PC-CUBE, A Personal Computer Based Hypercube

Alex Ho, Geoffrey Fox, David Walker
Scott Snyder, Douglas Chang, Stanley Chen, Matt Breaden

206-49, California Institute of Technology,
Pasadena, CA 91125, USA

DOE/ER/25009--587

DE88 015008

Terry Cole

180-500, Jet Propulsion Laboratory
Pasadena, CA 91125, USA

Abstract: PC-CUBE is an ensemble of IBM PCs or close compatibles connected in the hypercube topology with ordinary computer cables. Communication occurs at the rate of 115.2 K-baud via the RS-232 serial links. Available for PC-CUBE is the Crystalline Operating System III (CrOS III), Mercury Operating System, CUBIX and PLOTIX which are parallel I/O and graphics libraries. A CrOS performance monitor was developed to facilitate the measurement of communication and computation time of a program and their effects on performance. Also available are CXLISP, a parallel version of the XLISP interpreter; GRAFIX, some graphics routines for the EGA and CGA; and a general execution profiler for determining execution time spent by program subroutines. PC-CUBE provides a programming environment similar to all hypercube systems running CrOS III, Mercury and Cubix. In addition, every node (personal computer) has its own graphics display monitor and storage devices. These allow data to be displayed or stored at every processor, which has much instructional value and enables easier debugging of applications. Some application programs which are taken from the book **Solving Problems on Concurrent Processors** [Fox 88] were implemented with graphics enhancement on PC-CUBE. The applications range from solving the Mandelbrot set, Laplace equation, wave equation, long range force interaction, to Wa-Tor, an ecological simulation.

1. Introduction

Parallel computer systems promise to provide unprecedented high performance (in large configurations) and better price/performance than sequential computers (in small configurations.) Commercial distributed-memory parallel systems have been available for a few years. Programming in a parallel environment is not difficult. However, it is also not as straight-forward as programming sequentially especially for those of us who have learned and done so for many years. Programming parallel systems to perform concurrent computations requires new techniques that are best learned through hands-on experience on real parallel computers.

High-performance parallel computers are not usually available in academic environment. It is also extravagant to use such a system to teach a class. An alternative is to build your own parallel computer using already existing microcomputers [Ho 88, 88b] like IBM personal computers in a microcomputer laboratory. This paper describes the use of PC-Cube parallel system, an IBM PC-based entry-level hypercube, as an instructional and developmental tool for parallel processing.

2. PC-Cube, An Entry-Level Hypercube

The main objective of the PC-Cube project is to develop a true hypercube system for use in a microcomputer laboratory as an instructional and developmental tool for parallel processing. The PC-Cube package consists of:

(1) Communication hardware requirement which enables the hypercube connection of IBM PC's or close compatible.

(2) Software environment for the PC-based hypercube which allows applications to be written in CrOS III [Kolawa 86], Cubix [Salmon 86], or Mercury [Lee 86].

(3) To illustrate the use of the PC-Cube package by

running some of the applications found in Solving Problems on Concurrent Processors.

PC-Cube is an ensemble of IBM PCs or compatibles interconnected in the hypercube topology. The PCs can be thought of being located at the vertices of an imaginary hypercube and the edges of the multi-dimensional cube are replaced by ordinary cables. The control processor(CP), another PC, is connected to Node 0 which is one of the node PCs. Instead of using special communication channels as in the commercial hypercubes, PC-Cube uses inexpensive RS-232 serial ports. To exploit the maximum communication capacity of the ports low level routines have been written to address the UART (Universal Asynchronous Receiver/Transmitter) chip on the serial board. A very high baud rate, which is hardware limited, of 115.2 Kbaud for node to node data transmission is achieved. PC-Cube provides a system that is balanced between processing and communication speed. Applications that are written for PC-Cube will also run on larger and faster commercial hypercubes with little or no modification.

3. Advantages of Using PC-Cube

As an instructional tool PC-Cube has several advantages over the commercial hypercubes. One major advantage is its ease of use. Nodes of commercial hypercubes are not equipped with I/O devices such as display monitors, yet each node of a PC-Cube, i.e., a PC, always has either a monochrome or a color display. The availability of a display device at each node allows users to see and to demonstrate how parallel algorithms work. The graphics display capability has high educational value, as applications can display their results in a more informative and descriptive manner. Viewing the actions of an application at the nodes also facilitates debugging because errors can be pinpointed to specific nodes. In addition, each node has a keyboard, which makes it possible to implement multi-user applications such as a multi-user expert systems.

