EXPERT SYSTEMS ON MULTIPROCESSOR ARCHITECTURES Summary

Stanford University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. 5291

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700
This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-91-76, Volume I (of four) has been reviewed and is approved for publication.

APPROVED:  
NORTHRUP FOWLER III  
Project Engineer

APPROVED:  
RAYMOND P. URTZ, JR.  
Technical Director  
Directorate of Command & Control

FOR THE COMMANDER:  
RONALD RAPOSO  
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(COE) Griffiss AFB, NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.
EXPERT SYSTEMS ON MULTIPROCESSOR ARCHITECTURES,
Summary

Edward A. Feigenbaum
Robert Engelmore
H. Penny Nii
James P. Rice

Contractor: Stanford University
Contract Number: F30602-85-C-0012
Effective Date of Contract: 14 March 1985
Contract Expiration Date: 31 March 1990
Short Title of Work: Concurrent Expert Systems
Architecture
Program Code Number: OE20
Period of Work Covered: Mar 85 – Mar 90

Principal Investigator: Edward A. Feigenbaum
Phone: (415) 723-4878

RL Project Engineer: Northrup Fowler III
Phone: (315) 330-7794

Approved for public release; distribution unlimited.

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Northrup Fowler III, RL (COE), Griffiss AFB NY 13441-5700 under Contract F30602-85-C-0012.
This final report documents the results of a five-year investigation of methods for achieving higher performance for knowledge-based systems through the design of innovative software and hardware systems architectures. Volume I summarizes the work performed and lessons learned, and serves as an annotated index to the set of over 50 project technical reports. Volumes II through IV contain the project technical reports.

NOTE: Rome Laboratory/RL (formerly Rome Air Development Center/RADC)
Abstract

This final report documents the results of a five-year investigation of methods for achieving higher performance for knowledge-based systems through the design of innovative software and hardware systems architectures. Volume 1 summarizes the work performed and lessons learned, and serves as an annotated index to the set of over 50 project technical reports. Volumes 2 through 4 contain the project technical reports.

1. Introduction

The Expert Systems on Multiprocessor Architectures (ESMA) project was initiated in March 1985, and technical work was completed in 1990. The research was conducted at Stanford University's Knowledge Systems Laboratory. The results and findings of the project were published in a series of technical reports, which comprise Volume 2 of this Final Report. Volume 1 sets forth the basic concepts that underlie the research, and provides a road map to guide the reader through that technical literature. Volume 1 ends with a project bibliography, which serves as a table of contents for Volumes 2 through 4. ESMA builds upon and straddles a number of areas of research in computer science, including artificial intelligence, programming languages, operating systems, communication protocols and hardware. Prior to this project, some work was done on analyzing the performance of rule-based systems on parallel architectures, most notably by Gupta [Gupta 86]. On the hardware side, there are commercially available machines that are similar in some respects to the architectures considered here, notably the Ametek machine. The relationship of ESMA to work in other fields is documented copiously in the papers cited below.

On the other hand, this investigation is unique: the project focussed on applications characterized by symbolic (largely non-numerical) computation; took an end-to-end multi-level approach toward identifying and exploiting concurrency; and used highly instrumented simulation to permit careful analysis of experimental results.

In the remainder of this chapter we set forth the goals of the project and list the personnel who contributed toward achieving those goals. Chapters 2 through 5 describe and summarize each of the four levels of analysis in our multi-level, vertical-slice strategy. Chapter 6 draws together the principal conclusions and lessons that were learned from this research. Chapter 7 is a full bibliography of the technical reports that were produced by project staff. Chapter 8 lists other referenced works.

1.1. Project Goals

The project's primary goal is to find ways to increase the performance of expert systems through the use of the new, emergent, parallel hardware designs.

The number of possible implementation strategies for such a project is huge. One has only to look at the large number of different hardware designs that are emerging and at the number of different problem-solving methods to see how combinatorial the problem would be if we endeavored to investigate all of the reasonable and plausible combinations of architectures. It was decided, therefore, that we could learn a great deal simply from making a commitment to one, or at least a small number of different options at each point in the system's make-up. We thus decided to take a "vertical slice" through the space of possible solutions. Clearly we did not intend to investigate any options that seemed non-useful, so we knew from the outset that, although we could not prove that we had the best design to
meet our goals, our design would nevertheless be at least a plausible architecture for a future computational environment.

We viewed the task of implementing concurrent expert systems as being one which was split into a number of implementation layers. If we could achieve speed-up at each one of these layers, then we could hope for a substantial overall performance improvement compared to existing AI systems. Our model of the layers into which the project could be split is shown in Figure 1.

![Layer Diagram]

*Figure 1.* The layers of system implementation through which we hoped to achieve computational speed-up in the project.

It was originally anticipated that the needs of the applications would drive the development of the problem-solving frameworks and so on down through the implementation hierarchy shown in Figure 1 until eventually the hardware would be designed under the constraints passed down from above. In practice, however, this did not happen. Because of the difficulty of finding and mounting an application suitable to our needs and the early availability of personnel interested in the hardware design aspect, the hardware design went ahead more rapidly than the other layers. This resulted in our designs being more hardware driven than application driven. This approach has its advantages, for example, an entirely top-down design process could easily have resulted in low-level system requirements which were not implementable.

