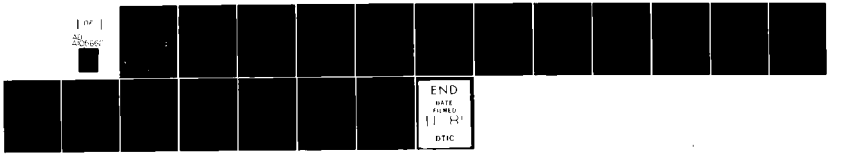
AD-A105 662 HARVARD UNIV CAMBRIDGE MA CENTER FOR RESEARCH IN COM--ETC F/G 9/2
FUNCTIONALLY SPECIALIZED MULTI-PROCESSOR ARCHITECTURES.(U)
AUG 81 N0014-76-C-0914
UNCLASSIFIED NL



END
DATE FILMED
11 81
DTIC

LEVEL II

FUNCTIONALLY SPECIALIZED MULTI-PROCESSOR ARCHITECTURES

FINAL REPORT

Prepared By

CENTER FOR RESEARCH IN COMPUTER TECHNOLOGY

HARVARD UNIVERSITY

For The

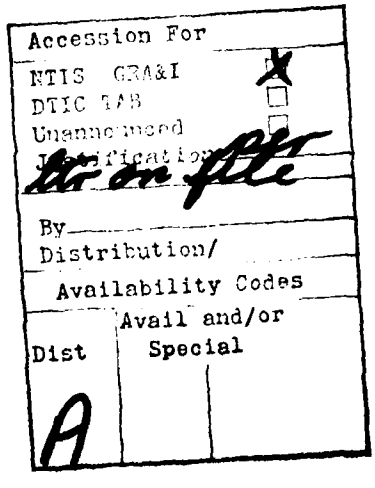
OFFICE OF NAVAL RESEARCH

CODE 430

CONTRACT N00014-76-C-0914 NR 048-637

15 AUG 1981

TABLE OF CONTENTS



1.0 Introduction

In this ~~final~~ report for the contract No. N00014-76-C-0914 which supported research on functionally specialized multi-processor architectures, we will summarize the results obtained in the various areas of research as well as some of the key publications that resulted from the work. The reader is referred to the referenced publications for greater detail on the work performed in this contract.

The research strategy pursued during this contract was based on a four-pronged approach covering those areas where, in our opinion, fundamental generalizations of the current knowledge base for computing system design was required. The areas in question are as follows:

- Data management
- Resource management
- Network management
- CPU architectures.

This report devotes a section to each one of these areas summarizing the results obtained and the publications that resulted.

We would like to point out in this introduction that the research effort resulted into two PhD theses that were published by Garland

Publishing Company on its "Outstanding Dissertations in the Computer Sciences" series. The theses in question are those of Dan Willard and Raja Jain. In addition, several research papers were published as a result of this work, and two other theses that promise to be outstanding are presently underway.

Another point of general interest is that although this program was supported under the heading of the hardware research program, three of the above topics are purely software research topics, and the fourth one, CPU architecture, is a topic which partakes of both software and hardware interests. This, of course, is not surprising, since, what is important in today's research in computer technology is not the distinction between hardware and software. What is important is the formulation of fundamental organizing concepts, which are the base of sound architectural design. Hardware, software, and firmware are simply media within which these concepts can be implemented. A choice of the media only reflects economic considerations as opposed to technical.

2.0 Data Management

The current state of the art, in the practice of computing with regard to data management, is characterized by a very heavy reliance on prestructured data models. Prestructured data models are models in which some of the possible relationships between the records in the data collection are designed into the data base itself using either indices or linked list. Data models such as: the various CODASYL DBMS's, TOTAL, IMS, etc., all belong into the category of prestructured data models. The reason why prestructured data models have had such a great success in the practice of computing is that present day mass storage systems rely heavily on rotating disks, which, as is well known, have a very high degree of latency. The very high latency of disks, in turn implies that preferred sort or search sequences through the data base must be anticipated and built-in into the data structure at design time. In other words, what we are saying is that as long as data storage is based on a high latency mechanical device, there is a considerable premium into utilizing data structures which have, ahead of time, anticipated and treated in a preferential way, via indices or special pointers, structures that prefer such sequences.