Although DOS has a 640 Kbyte memory barrier, users of PC-Cube employing parallel processing technique such as domain decomposition can perform computation on large data sets simply by decomposing them into smaller subsets and distributing over the system. Roughly speaking, a 2 Mbyte data set can be distributed among 4 node processors which makes the per node memory requirement drop to 512 Kbyte, or if 8 node processors are used each node would only need 256 Kbyte memory.

Another advantage that may not seem obvious is that PC-Cube is extremely easy to install or take apart. This feature of PC-Cube provides the opportunity for every user to have hands-on experience in setting up a hypercube. Understanding the physical connectivity of a hypercube and its relation with the control processor in the beginning should make it easier to implement communication strategies subsequently.

Even when several PCs are physically connected as a hypercube, users can still use them as standalone computers to run their usual PC software. In addition, the maintenance cost of a PC-Cube is very low as compared to that of a commercial hypercube.

4. Hardware Requirements

The hardware requirements for PC-Cube depend on the dimension of the hypercube to be set up. PC-Cubes of dimensions up to 3 have been tested. In principle, however, PC-Cube can be made larger.

PC-Cube hardware essentially consists of three components:

(1) One IBM PC or compatible for each node in the hypercube and for the control processor (CP);
(2) Standard RS-232 serial port(s) in each PC;
(3) Ordinary cables with at least 7 wires to be used to connect the node PCs and the control processor.

In general, if n is the number of nodes and d is the dimension of the hypercube, then

$$n = 2^d$$

and

$$\begin{aligned} \#\ of\ PCs &= 1 + 2^d \\ \#\ of\ cables &= 1 + n \times \tfrac{d}{2} \\ \#\ of\ serial\ ports &= 2 + n \times d \end{aligned}$$

Each node and the CP must be an IBM PC or close compatible, set up as described in the IBM *Guide to Operations* manual. Although a hard disk is not a required device for PC-Cube, it is recommended that the computer on which applications are to be developed (usally the CP) shall be equipped with one. The system software and utilities does not depend on the type of display monitor or graphics adapter. However, some of the included demonstration programs will display graphics at the CP and the node monitors. This means graphics monitors are required for these graphics demonstration programs to work properly.

Specifically, a PC-Cube of dimension d requires d serial ports in each node, except for Node 0 which requires $d + 1$ ports. The CP always requires one serial port.

Each serial port must be located at a unique address in the I O address space. When purchasing serial ports for PC-Cube, it is important to ensure that the I O addresses used by the serial ports do not conflict with each other (or with the I O addresses of any other hardware on the PC.)

PC-Cube keeps a list of possible serial port addresses. During initialisation, PC-CrOS examines each one of these addresses in turn to determine whether or not there is a serial port located at that address. The first port that PC-CrOS finds becomes channel 0. The second port that it finds becomes channel 1, and so on.

A PC-Cube cable is an ordinary cable with at least 7 wires and two 25-pin RS-232 connectors. The cable is used to tie the following pins of the two RS-232 connectors together as indicated in Fig. 1:

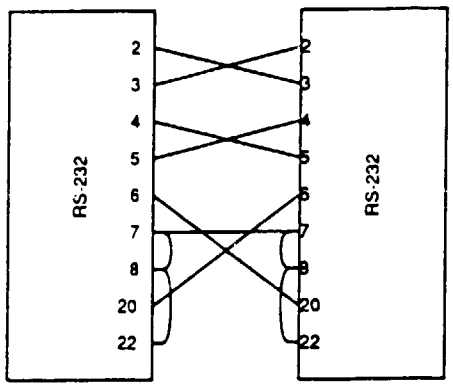

Fig.1 Cable Configuration

In addition, connect pins 8 and 22 to 7 of the same connector using short pieces of wire, i.e., short pins 8 and 22 to ground (pin 7).

5. Software Requirements

For PC-Cube to run properly the minimum software required are as follows:

(1) DOS version 2.0 or later (except for the execution profiler which requires at least 3.0)

(2) A C compiler. The PC-Cube software package was developed with Microsoft C version 4.0. Most of the code in this package also works with Turbo C version 1.0.

If a user knows how to use the Microsoft mixed language interface, the C library provided by this package can be accessed from MS-Fortran or MS-Pascal programs.

