AD-A065 442 STANFORD UNIV CALIF STANFORD ELECTRONICS LABS F/G 9/2
DISTRIBUTED DATA PROCESSING FOR BMD.(U)
FEB 79 M J FLYNN DASG60-77-C-0073

UNCLASSIFIED NL

1 OF 1
AD
A065442

END
DATE
FILMED
4-79
DDC

COMPUTER SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

LEVEL II

DISTRIBUTED DATA PROCESSING FOR BMD

DDC
RECEIVED
MAR 8 1979
C

FINAL REPORT ON CONTRACT NO.
DASG60-77-C-0073

FEBRUARY 1979

SUBMITTED TO:

BALLISTIC MISSILE DEFENSE ADVANCED TECHNOLOGY CENTER

BY:
MICHAEL J. FLYNN

AD A065442

DDC FILE COPY

"Approved for public release: Distribution Unlimited"

79 03 05 116

(6) DISTRIBUTED DATA PROCESSING FOR BMD.

(9) FINAL REPORT. 11 Jan 78 - 23 Jan 79

Submitted to:

Director
Ballistic Missile Defense Advanced
Technology Center
ATC-P
P.O. Box 1500
Huntsville, AL 35807

(11) Feb 79

From:

(10) Michael J. Flynn
Professor, Electrical Engineering
Computer Systems Laboratory
Stanford University
Stanford, CA 94305

(12) 18 p.

Contract No:

(15) DASG60-77-C-0073
~~Mod. No. P00001~~

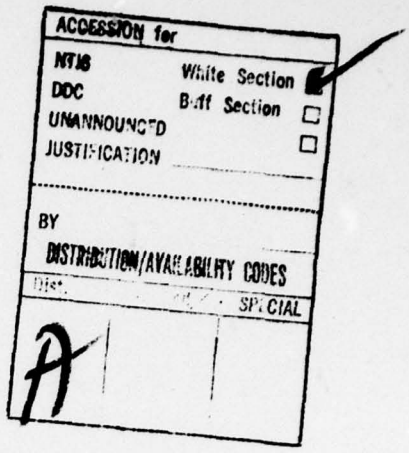
Period Covered: January 11, 1978 - January 23, 1979

The views, opinions, and/or findings contained in this report are those of the author and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

Approved for public release: Distribution Unlimited

79 03 05 116

332 400



FINAL REPORT

January 11, 1978 - January 23, 1979

DISTRIBUTED DATA PROCESSING FOR BMD

Michael J. Flynn

Principal Investigator

INTRODUCTION

During the past year, the emphasis of ~~our~~ this group in support of Distributed Data Processing for BMD was on the development of tools for more accurately predicting the performance of distributed data processing systems and investigation alternatives for novel DDP architectures. Specifically three study areas were covered:

(1) Directly Executed Languages and "Ideal" BMD Architecture Development;

(2) Parallel Nodal Architectures; and

(3) Communication Networks.

DIRECTLY EXECUTED LANGUAGES AND "IDEAL" BMD ARCHITECTURE DEVELOPMENT

Using conventional architectures, most high level language statements must be compiled to more than one machine language instruction, even those statements that involve only one functional operation (e.g. add, multiply). During 1978, we continued our research into Directly Executed Language, or DELs. The DEL approach to the machine language representation of a program is to directly compile each functional operation in the high level language program into a single corresponding machine language instruction. Each DEL instruction actually can be interpreted in less time than a traditional instruction and since fewer are needed for a given computation, the computation will go significantly faster. In addition, the instructions are highly encoded so that they take up less space in memory and require less memory bandwidth devoted to the instruction stream; this also contributes to faster execution for a given hardware expense. There are two aspects to the encoding. First, variables are not identified by their addresses. Instead, each variable is assigned a minimum-length string of bits. The compiler determines the length by counting the number of variables, local and non-local, referred to in each scope (procedure) and taking the log base two. (e.g., three variables can be distinguished by two bits.) Second, if a variable occurs more than once in an instruction (e.g. i := i + 1) then that fact is recorded in the opcode and the variable's bit string identifier need not appear more than once in the instruction.

