[illegible] /[illegible] N95-19764

PCLIPS: PARALLEL CLIPS

Lawrence O. Hall[1], Bonnie H. Bennett[2], and Ivan Tello

Hall & Tello: Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
hall@csee.usf.edu

Bennett: Honeywell Technology Center
3660 Technology Driver MN65-2600
Minneapolis, MN 55418
bennett@src.honeywell.com

ABSTRACT

A parallel version of CLIPS 5.1 has been developed to run on Intel Hypercubes. The user interface is the same as that for CLIPS with some added commands to allow for parallel calls. A complete version of CLIPS runs on each node of the hypercube. The system has been instrumented to display the time spent in the match, recognize, and act cycles on each node. Only rule-level parallelism is supported. Parallel commands enable the assertion and retraction of facts to/from remote nodes working memory.

Parallel CLIPS was used to implement a knowledge-based command, control, communications, and intelligence (C^3I) system to demonstrate the fusion of high-level, disparate sources. We discuss the nature of the information fusion problem, our approach, and implementation. Parallel CLIPS has also been used to run several benchmark parallel knowledge bases such as one to set up a cafeteria. Results shown from running Parallel CLIPS with parallel knowledge base partitions indicate that significant speed increases, including superlinear in some cases, are possible.

INTRODUCTION

Parallel CLIPS (PCLIPS) is a rule-level parallelization of the CLIPS 5.1 expert system tool. The concentration on rule-level parallelism allows the developed system to run effectively on current multiple instruction multiple data (MIMD) machines. PCLIPS has been tested on an Intel Hypercube iPSC-2/386 and I860. Our approach bears similarities in focus to research discussed in [12, 6, 7].

In this paper, we will show an example where the match bottleneck for production systems [1, 3] is eased by utilizing rule-level parallelism. The example involves setting up a cafeteria for different functions and is indicative of the possibilities of performance improvement with PCLIPS [13]. A second example of a battle management expert system provides a perspective to real world applications in PCLIPS.

---

[1]This research partially supported by a grant from Honeywell and a grant from the Software section of the Florida High Technology and Research Council

[2]Also at Graduate Programs in Software, University of St. Thomas, 2115 Summit Ave PO 4314, St. Paul, MN 55105, bhbennett@stthomas.edu.

The rest of the paper consists of a description of PCLIPS, a section describing the knowledge bases (and parallelization approaches) of the examples and speed-up results from running them using PCLIPS, and a summary of experiences with parallel CLIPS.

THE PCLIPS SYSTEM

Based on experience with an early prototype, the design of the PCLIPS user interface models that of CLIPS as much as possible. A small extension to the syntax is used to allow the user to access working memory on each node, add/retract facts or rules to specific nodes, etc. For example, a load command with four processors allocated now takes the form: (0 1, load "cafe") and (, load "cafe"). The first command will load files cafe0 to node 0 and cafe1 to node 1, and the second command loads files cafe0, cafe1, cafe2, and cafe3 onto nodes 0, 1, 2 and 3. Other commands operate in the same way with (2, facts) bringing in the facts from node 2 and (3 7, rules) causing the rules from processors 3 and 7 to be displayed.

After rule firing is complete in PCLIPS, the amount of time spent by each node in the match, recognize, and act cycles is displayed. The amounts of time are given as percentages of the overall time, which is also displayed. Sequential timings are obtained from running PCLIPS on one node.

A complete version of CLIPS 5.1 enhanced with three parallel operations, xassert, xretract and mxsend, runs on each of the nodes and the host of an iPSC2 hypercube. The host node automatically configures each of the allocated nodes without user intervention when PCLIPS is invoked. The xassert command simply asserts a fact from one node to a remote node's working memory. For example, (xassert 3 (example fact)) makes its assertion into the working memory of node 3. The general form is (xassert node_number fact_to_assert). To retract a fact from a remote working memory use (xretract node_number fact_to_retract). Both operations build a message and cause it to be sent by the hypercube operating system. Neither command depends upon a specific message passing hardware or software mechanism.

