nodes, (2) assign the same identifier to the base node, and (3) (at the base node) issue the search operation with the identifier to find descendants nodes, and set a new marker bits in parallel into descendent nodes just searched.

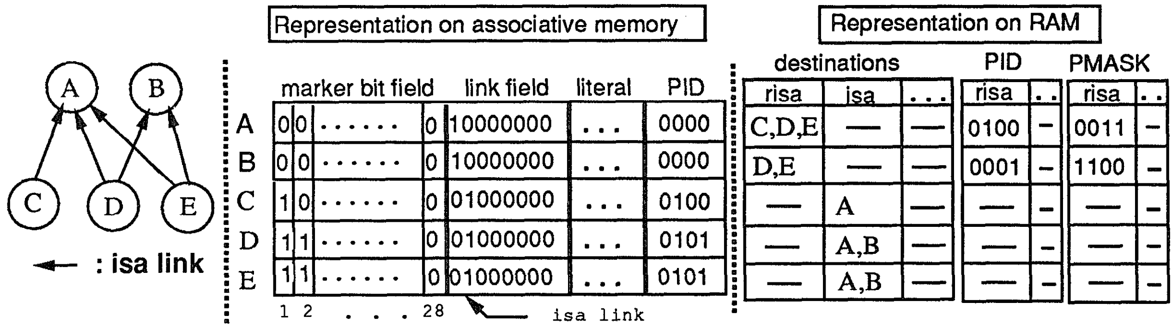
The identifier in (1) and (2) is a parallel marker propagation identifier (PID) described earlier. This is provided beforehand by the allocator, and loaded with the network. In the search in (3), a search mask (PMASK) defined before is used to search only for the bits satisfying the matching.

Using this method, parallel marker propagation is performed in Figure 2 as follows. Suppose parallel marker propagation is to be performed from the A node to C, D and E nodes. At first, the PID and the PMASK are retrieved from the RAM area for the A node: '0100' for the PID and '0011' for the PMASK. They are set into the search data register (SDR) and the search mask register (SMR)of the associative memory respectively, as shown in Figure 3 (b); the bits of the dotted area in the search mask register are all one and the search for those bits is disabled. Next the search operation is executed; the words for C, D and E node are hit by this search. Finally, the parallel write operation, which is a function of the associative memory, is performed to set a marker bit 1 at the same time in each of the three words just searched. Similarly, marker bit 2 of members of set B (nodes D and E) are written using parallel marker propagation.

The data for parallel marker propagations such as PID and PMASK are prepared by the semantic network allocator. The allocator recognizes pairs of { a base node, link type, descendent nodes } and gives to each pair a unique identifier (PID) and a search mask (PMASK). The recognition of such pairs is based on the number of the fan-out which is given to the allocator as a parameter. If the parameter is N, only the nodes with more than N fan-out are recognized as the candidates of the parallel marker propagation. For example, in Figure 2, N is 2 and two pairs are recognized: { A, risa, (C, D, E) } and { B, risa, (D, E) }. The number of pairs recognized can be controlled by changing the value of N, although PID has an enough length of 22 bits for 1000 nodes on each PE and so it is unlikely that these 22 bits get used up.

4.3 Set Intersection

To obtain the intersection of sets A and B, we only have to search those words at which both marker bits 1 and 2 are set. Figure 3 (a) shows the status of associative