Work completed before 1978 had explored the above ideas and Lee Hoevel had proposed a DEL for Basic Fortran and had microcoded a universal host, the Stanford EMMY, to run programs written in that DEL. Some of the findings of

that study are summarized in the following table, which for various measures shows the ratio between the DEL representation and three conventional machines, for one test program.

	Static size in bits	Dynamic # of instructions executed	Dynamic # of memory references
IBM 370:DEL	5.1	4.1	4.9
Honeywell:DEL	5.4	2.9	3.8
Burroughs S-Language:DEL	4.5	3.5	2.6

In 1979, we extended the DEL research as follows: First, we surveyed constructs and methods used in current production compilers for conventional machines, to see how problems that DEL's and conventional machine language have in common are solved.

Next, we began to design a DEL and write a compiler for a high-level language more generalized than Basic Fortran. This, along with an interpreter (written for the Stanford EMMY) and a set of test programs will allow us to determine by how much DEL's represent an advance over conventional architectures and (2) to provide a concrete design of a full DEL allowing us to estimate the performance of a possible DEL machine whose hardware will have been designed to suit a particular DEL.

We have successfully taken a small high level language test program through the compiler and run the resulting DEL code on EMMY using the DEL interpreter. The test program was restricted to assignments of integer expressions, but exercised all 21 instruction formats involving assignments and expressions.

In addition, work was completed on the study of research into neural networks as to its possible applicability to the DDP problem. This study was documented in CSL TN #138, "On Research into Neural Network Principles", by Scott Wakefield.

PARALLEL NODAL ARCHITECTURES

The objective of this study is the development of analytic tools for evaluating the performance of parallel nodal architectures. The architectures considered are tightly-coupled parallel-processor organizations, of the SIMD or MIMD variety. These are nodal architectures since the tight-coupling between the processors requires them to be located in a geographically common site. Each processor can execute an operation in one time-unit, and anywhere from 1 to p processors may be busy executing an operation in a time-unit. For the purpose of this study, we may assume that the system is processor bound, so that there are no delays due to memory, interprocessor communications and input-out processing. Hence, we are concerned with the performance potential of parallel processor organizations, and this potential may not be full realized due to the delays caused by the cooperation and communication between system components which exist in an actual system.

1. Control Dependencies - Characterization and Minimization

A basic observation underlying this study is that the performance of a parallel processor organization is limited not only by the above physical constraints imposed by the computations being executed. Although an organization with p parallel processors has a maximum processor bandwidth of p operations per step, the computation being executed usually cannot utilize all p processors at each step of the computation. This is because of control and data dependencies in the program which force a sequential chain of execution for the dependent operations.

We studied the problem of control dependencies, which are caused by conditional branches within the program. Since the results of the condition

upon which the branch will or will not be taken is determined at run time, execution is held up and severe time penalties are paid, especially by overlapped systems with parallel or pipelined processors. In TR 156 we showed how to characterize the uncertainty caused by these embedded dynamic decisions as the "decision entropy" of the program. From this, we can define and then construct optimal program control structures which minimize the control dependencies in the execution of the program.

A whole field of future research would involve a similar characterization and then minimization of the data dependencies in a program. We should also examine how operator precedences limit the parallelism in programs, and how to minimize such degradations in the performance. Finally, we need to study how to combine the efforts of control dependencies, data dependencies and operator precedences.

2. Characterization of Parallel Computations - Canonical Forms and Performance Measures

Since the performance of parallel processor organization depends so much on the parallelism inherent in computations, we need to learn how to effectively characterize parallel computations by simple canonical forms, and define their performance with respect to different criteria.

In TR 158 we have tried to find simple and effective ways to represent the parallelism inherent in computations, which facilitate the calculation of important performance characteristics. We introduced the canonical form called the Parallelism Profile which substantially reduced the complexity of modeling an arbitrary parallel computation. The Parallelism Profile retains only the information on the frequency of different degrees of parallelism in the computation. The most succinct representation of a parallel computation, for the purposes of calculating performance measures defined in this paper, is given by the

TOP-form of the computation. This is a 3-tuple consisting of the execution Time, the total number of Operations executed, the maximum degree of Parallelism or the maximum number of Processors used in any step of the computation.