Long messages can take less time to send than many short messages on Intel Hypercubes [2, 8] so mxsend() provides the user with the capability of asserting and/or retracting multiple facts into/from one processor to another processor. The syntax of the function is as follows: (mxsend node_numbers). Mxsend() needs a sequence of calls in order for it to work as desired. The first step in correctly building a message to be used by mxsend() is to call the function clear_fact(). The syntax for this function is as follows: (clear_fact). This function simply resets the buffer used by mxsend() to the '~' character. This character is necessary for a receiving processor to recognize the received message was sent by using mxsend(). The second step is to actually build the message to be sent. In order to do this, a sequence of calls to the function buildfact() should be performed. The syntax of buildfact() is as follows: (buildfact action fact). There are four possible values for the action variable. They are '0', '1', 'retract', and 'assert'. The '0' flag and 'retract' will both cause the building of a message to retract a fact (this is done by inserting a '$' character in the message buffer followed by fact), and '1' and 'assert' will both cause the building of a message to assert fact (this is done by inserting a '#' character in the message buffer followed by fact). If the following sequence of calls is performed, (buildfact assert Hello World) (buildfact retract PARCLIPS is fun) (buildfact assert Go Bulls!!!) (buildfact assert Save the Earth) the following string will be created: "~#Hello World$PARCLIPS is fun#Go Bulls!!!#Save the Earth" Finally, the function mxsend() can be called. Mxsend() will send the message built to the specified processors so that the message will be processed by the receiving processors. The call (mxsend 10 11 12), will cause the previously built message to be sent to processors 10, 11, and 12. The proper action is taken by the receiving processors who either assert or retract facts into/from their working memory.

Since PCLIPS is a research prototype, the user is free to use or misuse the parallel calls in any way he/she chooses. No safeguards are currently provided. On the other hand, the interface is simple and the calls straightforward. The question that comes to mind is whether they provide enough power to enable useful speed-ups on MIMD architectures. Our current work shows that they are suitable for obtaining useful speedups [13], if the knowledge base is parallelized in a careful and appropriate way.

Examples

In this section we show results from parallelizing a knowledge base and discuss a real application for parallel expert systems. All speedups are reported as the sequential time or time on one node divided by the time to process the same set of initial facts and obtain the same set of final facts in parallel (Sequential Time/Parallel Time).

Before discussing examples, we discuss a few guidelines for parallelizing rule bases that have become clear in the course of developing and testing PCLIPS.

Parallelizing Knowledge Bases

There are several approaches that have been taken to parallelizing knowledge bases [5, 7, 9, 11]. An important aspect is that the parallel results be equivalent to the serial results. Methods of explicit synchronization [11] do not seem feasible until communication times are significantly reduced on parallel machines. Hence, we have pursued serialization through rule base modification. This means that the rules in a parallel knowledge base generated under our paradigm are not necessarily syntactically the same as a set of sequential rules.

There are two approaches to parallelizing the rules of a specific sequential knowledge base. The first, and most usual one, is to partition the rules independent of the types of facts they will most likely be used with. In this approach, bottleneck rules that may need to be distributed to multiple processors must be searched for during a sequential trace of the knowledge based system's operation. Processors must be load balanced with an appropriate number of rules. All parallel actions must be inserted into the right hand side of the rules. All facts will be distributed to all nodes under this paradigm.

The second approach to parallelizing the knowledge base is to parallelize it based upon the rules and the expected type of facts. This approach is only feasible if a rule base may be expected to work with one type or set of facts (with the facts themselves changing) in most cases. This approach involves an analysis of the sequential performance of the knowledge based system with a specific set of facts and then a parallelization of the knowledge base for a set of processors. In the limited testing done in our work, this approach to parallelizing rules provides a greater speed-up.

Cafeteria

There are 93 rules in our version of the cafeteria knowledge base. The rules are grouped into contexts, where an example context involves setting a table. A rule and fact partitioning of the cafeteria knowledge base was done with the use of xassert and xretract and a speedup of 5.5 times was obtained using eight processors. The speedup obtained without using these functions was 6.47 times also using eight processors. Both speedups are less than linear but notice the decrease in speedup when using xassert and xretract. The decrease in speedup is here attributed to inter-processor communication. The time required to decode a message and assert it into working memory is between 1-2 msec [10]. The time used to obtain the above results includes the time required to transmit facts across to other nodes, retract/assert them into working memory, and do the complete inferencing. A single message of 1K takes 1.1 msec to process [2]. Larger messages, however, take considerably more time to process, as shown by Boman and Roose [2]. Since, for these partitions, the messages sent across the nodes are larger than 1Kbyte (every node concatenates approximately 50 35-byte messages, making messages of 1.7 Kbytes that are sent using mxsend()), and all nodes transmit their messages to the same node (messages might have to wait on intermediate nodes and hence are blocked until memory on the destination node is available to receive the complete message [10, 2]). It is clear from the above that communication is the reason for the decrease in speedup.

The cafeteria knowledge base was also partitioned using 11 and 13 processors. A speedup of 11.5 was obtained using 11 processors, whereas the 13-processor partition produced a speedup of 22.85 times. A fact-based partitioning method was used to obtain both of these partitions. These speed-ups are clearly super-linear and occur because the match percentage of time is reduced in a non-linear fashion by this partitioning approach [13]. Due to space limitations, we will not explore this phenomenon further but refer the reader to our technical report [4].