6. The PC-Cube Package

Two hypercube communications systems have been ported from the Caltech/JPL Mark III hypercube to the PC-Cube environment: CrOS III and Mercury Operating System. Also ported are the Cubix and Plotix parallel I/O and graphics library.

PC-CrOS, PC-MOS, PC-Cubix and PC-Plotix are highly compatible with the original versions. A well-written hypercube application can be ported among PC-Cube and other commercial hypercubes with only minimal modifications. In other words, programmers who use PC-Cube for code development can scale up the size of the physical problem and run the same piece of code on other hypercubes.

PC-specific utilities include a build-in performance monitor for PC-CrOS, a general execution profiler, and a simple graphics library. Brief description of PC-Cube system software and utilities are given as follows (for detail descriptions of the software, please see the C^3P documents referenced):

PC-CrOS

CrOS is a channel-based, point to point polled communication system. CrOS allows directly connected nodes of a hypercube to communicate with each other. Messages can be sent over long distance by using several hops but the programmer has to provide the forwarding instructions explicitly.

When a node is ready to send a message, that node is blocked until the receiver is ready. Similarly, if a node wants to receive a message, that node is blocked until the message is sent. Since the processors cannot proceed until the read or write is done, the processors move in locksteps through their program. Each read and write command resynchronises the processors.

In order to achieve reliable communications PC-CrOS uses synchronous communications protocol. With the serial port hardware configured to operate at 115 Kbaud, PC-CrOS is able to achieve an effective data transmission rate of approximately 47 Kbaud.

PC-MOS

MOS is an interrupt-driven communication system which provides the nodes of a hypercube the capability of performing message passing between nodes that are not physically linked by a channel. When a packet arrives, the processor is interrupted and the packet is read. If a packet for another processor arrives, it will be forwarded without any effect on the

application program. In other words, message transmission and reception can proceed concurrently with the execution of the application program. Since processors are interrupted for read and write, there is no requirement of synchronisation between the communicating processors, i.e., the processors can run asynchronously. MOS also provides synchronous mode of communication similar to CrOS.

On a PC-Cube, CrOS communications are typically twice as fast as MOS communications.

PC-Cubix

Cubix is a model of programming a hypercube without programming the control processor. It transparently provides the functionality of I/O to the node program. One of the features of Cubix is that a properly written Cubix program for hypercube computers can be compiled and executed with no changes on a sequential machine. The only requirement is that an appropriate Cubix library exists on the sequential computer.

PC-Cubix implements a subset of the Cubix library. Unix functions such as getgid, ttyname, setgid, getlogin are not supported. Additional functions such as mkdir, rmdir, ungetch, getch, putch, kbhit are available for PC-Cubix.

It also allows I/O operations to be performed on the nodes. For low-level I/O, this is done by introducing another set of low-level routines that operate on the nodes instead of on the CP. For stream I/O, locality is just another stream attribute (like singular/multiple.) Thus, the same stream I/O routines can be used to manipulate both local and CP file streams. Local I/O provides useful information for debugging concurrent programs.

PC-Plotix

Plotix is a simple graphical system for the hypercube. It runs under Cubix and Unix. It is an extension to Cubix which allows node programs to draw graphics on the CP in a portable manner.

PC-Plotix is an implementation of Plotix. PC-Plotix does not support functions such as polygon fill and certain line types. It supports both CGA and EGA.

CrOS Performance Monitor

CrOS Performance Monitor is a facility built into PC-CrOS. It allows the measurement of how much time a program or program segment spends on computation, communication, idling – waiting for communication to begin (usually communication and idle time is lumped together as communication overhead), and on performing file I/O from inside of a CrOS function.

A user can at any time turn data collection on or off, retrieve the current statistics, or prints a summary of profiling statistics to a file stream. The CrOS Performance Monitor has a real-time mode. When real-time mode is turned on, this facility will continuously display on each node both the current state of CrOS (computing, sending, receive, or waiting on a particular channel) and the timing data, presented in a graphical form. The performance monitor provides information about load-balancing of various algorithm implementations and indicates inefficiencies of different decomposition strategies.

Execution Profiler

While the CrOS Performance Monitor keeps track of profiling statistics for CrOS-specific functions, the execution profiler will work on parallel and sequential programs. Programs are large, complex systems. A personal computer executes hundreds of thousands of instructions per second. The execution profiler is a tool which samples the instruction pointer (IP) of a PC at fixed time interval and gives a measure of how a program or program segment spends its execution time in a statistical sense.