We defined quantitative measures which characterize the absolute and relative performance of a parallel computation, compared with an equivalent serial computation. The absolute performance measures are the Parallelism Index, PI(P), the Utilization, U(P), and the maximum Quality, O(P). The corresponding relative performance measures are the Speedup, S(P,1), the Efficiency, E(P,1), and the Quality, Q(P,1). We also examine the range of permissible values for each performance measure.

Although speed is the performance characteristic we are most interested in, a measure of the cost-effectiveness of executing a computation on a parallel processor organization is also important in practice. We defined Utilization as the proportion of the total processor-time space, P*T(P), actually used in executing operations of the parallel computation. The corresponding performance measure relative to the minimum size of an equivalent serial computation is the Efficiency measure, E(P,1). Again, if the serial computation chosen for comparison is an optimal one, then the Utilization, U(P), is an upper bound for the Efficiency, E(P,1').

It is important to realize that if we measure Speedup, S(P,1), and Efficiency, E,(P,1), with respect to a nonoptimal serial computation, then it is possible to have a speedup greater than P, and an efficiency greater than one. In these cases, the values do not reflect the performance improvements due solely to parallel versus serial processing, since part of the improvments are due to optimizing the serial computation itself.

3. The Average Processor Bandwidth, or Speed

Having learned to characterize parallel computations, we turn to the problem of evaluating the average speed, or average processor bandwidth of a parallel processor organization. In TR 158, this was identified as the computation's Parallelism Index (PI), defined as the average degree of parallelism in a step. We have discussed that Pl is an upper bound for the speedup, S, of a parallel computation over an equivalent optimal serial computation.

In Some earlier work, we showed that the speedup n as an upper bound of P/ln p. In TR 158, we show how we may calculate the expected speed, based upon a realistic and tractable model of all computations executing on a p-parallel processor.

We define C<p*,n*> to be the class of all computations with maximum degree of parallelism $\leq$ p, and total number of operations $\leq$ n. Then, as $n \to \infty$, C<p*,n*> contains all possible computations executing on a p-parallel processor. We then reduce all computations in C<p*,n*> to their Parallelism Profile canonical form. It turns out that there are only a finite number pf Parallel Profiles in C<p*,n*> for any pair (p,n). From this, we are able to use combinatorial arguments to find the expected processor bandwidth. In fact, this expected processor bandwidth is larger than ln p, for n is sufficiently large. Furthermore, it tends to ln p, as $p \to \infty$.

Although we have made significant advances in studying the speed and speedup performance measures, further work is needed to refine the results. Also, we need to study the characteristics of the other performance measures of efficiency, redundancy, and quality defined in TR 158. The question of when and how to introduce redundant operations into computations to speed up their executions, and the trade-offs involved, is an important topic for future study. Efficiency and Quality measures may then be deduced from known results of Speedup and Redundancy measures.