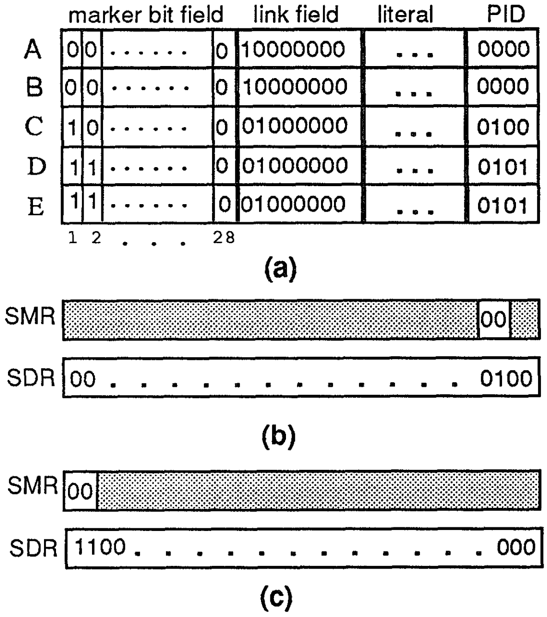


	marker bit field				link field	literal	PID
	1	2	. . .	28			
A	0	0	······	0	10000000	...	0000
B	0	0	······	0	10000000	...	0000
C	1	0	······	0	01000000	...	0100
D	1	1	······	0	01000000	...	0101
E	1	1	······	0	01000000	...	0101

destinations			PID		PMASK	
risa	isa	. . .	risa	. .	risa	. .
C,D,E	—	—	0100	-	0011	-
D,E	—	—	0001	-	1100	-
—	A	—	—	-	—	-
—	A,B	—	—	-	—	-
—	A,B	—	—	-	—	-

Figure 2: Data representation of semantic network

memory after two parallel marker propagations. By setting the search mask and the search data registers as in Figure 3 (c), intersection can be found in one search operation. Then, a parallel write operation can be performed to set marker bit 3 of the D and the E nodes. Thus, set intersection can be done in O(1).



	marker bit field				link field	literal	PID
A	0	0	······	0	10000000	...	0000
B	0	0	······	0	10000000	...	0000
C	1	0	······	0	01000000	...	0100
D	1	1	······	0	01000000	...	0101
E	1	1	······	0	01000000	...	0101
	1	2	. . .	28			

(a)

SMR 00

SDR 00 0100

(b)

SMR 00

SDR 1100 000

(c)

Figure 3: Set intersection with associative memory

5 Performance Evaluation

This section discusses the performance of IXM2 in two contexts: basic operations and applications. The IXM2 performance is compared with results from other high performance machines such as the Connection Machine (CM-2), the Cray X-MP and the SUN-4/330.

The figures for IXM2 shown below were measured using IXM2 of which clock speed is 17.5 MHz. In the programs for IXM2, IXM machine instructions are written in occam2. Programs for Cray and SUN are written in C and are optimized with -O4. Programs for CM-2 are written in C* and are optimized. Programs are executed on CM-2 of which PE clock is 7.0 MHz. Execution times have been measured using timers on CM-2 and they show CM busy time, excluding host interaction timings.

Table 2: Execution times of set intersection (μs)

set size	1000	10000	64000
IXM2	18	18	18
CM2	103	103	103
Cray X-MP	1829	7372	39823
SUN-4/330	28998	44332	142827

5.1 Basic operations

5.1.1 Association and set intersection

Because of the bit-parallel processing in the associative memory, the execution time of the association operation on IXM2 is always 12 μs for any amount of data up to 64K. It is also possible on sequential machines to implement O(1) association time using hashing or indexing.

However, set intersection for large data is very time consuming on sequential machines. Table 2 shows the performance of set intersection on IXM2, CM-2, Cray X-MP and SUN-4/330, where two sets of the same size are intersected.

IXM2 can consistently perform the set intersection in 18 μs for any size of data up to 64K; the set intersection is performed in O(1). Although CM-2 can also perform the set intersection constantly in 103 μs, IXM2 is faster because of bit-parallel associative memory.

Sequential computers become very slow as the data grows larger. It is true even for the Cray, in spite of the indexing algorithm of O(N). Although the Cray is much faster than the SUN-4 because of the vectorization available in the algorithm, there is a difference of three orders of magnitude between IXM2 and Cray in the processing of 64K data.