The relationship between this lack of generality of current practice data structures and multi-processor architectures, or, as it is now referred to, distributed data processing, is that, in the case of distributed data processing, it is much less likely that preferred sort sequences can be determined *a priori*. Thus, one of the conclusions of our own research planning was that, in order to make fundamental advances in the area of multi-processor architectures or distributed processing, it was imperative to have some significant breakthroughs in the area of file structures. In fact, superior underlying file structures can be utilized to accommodate a much wider range of search or sort sequences than normally is accommodated via the current data models.

At this point, it is worth pointing out that one of the technically accepted data models; namely, relational data bases, in principle, has the kind of flexibility that we are pursuing. However, problems arise when a relational data model is implemented with current mass storage technology and current ideas with regard to file structures. As we will point out at the end of this section, the more general file structures that were discovered during this research solve these implementation problems of the relational data model, and, therefore, promise to make the relational data model truly implementable in the real world where cost effectiveness and economic considerations are as important as technical considerations, if not more important.

Dan Willard, in the work leading to his PhD thesis, which was supported by this contract, has discovered a very significant generalizations of the binary trees or B-trees. B-trees are the basic structures which underlie modern file systems. Willard's generalization is called by him super-B-trees. Super-B-trees are B-trees whose internal nodes are associated to a Sub-tree Descriptor Structure (SDS).

Willard defined these trees in a very general way, since the SDS's are only requested to be any description of the descendants of the node to which the SDS is associated. So, one of the many possibilities is that an SDS, associated with a given node, represents a sorted list of the descendants of that node according to a key other than the key ordering the B-tree. Another possibility is that the SDS is, in itself, a super-B-tree ordering the descendants of a given node along several sorted sequences associated with K distinct keys.

During the research done under his contract, Willard was able to demonstrate a number of mathematical properties of these trees, and, most important of all, to exhibit some very powerful algorithms for insertion and deletion of entries into these file structures which enjoy the property that the search time of such trees remains of the O (log ^{K}N) for all poly-dimensional searches of the type.

$A_1 < K_1 < B_1 \;\&\; A_2 < K_2 < B_2 \;\&\; \ldots \;\&\; A_K < K_K < B_K$

Where:

- k is the maximum number of keys that can be simultaneously searched
- N is the number of records in the data base
- K_I is the Ith key in the search
- A_I, B_I is the allowed range for the Ith key in the search

The algorithms in question, and numerous theorems, relating to properties of these structures that Willard calls also $P_0(K)$ pyramids are given in Ref. 1.

In Ref. 3, Willard presents an improved data structure which is capable of giving search times of the O (log $^{K-1}$ N) for K dimensional searches without presenting any serious accompanying disadvantages.

For the case that K is equal to 2, this results into a data structure which occupies O(N log N) space and has the same O(log N) worst case for retrieval time as the traditional one dimensional sorted lists.

The results briefly summarized above are very significant from two points of view:

A) Academic: Since they introduce a whole new class of structures with very interesting mathematical properties

B) Practical: They show systematic (algorithmic) ways of trading off disk space for very good bounds for search time.

The practical significance stems from the fact that the disk space is becoming very plentiful and inexpensive (as is well-known, disk drives in excess of a gigabyte capacity are now available at reasonable cost) and the fact that typical queries posed by high-level decision makers are poly-dimensional.

Thus, while the military services have pursued since the mid-60's a data base management capability for Command Control systems and this pursuit has been characterized by considerable frustration and major failures (to wit, the 465L Command Control system of the USAF), such systems will now become possible once these file structures find their way into the actual practice of computing.

A final point on the practical importance of these discoveries is that these structures supply a very good implementation media for relational data base systems. Relational data base systems are characterized by a very open-ended and flexible end user functionality, but have been demonstrated, in practice, to be plagued by serious implementation problems arising from the

interaction of the mechanical characteristics of disk devices and the structural limitations of conventional file structures. These new file structures, by eliminating the limitations of the conventional file structures, hold the promise of resolving the serious implementation impasse for relational data bases

References

1) D. E. Willard "Predicate Oriented Data Base Search Algorithms" Phd thesis, Harvard University, Outstanding Dissertations in the Computer Sciences series, Garland Publishing Company, 1978.