4. Experimental Verification

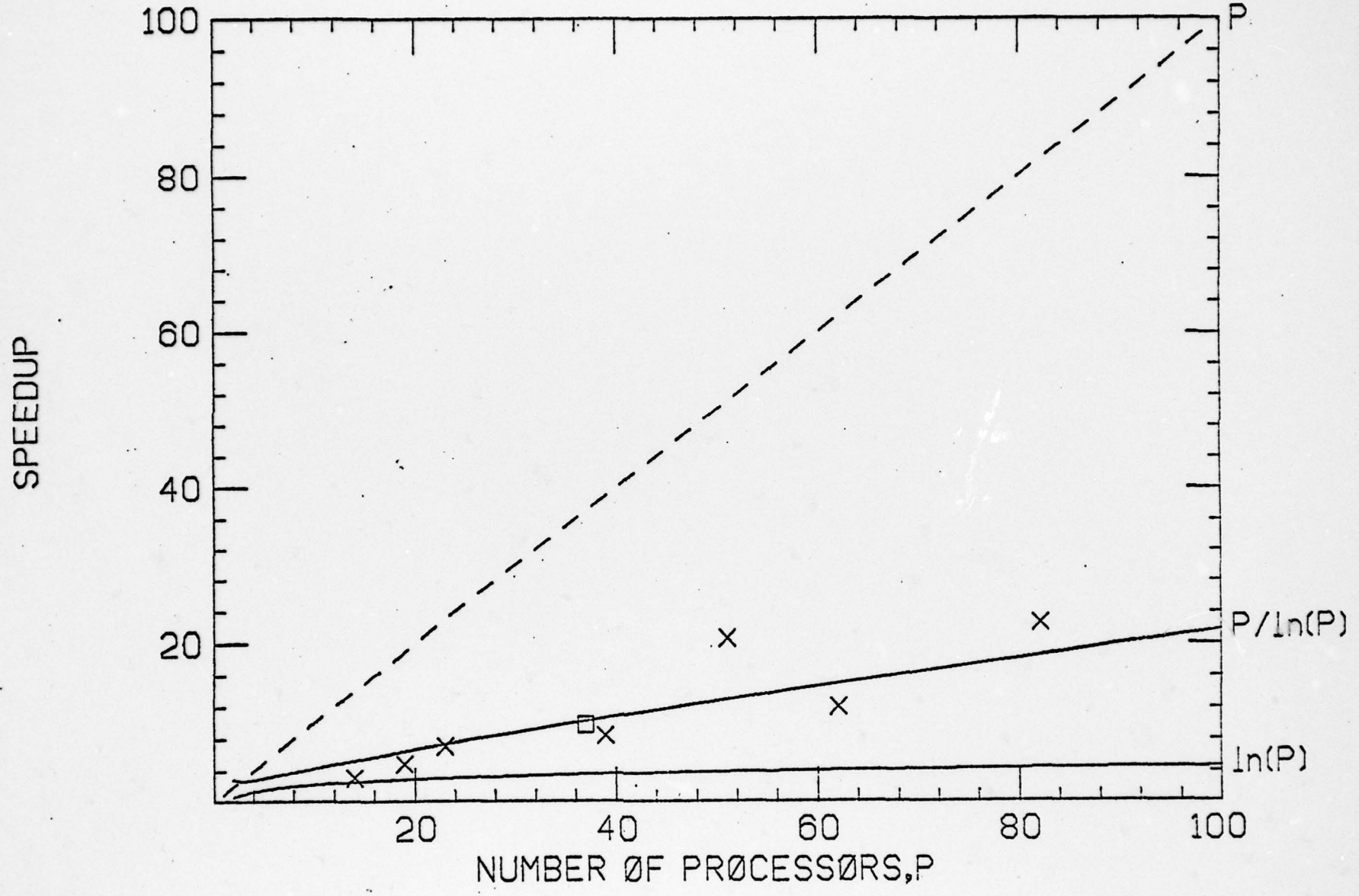
We compared our theoretical predictions with the experimental results in Version I and Version II of the University of Illinois Analyzer. This analyzer takes ordinary FORTRAN programs as input and generates parallel programs as output. In the process, the TOP-form (execution-Time, number of Operands executed, and maximum number of Processors required) of each program analyzed is calculated. From this TOP-form, we were able to calculate the absolute and relative speeds, P1 and S, for each program.