5.1.2 Marker propagation

Next we compare the performance of marker propagation. First, we compare the time to complete propagation of markers from one node to all descending nodes. The left chart of Figure 4 shows performance by each machine with different fanout from the node. IXM2 is outperformed when only one link exists from the node. However, if an average fanout is over 1.75, IXM2 outperforms the Cray and the SUN-4.[2] If an average fanout is nearly 1, using a parallel machine is not a rational decision in the first place. It should be noticed that IXM2 completes propagation at a constant time due to parallel marker-passing capability with associative memory. The parallel marker propagation by one AP constantly takes 35 μs, independent of the number of descendent nodes N. On serial machines (Cray and SUN-4), computational time increases linearly to the number of fanouts. CM-2 also requies more linear time as fanout increases. This is due to its serial link constraints that markers for each descending node has to be send in a serial manner. As we have discussed in section 2, CM-2 does not gain advantage of parallelism at each processor. Thus, if average fanout is over 1.75, IXM2 will provide a faster marker-propagation than any other machines.

The right chart of Figure 4 shows performance against parallel activation of marker-propagation. We used a network with 1000 nodes with fanout of 10. By parallel activation, we mean that more than one node is simultaneously activated as a source of marker-propagation. On the X-axis, we show a level of parallelism. Parallelism 1,000 means that markers are propagated from 1,000 different nodes at a time. Time measured is a time to complete all propagations. Obviously, serial machines degrade linearly to parallelism. IXM2 shows similar linear degradation, but with much less coefficient. This is because IXM2 needs to fire nodes sequentially at each T800 processor. The data in this graph is based on one AP out of 64 APs. Thus, when all 64 APs are used, the performance improves nearly 64 times. CM-2 has almost constant performance because all nodes can simultaneously start propagation. It is important, however, to notice that CM-2 outperforms only when the parallelism exceeds certain level (about 170 in this example), in case only one AP of the IXM2 is used. This would be equivalent to 10,880 with 64 APs. This implies that if applications do not require more than 10,880 simultaneous marker-propagation, IXM2 is a better choice than CM-2. This trade-off point, however, changes as average fanout changes.

5.1.3 Arithmetic and logical operations

Node information can contain the literal field in associative memory when a node is itself a value. This literal field can be processed with bit-serial algorithms for associative memory [Foster, 1976]. Execution time is constant, independent of the number of data items. Therefore, the execution time per item becomes extremely fast if the number of data items stored in associative memory is large.

The *less than* operation takes on average 36 μs for the comparison of 32-bit data. This seems quite slow when compared with the execution time on sequential computers; for example, it takes 1.25 μs in an occam2 program run on a T800 transputer at 20 MHz. However, it corresponds to an execution time per datum of 0.56 ns (nano second) when each of 64 K nodes contains a literal field and is processed in parallel. Although the number of data items available as candidates for the processing is application dependent, the associative memory algorithm will surpass the performance on sequential computers if there are at least 100 candidates. The additions for 8-bit data and 16-bit data take 46 μs (0.72 nano second per datum) and 115 μs (1.80 nano second per datum) respectively.

Table 3: Query processing time (milli sec.)

IXM2	CM-2	SUN-4/330
0.8	28.5	3.0

5.2 Application

Two applications have been developed so far on IXM2: a French wine query system, and memory-based natural language processing system [Kitano and Higuchi, 1991].

The wine knowledge base consists of 277 nodes and 632 links. The sample query elicits wines which belong to Bordeaux wine with the ranking of 4 stars. Table 3 shows the results on IXM2, CM-2 and SUN-4/330. Although the network is relatively small to take advantage of parallelism, IXM2 performs better than other machines. Maximum parallel activation is 87 in this knowledge-base, so that CM-2 is much slower than IXM2. Plus, over 95% of computing time in CM-2 was spent on communication to propagate markers between PEs.

In the natural language processing task, a network to cover 405 words and entire corpus has been created. The inputs starts from phoneme sequence so that speech input can be handled. The network has relatively large fanout (40.6). Figure 5 shows performance of the IXM2, the SUN-4, and the CM-2. Due to a large fanout factor, IXM2 far surpasses processing speed of other machines (SUN-4 and CM-2). SUN-4 is slow because set intersections are heavily used in this application.