Finally, several two-processor partitions of cafeteria were performed partitioning the rules only. A speedup of 2.035 times (65.99% matching, 21.11% acting) with two processors was obtained. In this case, rules were copied to each partition unmodified, causing the assertion of facts that are never used by the partition (since the asserted facts enable the firing of a context present in another partition). Partitioning the facts also, the speedup obtained was 2.06 (67.41% matching, 20.26% acting), which is only slightly higher than the speedup obtained when the facts were left intact. Notice that this result suggests that the number of extra unnecessary facts does not significantly affect the overall parallel execution time. A final two-processor partition was performed by modifying the rules left in each partition so that they assert only the context facts needed in the partition. A speedup of 2.13 times was obtained in this case.

Battle Management Expert System

The information fusion problem for battle management occurs when multiple, disparate sensor sources are feeding an intelligence center. This intelligence center is trying to produce timely, accurate and detailed information about both enemy and friendly forces in order for commanders to make effective battle management decisions. The challenge to the C^3I operation is to integrate information from multiple sources in order to produce a unified, coherent account of the tactical, operational or strategic situation.

There has recently been a vast proliferation of fixed and mobile, land- and air-based sensors using acoustic, infrared, radar and other sensor technologies. The result of this proliferation has made more work for the C^3I operation.

Sensors can vary in a variety of dimensions including:

- Coverage Area
- Temporal Characteristics of Coverage
- Field of View
- Angle of View
- Range
- Resolution
- Update Rate
- Detection Probability
- Modality of Imagery
- Degree of Complexity/Realism of Imagery
- Type of Target Information
- Temporal Characteristics of Reports

Each collection system, then, gives a specialized sampling of conditions to a particular level of detail, in specific locations, at a specific point in time, and with a particular level of accuracy. As a result, the analyst receives information that may be incompatible, fragmentary, time-disordered, and with gaps, inconsistencies and contradictions.

Honeywell's Combat Information Fusion Testbed (CIFT) has been developed to provide the hardware and software environment that can support development of tools powerful enough to assist intelligence analysts in correlating information from widely disparate sources. The current testbed capabilities were chosen for the context of handling three sensors: an airborne moving target indication (MTI) radar, a standoff signal intelligence (SIGINT) system, and an unmanned aerial vehicle (UAV) with a television camera payload. This correlation capability is fundamental for information fusion. By integrating Honeywell's proprietary real-time blackboard architecture (RTBA) with the proprietary spatial-temporal reasoning technique called topological representation (TR), the testbed has been able to perform the data association task. CIFT was developed and tested against a four-hour European scenario involving troop movement in a 40X60 km area that was observed by an MTI radar, a SIGINT system, and a UAV. We determined the target detections and circular error probabilities and time delay that these three systems would be expected to make. CIFT was found to operate effectively on this data, associating reports from the different sensors that had emanated from the same target.

CIFT was then implemented on the Intel iPSC-860 parallel processor [14] producing Parallel-CIFT (or Parallel-CIFT). This processor has eight parallel nodes. There are three major components of the CIFT system: Geographic/Scenario data, Blackboard Control Structures, Spatial/Temporal Reasoners.

Geographic/Scenario Data: These contain the bit maps of the map overlays and the scenario-specific operational and doctrinal data. The current scenario illustrates a Motorized Rifle Regiment in the Fulda area of eastern Germany mobilizing for a road march. This activity includes SIGINT, AUV (airborne unmanned vehicles with video camera payloads), and MTI (moving target indicator radar) sensor reports to a G2 intelligence workstation. The geographic data includes overlays for cities, primary and secondary roads, dense vegetation, and railroads.

Blackboard Control Structures: CLIPS provides the control and representation structures for the blackboard control architecture. Honeywell wrote data structures and fusion rules in the CLIPS format on a Sun workstation. These components were then parallelized and ported to the iPSC-860. Three demonstrations are available: one uses only one of the nodes on the parallel processor (this simulates a traditional serial computer for bench marking purposes), one uses two parallel nodes, and one uses four parallel nodes.

Spatial/Temporal Reasoning: The spatial/temporal reasoner for this system is built on a four-dimensional reasoner developed from Allen's temporal interval reasoning system. It defines the relations that can exist between time and space events and reasons from these primary relations.

This system represents a demonstration of concept of the Parallel CIFT system, a challenging problem in a challenging domain which effectively uses the Parallel CLIPS tool.

Current research efforts include:

- **Auto allocation of parallel components**—This work requires some basic and applied research. We propose using a nearest neighbor shear sort algorithm to dynamically allocate processing tasks across multiple processors. This will balance the load among the processors and ensure optimal performance.