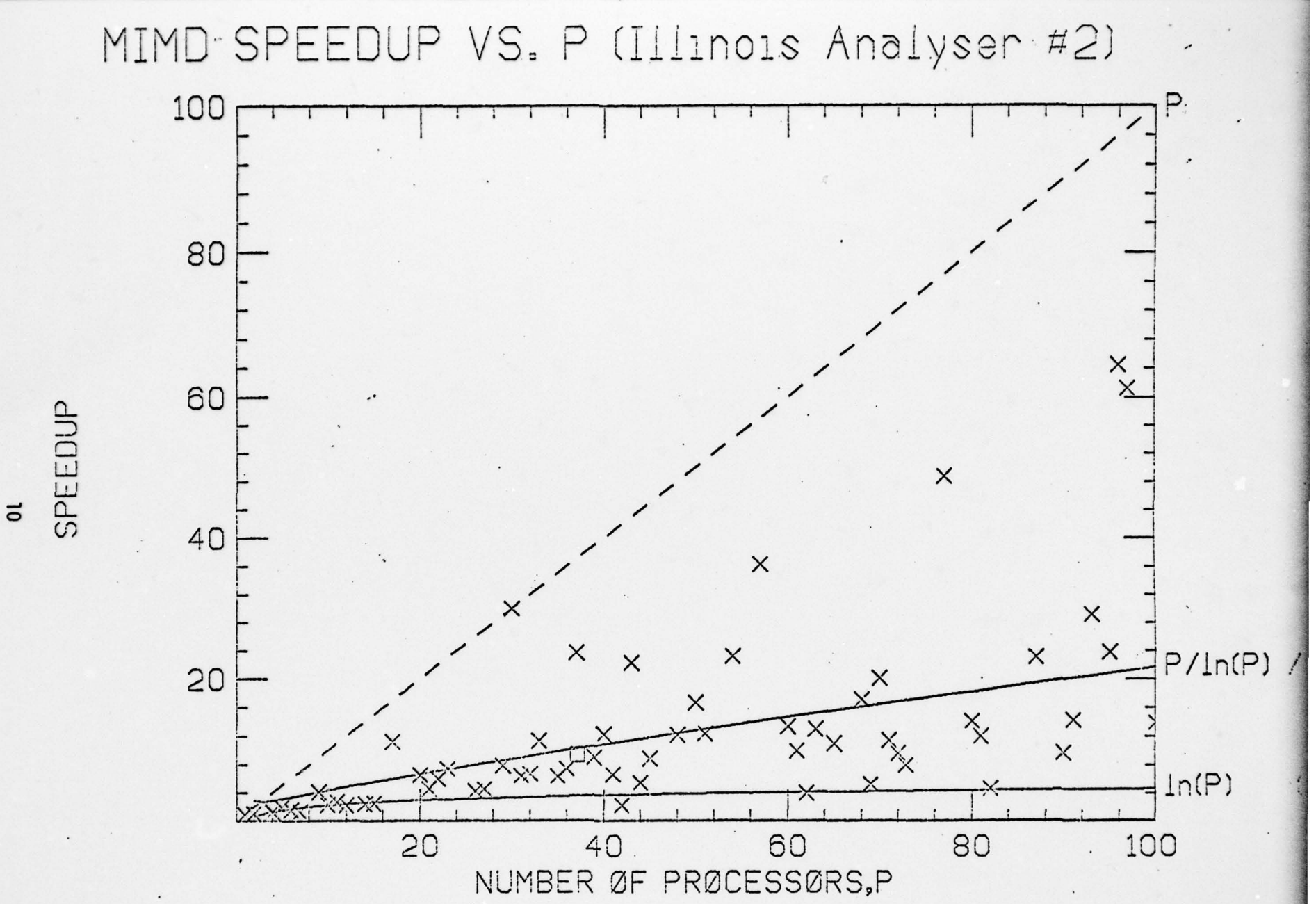
Our theoretical results in which we predict a speedup of N/ln N for N processors compared very favorably with the experimental data. The data is attached; graphs indicate the Version I and II data as well as a processor limited model (n = 16).