6 Discussions

First, drastic difference of performance in set operation between serial processors (SUN-4 and Cray) and SIMD parallel processors (CM-2 and IXM2) rules out the possibility of serial machines to be used for a large scale semantic network processing in which extensive set operations involving over 1,000 nodes are anticipated.

Second, performance comparison in marker propagation indicates that IXM2 exhibits superior performance than CM-2 for many AI applications. IXM2 is consistently faster for processing semantic network with large fanout, but limited simultaneous node activation. When the average fanout is large, IXM2 have advantage over CM-2, and CM-2 gains benefits when large simultaneous activations of nodes take place. Let F and N be an average fanout, and number of simultaneously activated nodes in the given task and network. IXM2 outperforms

[2] The reason why Cray is slower than SUN-4 in marker propagation is considered to be the overhead of recursive procedure calls used in link traverse. Another Cray (Y-MP) was also slower.

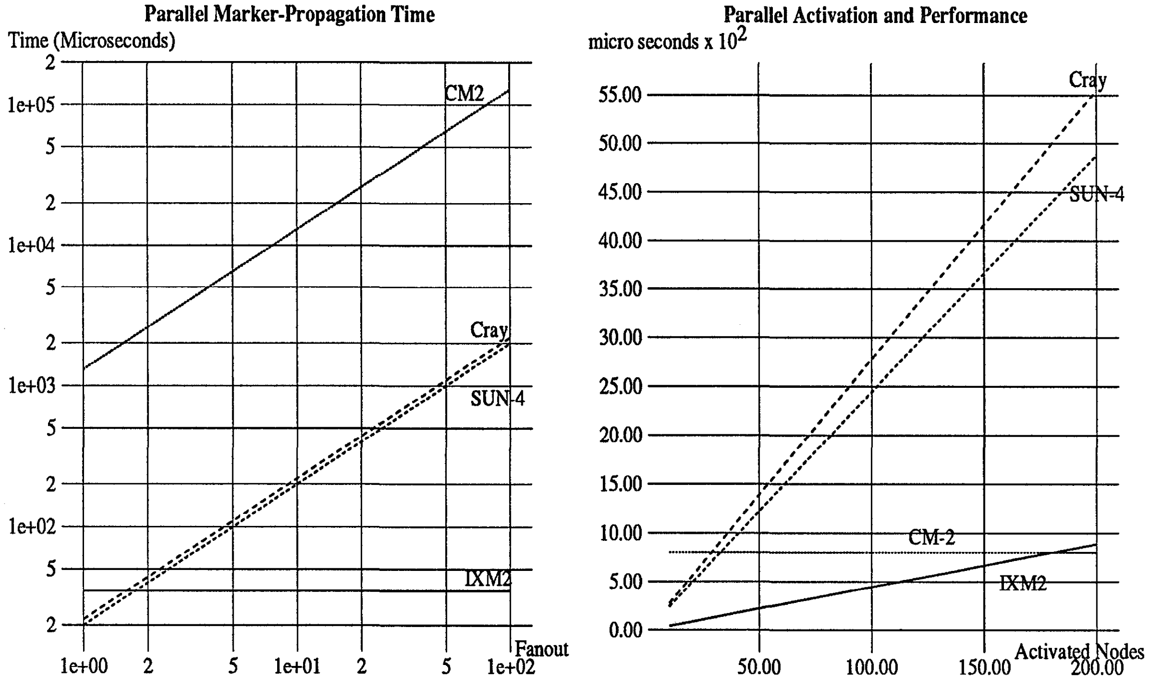


Figure 4: Marker Propagation Time

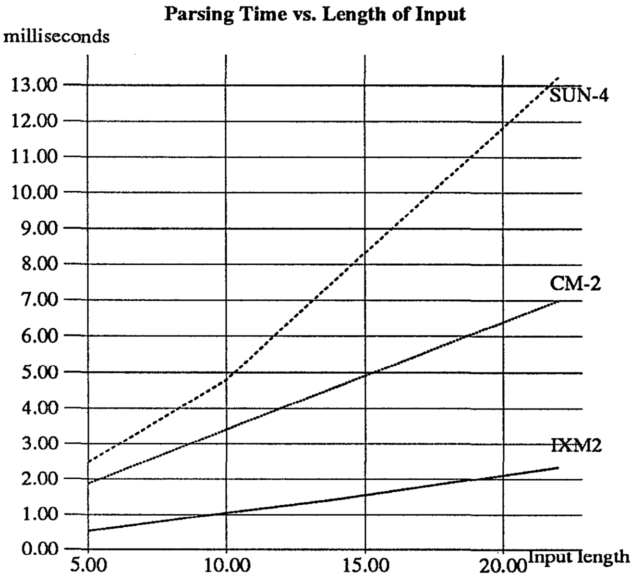


Figure 5: Parsing Time vs. Input String Length

CM-2 when the following equation stands:

$$\frac{T_{CMlink}}{T_{IXMnode}} > \frac{N}{F \times AP} \qquad (1)$$

AP is a number of APs used in IXM2 which ranges from 1 to 64. T_{CMlink} is a time required for the CM-2 to propagate marker for one link from a node. $T_{IXMnode}$ is a time required for the IXM2 to propagate markers from one node to all descending nodes. In this experiment, T_{CMlink} was 800 micro second[3], and $T_{IXMnode}$ was 35 micro seconds. Of course, this value changes as system's clock, software, compiler optimization and other factors change.

Average fanout of actual applications which we have examined, wine data-base and natural language, were 2.8 and 40.6, respectively. In order for the CM-2 to outperform IXM2, there must be always more than 4097 and 59392 simultaneous activation of nodes, respectively. Notice that it is the "average number" of simultaneous activation, not the peak number.

In the wine data-base, maximum possible parallel activation is 87. In this case, all terminal nodes in the network simultaneously propagate markers. For this task, obviously IXM2 outperforms CM-2. In the natural language processing, maximum parallelism is 405 in which all words are activated (which is not realistic). Since syntactic, semantic, and pragmatic restriction will be imposed on actual natural language processing, the number of simultaneous marker-propagations would be a magnitude smaller than this figure. Thus, in either case, IXM2 is expected to outperform CM-2, and this has been supported from the experimental results on these applications.