2) D. E. Willard "Balanced Forests of K-D* Trees as a Dynamic Data Structure", Center for Research in Computer Technology, Harvard University, TR-23-78. Submitted to CACM.

3) D. E. Willard "New Data Structures for Orthogonal Queries", Center for Research in Computer Technology, Harvard University, TR-22-78. Submitted to CACM.

4) D. E. Willard "The Super-B-Tree Algorithm", Center for Research in Computing Technology, Harvard University, TR-03-79.

3.0 Resource Management

In addition to the important generalizations that were discovered in the area of data management during this research, we felt that significant generalizations of the operating system approach to resource management were required in order to tackle the multi-processor architectures or distributed processing systems. In fact, distributed data processing systems are too complex to rely on design-time resource management models, and, consequently, a theory had to be developed whereby dynamic resource management policies could be formulated. In other words, it is our feeling that in the area of distributed data processing systems, dynamic resource management policies are a must.

The research done under this contract by Raja Jain explored the application of Control Theory to the dynamic optimization of computer system performance. Until Jain started his research, queueing theory had been extensively used in the evaluation and modeling of computer systems performance. Incidentally, the application of queueing theory to the evaluation and modeling of computer system performance had been pioneered at Harvard by our research group in the early 70's. Queueing theory is a good design and static analysis tool. This limitation comes primarily

from the fact that the theory of stochastic processes, on which queueing theory is based, leads to algebraic solutions only for the steady state or equilibrium probabilities. This limitation of queueing theory, of course, provides no run-time guidance. In order to obtain tools for the dynamic optimization (run-time optimization) of computing systems, one must exploit modern Control Theoretic technique such as state space models, stochastic filtering and estimation, time series analysis, etc.

The contract research resulted in showing that the general Control Theoretic approach can be used in formulating operating system resource management policies, and, in particular, this was done by formulating policies for CPU and memory management since these are two of the most important resources in computing systems.

The problem of CPU management is that of deciding which tasks from among the set of ready tasks should be run next. The main problem encountered in the practical implementation, of theoretically optimal algorithms, is that the service time requirements of tasks are unknown. The solution which resulted from our research is to model the CPU demand as a stochastic process, and to predict the future demands of a job from its past behavior. Several analytic results concerning the effect of prediction errors were derived. An empirical study of

program behavior was conducted to find suitable predictors. Several different models were compared. Finally, it was shown that at zeroth order autoregressive moving average model is the most appropriate one. Based on this observation the research work proposed an adaptive scheduling algorithm called shortest predicted remaining processing time.

During this research the problem of memory management was also formulated as the problem of predicting future page references from past program behavior. We were able to show, using a 0-1 stochastic process model for page references, that the process in question is non-stationary. Further empirical analysis was done and showed that the page reference pattern can be satisfactorily modeled by an autoregressive integrated moving average model of the order of 1, 1, 1. We were also able to derive a two-stage exponential predictor for this model. Based on this predictor, a new algorithm called the ARIMA page replacement algorithm was formulated. This algorithm is shown to be easy to implement. One of the most interesting set of results from this research was that many of the conventional page replacement algorithms, used in current operating system design practice, including the working set algorithms, are merely boundary cases of the ARIMA algorithm. The theory that was derived during this research spells out the precise conditions under which these conventional algorithms are optimal page replacement algorithms.

The basic significance of this research work is highlighted by the last mentioned result; namely, that the ARIMA algorithm has, as boundary cases, most of the conventional page replacement algorithms. The point is that one can show that control theory determines, in state space, a region of feasibility. The boundary of the region of feasibility corresponds to conditions where the conventional algorithms are optimal. The ARIMA algorithm allows to adapt the memory management page replacement policy as the operating point of the system moves through the state's space. What this means is that these are much more general purpose run-time dynamic algorithms which are capable of adaptive behavior. Another way to put it is, that by using these kinds of algorithms, it is possible to design operating system resource managers which are truly adaptive to the actual workload as it shifts at run time from one workload profile to another.