- **Demonstrate P-CIFT and extensions on Paragon**—This requires three preliminary steps: 1) Port CLIPS (the forward-chaining inference engine, on which P-CIFT is built) to the Paragon, 2) Port P-CIFT to the Paragon, 3) Extensions developed to P-CIFT. See following points.

- **Addition of object-oriented data base (OODB) capabilities**—This should be easily completed with use of the CLIPS 6.0.

- **Development of a domain specific information fusion shell**—Common elements from a variety of information fusion applications (Honeywell currently has Army and Navy scenarios, with plans to extend into commercial domains, medical imaging, robotics, and electronic libraries specifically, in the next year) will be formalized and generalized for future use on other systems. This organic growth of generic components will assure the applicability, generality and usefulness.

- **Multi-hypothesis reasoning**—This will require integration of techniques for multiple hypothesis generation, maintenance, and testing. Previous related work [15] has demonstrated successful approaches in tasks with similar multiple assignment requirements. New research would be required to examine parallel implementation of these approaches. It is likely that a parallel approach could be much more efficient.

- **Quantification of performance results**—Past work has provided demonstrations of concept, but has provided no performance results.

SUMMARY

In this paper, we have discussed a parallel version of the CLIPS 5.1 expert system tool. The parallel tool has a simple interface that is a direct extension of the usual CLIPS interface for parallel use. The tool makes use of rule-level parallelism and has been tested on Intel Hypercubes. Examples of expert systems that may be parallelized have been shown. The major bottleneck involves developing effective and automated methods of parallelizing knowledge bases.

The cafeteria knowledge base example shows that good speed-up is possible from just rule-level parallelism. In fact, in the cases where both rule and fact partitioning can be done the speed-up is super-linear in this example. It appears the approach of rule-level parallelism holds significant promise for parallel expert system implementation on MIMD distributed memory computers.

The Parallel Combat Information Fusion Testbed represents a challenging real-world application of Parallel CLIPS technologies.

REFERENCES

[1] Newell, A., Gupta, A., and Forgy, C. "High-Speed Implementations of Rule-Based Systems," ACM TRANSACTIONS ON COMPUTER SYSTEMS, Vol. 7(2), 1989, pp. 119-146.

[2] Boman, L. and Roose, D. "Communication Benchmarks for the ipsc/2," PROCEEDINGS OF THE FIRST EUROPEAN WORKSHOP ON HYPERCUBE AND DISTRIBUTED COMPUTERS, Vol. 1, 1989, pp. 93-99.

[3] Gupta, A. PARALLELISM IN PRODUCTION SYSTEMS, Morgan-Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[4] Hall, L.O. and Tello, I. "Parallel Clips and the Potential of Rule-Level Parallelism," ISL-94-71, Department of CSE, University of South Florida, Tampa, FL, 1994.

[5] Kuo, S and Moldovan, D. "Implementation of Multiple Rule Firing Production Systems On Hypercube," AAAI, Vol. 1, 1991, pp. 304-309.

[6] Miranker, D. P., Kuo, C., and Browne, J.C. "Parallelizing Transformations for a Concurrent Rule Execution Language," TR-89-30, University Of Texas, Austin, 1989.

[7] Neiman, D.E. "Parallel ops5 User's Manual and Technical Report," Department of Computer Science and Information Science, University of Massachusetts, 1991.

[8] Nugent, S.F. "The Ipsc/2 Direct-Connect Communications Technology," COMMUNICATIONS OF THE ACM, Vol. 1, 1988, pp. 51-60.

[9] Oflazer, K., "Partitioning in Parallel Processing of Production Systems," PROCEEDINGS OF THE 1984 CONFERENCE ON PARALLEL PROCESSING, 1984, pp. 92-100.

[10] Prasad, L. "Parallelization of Expert Systems in the Forward Chaining Mode in the Intel Hypercube," MS Thesis, Department of CSE, University of South Florida, Tampa, FL, 1992.

[11] Schmolze, J.G. "Guaranteeing Serializable Results in Synchronous Parallel Production Systems," JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, Vol. 13(4), 1991, pp. 348-365.

[12] Schmolze, J.G. and Goel, S. "A Parallel Asynchronous Distributed Production System," PROCEEDINGS OF AAAI-90, 1990, pp. 65-71.

[13] Tello, I. "Automatic Partitioning of Expert Systems for Parallel Execution on an Intel Hypercube," MS Thesis, Department of CSE, University of South Florida, Tampa, FL, 1994.

[14] Bennett, B. H. "The Parallel Combat Information Fusion Testbed," Honeywell High Performance Computing Workshop, Clearwater, FL, December 1992.

[15] Bennett, B.H. "A Problem-solving Approach to Localization," PhD Thesis, University of Minnesota, February 1992.