TECHNICAL REPORT NO. 271

Inference Supercomputers

by

J. W. Mills, M. Burroughs, R. Wehrmeister, and D. Winkel

February 1989

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

Inference Supercomputers

by

J. Mills, M. Burroughs, R. Wehrmeister, and D. Winkel
Computer Science Department
Indiana University
Bloomington, IN 47405

TECHNICAL REPORT NO. 271

Inference Supercomputers

by

J. Mills, M. Burroughs,
R. Wehrmeister, and D. Winkel

February 1989

Inference Supercomputers

J. Mills, M. Burroughs, R. Wehrmeister, and D. Winkel
Computer Science Department
Indiana University
Bloomington, Indiana 47405-4101

Abstract

Inference supercomputers have received less attention from computer architects recently because relatively few supercomputer users require Lisp and Prolog, interest has grown in artificial neural networks, and the Japanese Fifth Generation project has encountered difficulties attaining its goals. Yet several important classes of problems can be solved using non-numeric algorithms such as unification, and would benefit from the development of powerful inference architectures. We suggest that attaching simple inference coprocessors to nodes of existing parallel computers is the most effective way to construct an inference supercomputer. The LIBRA processor being built at Indiana University is proposed as the prototype for such a node.

1. Introduction

The need for numeric computing which drove the development of early computers, and the increasing demands which led to their successors, spawned a class of supercomputers that are not well-suited as inference engines. The current problems with inference supercomputing are typified by automated theorem provers: at least one small class of problems (geometry theorems) is solvable both numerically and by using resolution, but the solutions are found approximately four orders of magnitude more slowly using resolution. The problem instances are identical, demonstrating the performance inequality between inference and numeric computing. Trainable architectures for pattern recognition (neural networks) offer one solution to certain types of inference, particularly when the problem may not be formulated precisely, but whether neural networks can be evolved as high performance symbolic computers is yet unknown. Just as there

exist problems that can be well-stated numerically, and which no one would initially attempt to solve with a neural network (weather prediction, weapons simulations), so there exist well-formed classes of problems that can be solved algorithmically — but the algorithms are not numeric. Real-time expert systems, symbolic mathematics, theorem proving, logic programming, and scheduling problems that are not amenable to linear programming (job shop scheduling) all fit into this category. We suggest that the numeric supercomputer, the inference supercomputer, and the neural network represent architectures that are best suited for specific problem classes: as such, each architecture merits independent study.

2. WHAT IS INFERENCE SUPERCOMPUTING?

Although no inference supercomputers have yet been built, architectures comparable to one node in the proposed inference supercomputer have been or are being built in Europe, Japan and the United States. Although the inference supercomputer is just as much a von Neumann machine as a numeric supercomputer, the uses to which they are put place different demands on the hardware.

At a very basic level two computing paradigms exist:

Numeric computing performs arithmetic on numbers to calculate a result.

Inference computing defines relationships between objects to reach a conclusion.

Numeric computing is more constrained than inference computing, in the sense that the objects manipulated are less abstract (although their *meaning* may be as abstract as the problem requires, it does not affect the fact that the calculations are limited to numbers or vectors and arrays of numbers). The constraints necessary for numeric processing are relatively well-understood, and their study has produced algorithms and architectures for numeric, vector, and array processing. However, the algorithms and architectures for inference processing have only recently begun to emerge.

Current architectures for inference computing are derived from seminal work done by David Moon (the Symbolics 3600), and David H.D. Warren (the Warren abstract Prolog machine). Yet even these architectures have not been the result of rigorous evaluation of the needs of complex and powerful inference engines, but rather the demands of specific programming languages: Lisp for the Symbolics 3600, and Prolog for the Warren abstract machine (WAM). The architectural requirements of resolution, reduction, term-rewriting systems are more *global* than the

requirements of a language. Consider this analogy: FORTRAN programs can be compiled to a single Motorola 68030 processor, but FORTRAN programs to solve weather prediction, ray tracing, and large systems of partial differential equations have requirements of scale that prevent their solution on the 68030. Having a hardware multiplier built into a single arithmetic logic unit is a far different thing than having an attached array processor. Similar requirements of scale turn inference computing into inference supercomputing: they must be isolated, identified and understood before an inference supercomputer can be built.

2.1 INFERENCE SUPERCOMPUTING IS MORE FUNDAMENTAL THAN KNOWLEDGE INFORMATION PROCESSING

Inference supercomputing is much less ambitious than knowledge information processing as defined in the Japanese Fifth Generation Computing effort (Moto-oka 1982). Inference supercomputing does not purport to give a solution to difficult problems in natural language recognition, VLSI design and machine intelligence. These problems may benefit from the availability of an inference supercomputer, but will require a much better understanding of the problem domain before solution is possible: for example, Prolog implementations today run at from 50,000 to 200,000 LIPS, yet computers still do not understand language, VLSI design has not been totally automated, and computers are not intelligent.

A faster inference engine cannot reduce the worst-case complexity of any NP-hard problem, but it *can* improve the ability to find solutions to those problems that are computationally feasible (as many such instances of NP-hard problems are). To use an automated theorem prover as an example, an increase in execution speed of 10^5 could reduce the time neceLIBRAry to find a proof from 2 hours 45 minutes to less than 1 second. This suggests the true importance of fast inference processing: the improvement of existing applications which can be expected if faster inference engines are available.

2.2 DESIGNING A PARALLEL INFERENCE SUPERCOMPUTER

No matter how well a programming system is understood, designing an architecture for that system will highlight incorrect design assumptions and less well-understood interactions within the system. Our proposed inference supercomputer is based on the design of two RISC architectures