References

1) Rajendra Kumar Jain "Control Theoretic Formulation of Operating Systems Resource Management Policies" Phd thesis, Center for Research in Computer Technology, Harvard University, TR-10-78, also published in the Garland Publishing Company, Outstanding Dissertations in the Computer Sciences series 1978.

4.0 Network Management

Another important area of generalizations required to support distributed data processing architectures is that of network management. At the time this contract expired work was in progress on a thesis in this area. Since research was not completed at the time, in this section we will only report a summary of the status attained at the end of the contract.

The research conducted by James Frankel while pursuing his Phd thesis utilizes, through a cooperative arrangement with a MIT research group, the Xerox local net (Ethernet) and the Alto's processors available at MIT.

Local networks will prove out to be central to distributed data processing and, contrary to widely held belief, are not very closely related to long distance or geographical networks which are the subject of most previous computer networking research.

The two key reasons why local networks represent a distinct field of research are:

A) Local network technologies, such as Ethernet, Token-net, etc., have bandwidths in the one megabyte per second to ten megabytes per second range, that is, the bandwidth is comparable to that of mass storage.

B) Local network technologies are much less noisy than long-distance transmission facilities.

The first point above implies that the architectural treatment of local network management and virtualization can be and should be very similar to that, for instance, of the mass storage resources in a traditional operating system. That is, it should be dealt with in the very deep layers of the distributed data processing system OS, and with a structure which is quite similar to that conventionally used for other physical resources.

By conventional architecture we mean a similar structure of multiple layers such as: the kernel/physical driver layer, the logical I/O layer, followed by file management or data base management layers, with the files, data bases, and directories defined as location independent abstractions.

The second point above implies that the logical I/O layer of the network management does not need to implement the complex "handshake" protocols which are traditionally associated with communication software.

Since this work was far from complete, no publications have resulted from it at the time of completion of the contract.

5.0 CPU Architecture

In the last decade data typing has emerged as a very important and central Computer Sciences notion.

It is an important notion for programming methodology since high-level languages which use it are capable of automatically checking a very rich class of serious programming errors. Furthermore, such high-level languages effectively prohibit the use of a large class of "dirty tricks" based on implicit re-interpretation of the semantics of data. Such dirty tricks are the bane of clean and maintainable software.

The notion is important for OS technology since it significantly assists the implementation of abstractions, or abstract data types.

The latter notion consists of a partitioning the design of an operating system into a number of information-hiding modules or virtual machines. Such machines by completely controlling access to data structures which they "own" insure that a consistent and less detailed (abstract) visibility of the data structures is furnished to all of their users.

Abstractions are important since the detail that is abstracted out can be modified in subsequent implementations without effecting the users of the abstraction.

Data typing assists the implementation of OS abstractions in two ways. First, in the programming of a module implementing the abstraction, it allows the implementation of the data structures and the related server code in a manner that reduces semantic ambiguity. Secondly, in the programming at the (higher) using level, the abstract data can be referred to as a data type of that level. This, in turn, means that one gains the same programming advantages at the abstraction application level that were available at the abstraction implementation level.

Another way to say this is that since modern notions of operating system design emphasize the partitioning of the data structures in classes, and their use, at a higher level, as abstractions (selected detail is left out) any hardware/programming tool which forces explicit and unambiguous definition of data structures will be beneficial.

Because of the importance at both the programming and system level of data typing, its usage is quickly becoming frequent. As many other times in the past, notions which are frequently used eventually find their way into the hardware.

Our research in this area, was and is aimed at formulating an approach to supporting data typing directly at the hardware level as opposed to the programming language level.

The work of Jeff Herman, a Phd candidate in our department, has led to the discovery of a class of hardware supported data descriptors which appear to support a, virtually unlimited, collection of data types, while not posing significant space/time degradations, or leading to unmanageable housekeeping algorithms for the data descriptors themselves.

These notions, once proved in a definitive way, will drastically revise current notions on CPU architecture, which have basically remained unchanged since the beginning of the computing era, and still reflect the Von Neuman machine.

We expect that the new class of CPU architectures will greatly increase the ease with which modern notions of programming languages, and OS structures will be implemented.

Since this work was unfinished at the time of the expiration of the contract, no publications have resulted during the contract period.

END

DATE
FILMED

11-81

DTIC