The function init_prof() opens a file for storing profiling data, and end_prof() closes the file. The start_prof() and stop_prof() functions turn profiling on and off. A special feature of this execution profiler is that it can accumulate profiling statistics following the control flow of a program instead of its structure by calling start_special() and stop_special() in the program. Two kinds of report can be generated: a symbol or a line report.

Users can use the execution profiler to fine-tune their programs for better performance.

GRAFIX

It is a collection of simple graphics primitives as well as functions to emulate some of the commercially available HALO graphics library. PC-Plotix uses the Grafix library to perform the actual graphics operations. PC-Cube programs using functions provided directly by Grafix or any other PC graphics libraries are not portable to other commercial hypercubes.

7. Experience with Code Compatibility

Several application programs were taken from the Caltech/JPL Mark Series hypercube and ported

to a PC-Cube. The algorithms used in these programs are described in detail in the book Solving Problems on Concurrent Computers. These programs range from solving the 1-D wave equation for a simple vibrating string, 2-D Laplace equation in rectangular coordination using finite difference approximation, a 3-D simulation of the dynamics of a number of particles governed by an attractive long range force such as gravity, to an ecological simulation of sharks and fish on the toroidal planet Wa-Tor [Dewdney 85]. Also, a Mandelbrot set solver [Dewdney 84] to explore the Julia curve was written.

Except for the graphics enhancement the PC-Cube version of these application programs are exactly the same as the implementations on the Mark Series Hypercubes, thus demonstrating software compatibility of PC-Cube with current CrOS III, Mercury, and Cubix programs.

8. Efficiency of PC-Cube

Although PC-Cube is designed to be an educational tool for parallel processing and not for high-performance, it provides performance speed up relative to the node PCs. Figures 2 to 5 illustrate the timing and efficiencies of PC-Cube applications using both PC-CrOS and PC-MOS communications to implement a Laplace Equation Solver on a 2-dimensional grid.

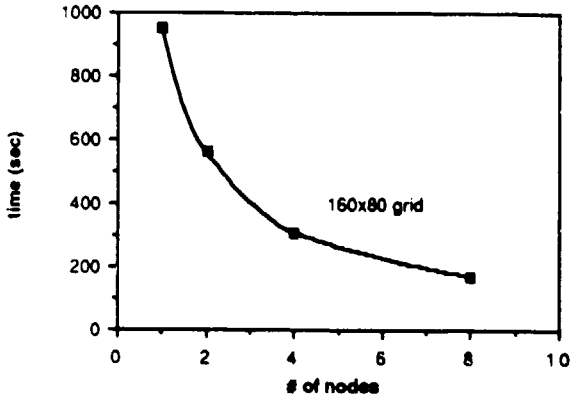

Fig. 2
Shows the decreasing amount of time needed to compute 100 iterations of updating on a 160x80 grid when an increasing number of processors are used. In this case the problem is solving the 2-D Laplace equation. The operating system used to get this data was CrOS III.

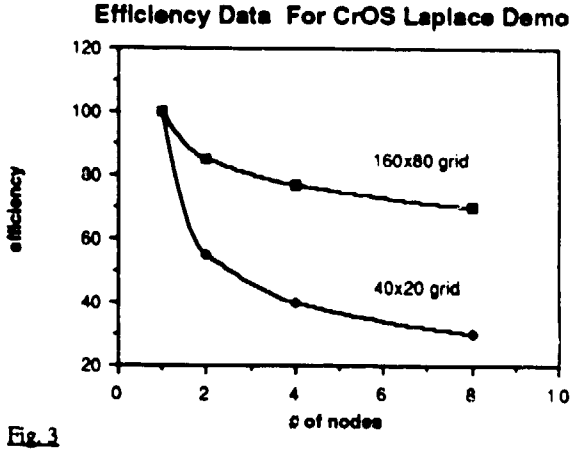

Fig. 3
Shows the efficiency (= speedup/# of nodes) of computing 100 iterations of updating on a 160x80 grid and on a 40x20 grid for the Laplace demo. Note that the efficiency is better for the 160x80 grid problem because the communications overhead is smaller so each node spends a larger proportion of its time on computation. The operating system used to get this data was CrOS III.