As well as the thrust of the project coming from the bottom rather than the top, the levels of abstraction actually implemented differed significantly from those shown in Figure 1. Figure 2 gives a more realistic representation of the layers that were actually investigated, as opposed to what we intended to do.
The Knowledge Systems Laboratory has considerably more expertise in software than in hardware. We thus decided early on not to build any hardware - there are many other research groups that could do this better than we. We decided, therefore, to simulate our hardware. This would allow us to modify our software and hardware designs easily and allow us to extract the maximum insight with the minimum effort.

The rest of this paper is split into sections which reflect the major layers shown in Figure 2. In each of these sections the work of the relevant sub-projects will be discussed. Because of the bottom-up thrust of the project the project's components will be discussed in a bottom-up order. This will also reduce the number of forward references made, since discussion of the higher layers will inevitably have to refer to the substrates on which they are implemented.

1.2. Personnel

This project has employed a large number of people over the years and it seems appropriate to name them all here:


2. Hardware-Level Systems Studies

As was mentioned above, hardware system design led the way in the project. In this section we discuss a little bit of the motivation for the hardware designs and briefly describe both the current generation of hardware designs on which we are working and the simulator we are using.
2.1. Simple and Helios

Figure 3. The Simple system provides a toolkit from which to build circuits to be simulated, a collection of probes to connect to the circuit and a set of instruments to connect to the probes.

The hub of all of the work done on the project has been the digital circuit simulator, upon which everything else is built. This simulator is called Simple. It is an event-driven simulator, designed to allow the user to design and specialize digital circuits in a simple and modular way, using a circuit design tool called Helios. A sophisticated set of instrument tools allow the user to design and specialize simulated probes which can be connected to the circuit while it is running. This allows the connection of a number of instruments to the probes that permit the user to see the behavior of the circuit as it operates without interfering with the behavior of the system. We like to view this model as one of a laboratory workbench equipped with collections of instruments, probes and circuit building compo-

¹Note: This simulator could be used to simulate events down to the gate level, but one of its powerful attributes is its ability to allow the programmer to define the behavior of composite objects in terms of methods that make these devices appear to be atomic black boxes. This ability obviates the need to do gate level simulation of those aspects of the system whose behavior is well understood. This has enormous benefits in terms of simulation time.
ments from which the user can build systems and on which the user can perform quantita-
tive experiments (see Figure 3).

The key factors that make the Simple simulator so powerful are detailed in [Delagi 86b, 2-
272]¹, [Delagi 87, 2-294] and [Saraiya 90a, 4-360]. In effect, the simulator focuses on the
critical design aspects of multiprocessor design, namely interprocessor communication and
topology. The simulation is less detailed in other areas. This allows the user to simulate
the execution of sophisticated problems, rather than the toy problems or small code
fragments possible with other simulators. The instrumentation in the simulator is powerful
and flexible, not only allowing the user to observe events in the simulated system at
multiple levels of abstraction, but also readily allowing the user to modify and specialize
instrumentation so as to focus the simulator more sharply on interesting application-specific
behavior. This allows the user to gain substantial insight from simulator runs, while still
allowing the user to reconfigure the system easily and quickly in the event of an unexpected
result prompting unplanned experiments.

It was found early on that simulations of the sort we wanted to do would be computa-
tionally very expensive. An experiment was performed, therefore, to parallelize the simulator
itself in an attempt to bring down the times taken for the simulations, which often exceeded
one day in duration. This resulted in AIDE, a distributed version of Simple [Saraiya 86, 4-
297]. Unfortunately, we were unable to achieve any speed-up at all for our simulations,
largely because of the communication bandwidth and latency associated with
communicating between the multiple Symbolics machines we were using via an Ethernet
and because the simulator, being event-driven, required frequent synchronization on the
event queue, which serialized the processing. Although this experiment yielded a negative
result, it was valuable in demonstrating the importance of process grain size and
synchronization effects.

2.2. CARE

The Simple simulator mentioned above was used to design and build what we refer to as
the CARE² machine and simulation system [Delagi 88a, 2-301] (see Figure 4). The CARE
machine is that simulated machine on which all of the experiments mentioned below have
been performed. The machine's design has a few key features which are worthy of note:

• Dynamic cut-through routing with local flow control, in order to optimize network
  throughput [Byrd 87c, 2-155]. This protocol uses special packet terminators and
  selective buffering to avoid deadlock during multicasts.
• Toroidal topology. Topology can be motivated by high-level, application domain
  considerations, but it is also motivated by such low-level concerns as packaging
  and communication protocols. Cost models were developed to characterize several
topologies and these topologies were tested under simulation. On balance, we
believe that toroidally connected networks have the best overall cost/benefit tradeoff
[Byrd 87b, 2-148].
• Non-blocking message sending, so as to encourage pipe-line processing.
• Communications network with alternative paths between points, so as to reduce
  communications problems due to busy communication paths.

¹ Citations for project reports point to the bibliography at the end of this volume and also to the page
number where the report can be found in volumes 2 through 4.
²The expansion for this acronym seems to have been lost somewhere in the wash. We think that it has
something to do with the words Concurrent and Array.