Although the average parallelism throughout the execution would be different in each application domain, it is unlikely that such a large numbers of nodes (4.1K and 59.4K nodes in our examples) continue to simultaneously propagate markers throughout the execution of the application. In addition, when such a large number

[3] This value is the best value obtained in our experiments. Other values we obtained include 3,000 micro second per link.

of nodes simultaneously propagate markers, a communication bottleneck would be so massive that performance of the CM-2 would be far less efficient than speculated from the data in the previous section.

In addition, there are other operations such as a set operation, and logic and arithmetic operations in which IXM2 has magnitude of performance advantages. Thus, in most AI applications in which processing of semantic networks is required, IXM2 is expected to outperform other machines available today.

7 Conclusion

In this paper, we proposed and examined the IXM2 architecture. Most salient features of the IXM2 architecture are (1) an extensive use of associative memory to attain parallelism, and (2) full connection architecture. Particularly, the use of associative memory provides the IXM2 with a truly parallel operation in each node, and nano-seconds order logical and arithmetic operations.

We have evaluated the performance of the IXM2 associative processor using three basic operations necessary for semantic network processing: intersection search, parallel marker-propagation, and logical and arithmetic operations. Summary of results are:

- Association operations and set intersections can be performed in O(1). IXM2 attains high performance due to bit-parallel processing on each associative memory word.
- Parallel marker propagations from a large fanout node can be performed in O(1) through the use of associative memory, while marker propagation implemented on a sequential computer and CM-2 requires linear time proportional to the number of links along which a marker is passed.
- Arithmetic and logical operations are executed in an extremely fast manner due to the algorithms developed for associative memory. These algorithms fully utilize parallel operations on all words, thus attaining nano-seconds performance in some cases.