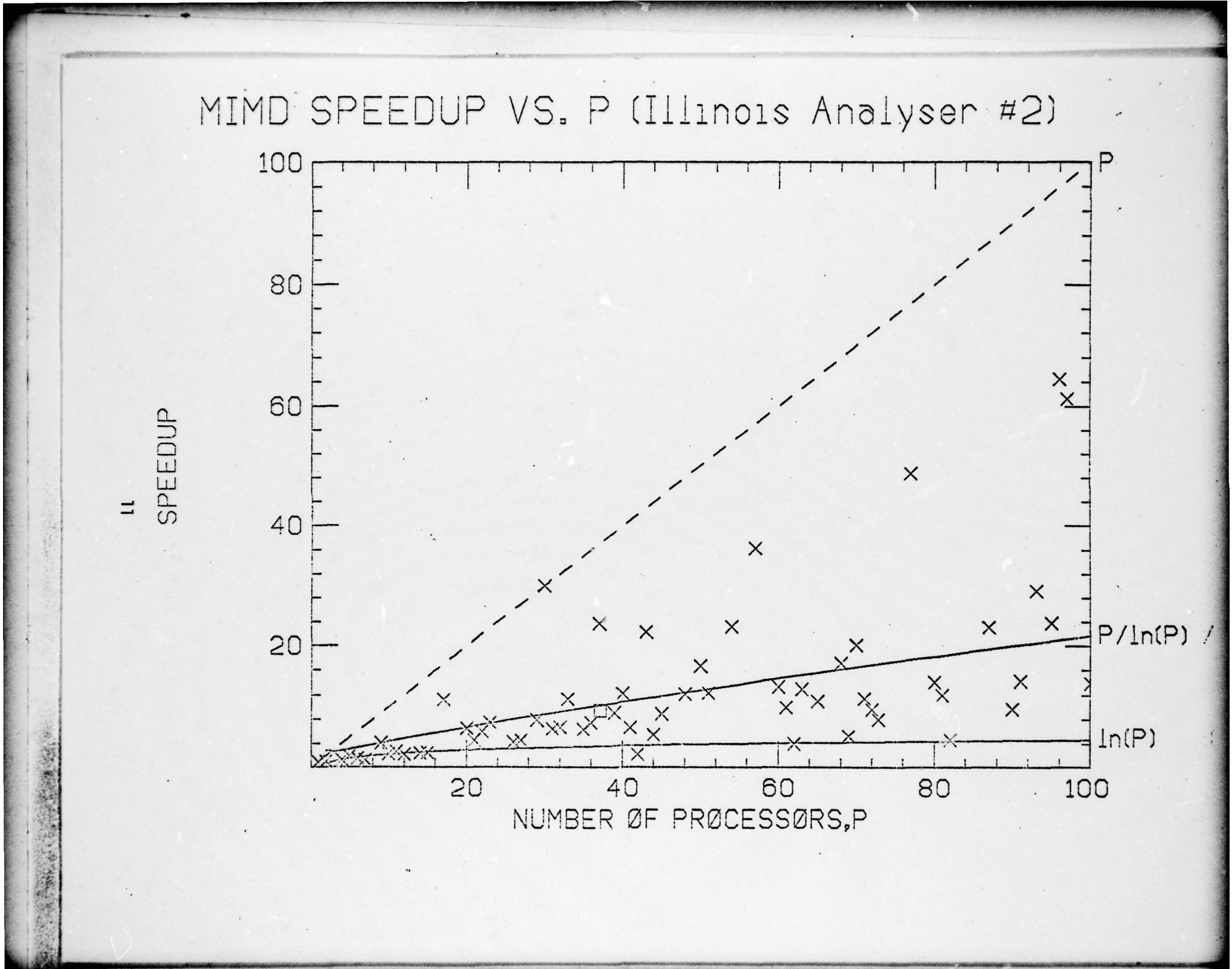
The extensive data from Version II of the analyzer (currently 355 computations) could be used not only for statistical confirmation of theoretical predictions, but also for potentially fruitful exploratory data analysis into the nature of parallelism inherent in programs.

We note that a whole field of research exists in investigating different interconnection structures for parallel processor organizations. Another field of research exists in studying how one may write parallel programs systmatically and effectively, somewhat in the style of Dijkstra's notes on "Structured Programming" for serial programs.