Cases where other machines possibly outperform IXM2 has been ruled out, because these situations are practically implausible. Thus, we can conclude IXM2 is a highly suitable machine for semantic network processing which is essential to many AI applications.

Beside performance of IXM2, one of major contributions of this paper is the identification of some of the benchmark criteria for massively parallel machines. While *parallelism* has been a somewhat vague notion, we have clearly distinguished a parallelism in message passing (by a fanout factor) and in simultaneous activation of PEs (by active node number). These critaria are essential in determining which type of mahcine should be used for what type of processings and networks. For example, we can expect IXM2 to be better on large fanout but not too large simultaneous activations (most AI applications are this type), but CM-2 is better when a fanout is small, but large number of PEs are always active (most scientific computings are this type).

Acknowledgement

Authors thank Hiroshi Kashiwagi and Toshitsugu Yuba (ETL) for assisting this research, Takeshi Ogura (NTT) for providing associative memory chips, and Jaime Carbonell, Masaru Tomita, Scott Fahlman, Dave Touretzky, and Mark Boggs (CMU), Yasunari Tosa (Thinking Machines) for various supports and valuable advice.

IXM2 technical appendix

Associative Processor	T800 Transputer (17.5 MHz), 4 KB on-chip RAM (57 ns), 32K X 32 bit SRAM (230 ns), 4 serial link (2.4 Mbytes/sec), 4096 X 40 bit associative memory (cycle time 375 ns) 4 link adaptors (IMS C012)
Network Processor	T800 Transputer (17.5 MHz), 32K X 32 bit SRAM (230 ns) 2048 X 40 bit associative memory, 16 link adaptors, broadcast up to 16 destination APs or NPs

References

[EDR, 1990] Japanese Electronic Dictionary Research Institute. An Overview of the EDR Electronic Dictionaries, TR-024, April 1990.

[Evett, 1990] Evett, M., Hendler, J. and Spector, L. PARKA: Parallel knowledge representation on the Connection Machine, CS-TR-2409, Univ. of Maryland, 1990.

[Fahlman, 1979] Fahlman, S.E. *NETL: a system for representing and using real-world knowledge*, MIT Press, 1979.

[Foster, 1976] Foster, C.C. *Content Addressable Parallel Processors*, Van Nostrand Reinhold Company,1976.

[Handa, 1986] Handa, K., Higuchi, T., Kokubu, A. and Furuya, T. Flexible Semantic Network for knowledge Representation, *Journal of Information Japan*, Vol.10, No.1, 1986.

[Higuchi, 1991] Higuchi, T., Furuya, T., Handa, K., Takahashi, N., Nishiyama H. and Kokubu, A. IXM2: a parallel associative processor, In Proceedings of 18th International Symposium on Computer Architecture, May 1991.

[Hillis, 1985] Hillis, D. *Connection Machine*, MIT Press, 1985.

[Kitano and Higuchi, 1991] Kitano, H. and Higuchi, T. High Performance Memory-Based Translation on IXM2 Massively Parallel Associative Memory Processor, AAAI-91, 1991.

[Lenart and Guha, 1989] Lenart, D., and Guha, R., *Building Large Knowledge-Based Systems*, Addison-Wesley, 1989.

[Moldovan, 1990] Moldovan, D. et. al. Parallel knowledge processing on SNAP, In Proceedings of International Conference on Parallel Processing, August 1990.

[Ogura, 1989] Ogura, T., Yamada, J., Yamada, S. and Tanno, M. A 20-Kbit Associative Memory LSI for Artificial Intelligence Machines, *IEEE Journal of Solid-State Circuits*, Vol.24, No.4, August 1989.

[Quillian, 1967] Quillian, M.R. Word concepts: a theory and simulation of some basic semantic capabilities, *Behavioral Science*, 12, 1967, pp.410-430.