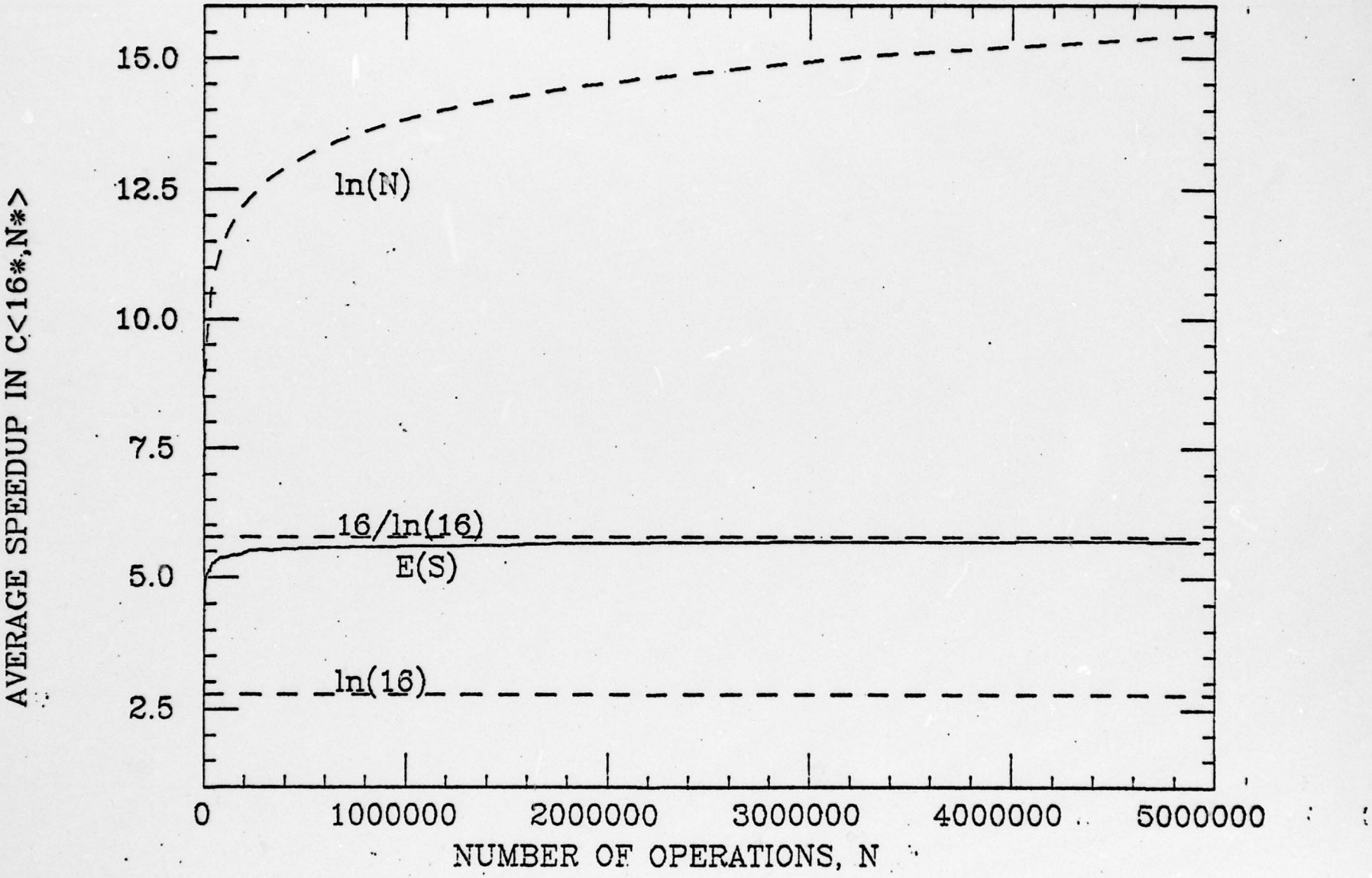
MIMD SPEEDUP VS. P (Illinois Analyser #1)




MIMD SPEEDUP VS. P (Illinois Analyser #2)
SPEEDUP
NUMBER ØF PRØCESSØRS,P
100
80
60
40
20
20
40
60
80
100
P
P/ln(P)
ln(P)


MIMD SPEEDUP VS. P (Illinois Analyser #2)
SPEEDUP
100
80
60
40
20
20
40
60
80
100
NUMBER ØF PRØCESSØRS,P
P
P/ln(P)
ln(P)

AVERAGE SPEEDUP FOR P=16 PROCESSORS



COMMUNICATION NETWORKS

Two main problems in modeling and analysis of computer communication networks have been identified as important. For each, a characterization of the major issues and an approach for attacking the problem have been given.

The first deals with the so-called "Packet-Radio" technology. This is based on the concept of packet-broadcasting which brings together the advantages of both packet-switching and broadcast communications. Packet switching (a data switching technique) offers the fair and efficient sharing of the communication resources by many contending users with unpredictable demands; the (radio) broadcast medium is a readily available resource, easily accessible and particularly suitable for mobile communications. The applications are indeed numerous and the resulting product is of great impact on the future trends of communications systems. In a packet-radio environment, the key requirement of direct communication among "terminals" over wide geographical areas leads to the existence of store-and-forward relays, called repeaters, which become integral components of the system. Such systems are then called multihop systems. The key variables and protocols which affect the system performance are: the network topology, the bandwidth management, the channel access policy, the operational protocols and the repeater design. It is clear that the design of a packet radio network involves a large number of variables which interact in a very complex fashion. In its general form, the optimum solution is hard to come by. The basic element which renders these radio systems different from terrestrial wire network is the broadcast nature of their communication: the outcome of the transmission of a packet by a repeater is dependent on the state

of the neighboring nodes and their action during the time period which constitutes the packet transmission. As a result the packet's "service time" at a node is dependent on the (entire) system state and its evolution over time. An exact analysis of systems with such a characteristic has proven to be intractable and we feel that approximate models are our only analytical recourse.

A viable approach is to create a mathematical model based on an assumption to be called the "state-independence assumption". This assumption consists of considering the effect of neighboring nodes on the evolution of the system state at a given locality as dictated by the average behavior of these neighboring nodes rather than by their actual state. The various steps that we perceive are necessary in this research are:

(i) validation of the state-independence assumption;

(ii) characterization of the input and output processes at a node describing the arrivals and departures of packets, as well as the service time distribution of a packet at a node;

(iii) creation of the approximate model based on the above two steps.

Such a model is expected to constitute a powerful tool for the design and performance evaluation of multihop packet radio systems, and in particular, to assess the effect on the system performance of many system parameters and attributes such as network connectivity, transmit power levels, channel data rates, channel access modes, etc.

The second problem identified as important is the design and analysis of data transmission protocols for the support of real-time applications. Prominent examples of applications with real-time constraints are digitized speech, video, sensor and tracking systems, seismic data, weather report, fire control, etc.

The basic distinction between real time data and regular computer-to-computer traffic are the following requirements and properties:

(i) with real-time applications, a small network delay is required;

(ii) with real-time applications, the information transmitted often is redundant (as, for example, in target tracking systems with multiple sensors);

(iii) with real-time applications, a low level of information loss is often tolerable;

(iv) the input traffic pattern to the network in real-time applications is different from computer-to-computer traffic and interactive traffic, and may not always be modeled by Poisson processes.

Although network analysis is not new, again given the real-time constraints, the analysis of real-time protocols differs from the more conventional one in that the analysis has to be extended so as to include delay distributions. The notion of "real-time" network capacity is yet to be defined and it is clear that its definition has to incorporate the tolerable message delay.

BMD SPONSORED REPORTS AND PUBLICATIONS

(January 11, 1978 - January 23, 1979)

Lee, Ruby B., "Optimal Program Control Structures Based on the Concept of Decision Entropy," TR 156, Computer Systems Laboratory, Stanford University, Stanford, California, July 1978.

Lee, Ruby B., "Performance Characterization of Parallel Computations," TR 158, Computer Systems Laboratory, Stanford University, Stanford, California, September 1978.

Lee, Ruby B., "The Expected Processor Bandwidth in Parallel Processor Ogranizations," TR 162, Computer Systems Laboratory, Stanford University, Stanford, California, November 1978 (in press).

vanCleemput, W. M., "A Structured Design Automation Environment for Digital Systems," TR 134, Computer Systems Laboratory, Stanford University, Stanford, California, December 1977.

Wakefield, Scott, "On Research Into Neural Network Principles," TR 138, Computer Systems Laboratory, Stanford University, Stanford, California, February 1978.

Yu, Philip S., "Notes on Modelling of Computer Systems and Networks," TR 154, Computer Systems Laboratory, Stanford University, Stanford, California April 1978.