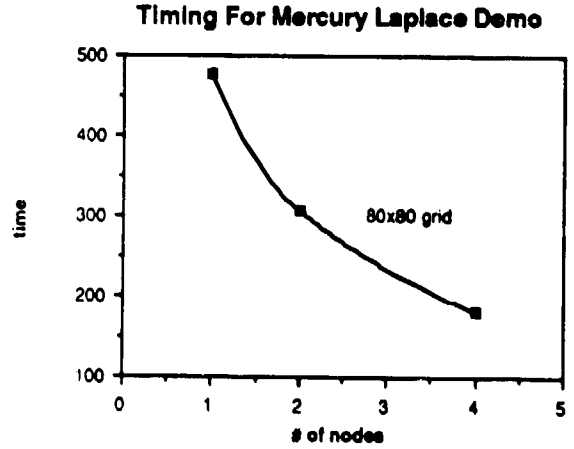

Fig. 4
Shows the decreasing amount of time needed to compute 100 iterations of updating on a 80x80 grid when an increasing number of processors are used. In this case the problem is solving the 2-D Laplace equation. The operating system used to get this data was Mercury.

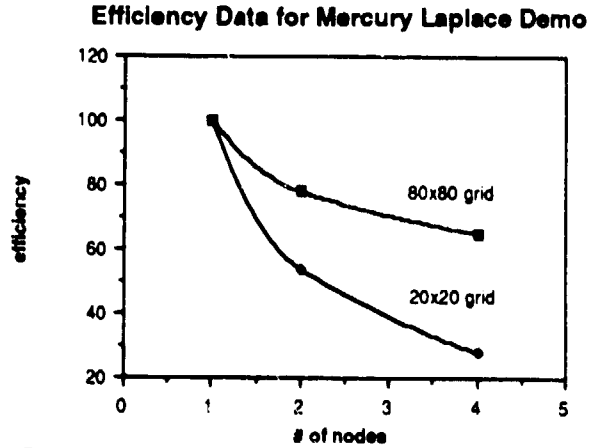

Fig. 5
Shows the efficiency (= speedup/# of nodes) of computing 100 iterations of updating on a 80x80 grid for the Laplace demo. Note that the efficiency decreases as the number of processors increases. This is due to the increasing amount of communication needed with more processors. The operating system used to get this data was Mercury

It is indicated in the figures that the finer the grid, the less the communication overhead relative to the computational load. Thus, the higher the efficiency. The CrOS version of Laplace Solver achieves about 75% efficiency with 8 nodes working on a 160 · 80 grid.

9. Conclusions

PC-CUBE is an inexpensive and easy to install hypercube system. It is an indispensable tool for learning hypercube programming and parallel processing in general. The capability of nodal text and graphics output is particularly important for debugging purposes as well as providing insights into the physical problem at hand. The performance monitor aids the development of efficient load-balanced concurrent codes for beginners and experienced programmers. The execution profiler helps users to fine-tune their programs, both sequential and parallel, for higher performance. Last but not least, concurrent programs that are written for the PC-CUBE is upward compatible. The same piece of code can be straight-forwardly ported to the larger commercial hypercube computers with little or even no modification.

10. Appendix

PC-CXLISP (concurrent XLISP) [Ho 88], a parallel version of the public domain software XLISP – an experimental LISP interpreter, has been implemented on PC-Cube. CXLISP adds Mercury communication functions to the original XLISP interpreter.

When CXLISP is executed on PC-Cube, the CP prints the CXLISP startup message and downloads the CXLISP node program. After the download process completes, the CP enters the read-eval-print loop and wait for input from the keyboard. The nodes, each running an XLISP interpreter, also print the CXLISP startup message and enter the read-eval-print loop; however, they do not read input from keyboards. Rather, they wait for a Mercury message from other nodes or from the CP.

There is also an alternate version of CXLISP node program which takes input from keyboards at the nodes instead of Mercury messages. This version is useful for debugging.

CXLISP is not distributed with the PC-Cube package because the original author of XLISP has not yet been contacted in this regard.

References

[Dewdney 84] A.K. Dewdney, *Scientific America, Vol. 251, #6*, pp. 16-24, Dec. 1984.

[Dewdney 85] A.K. Dewdney, *Scientific America, Vol. 253, #2*, pp. 18-23, August 1985.

[Flower 86] Jon Flower and Roy Williams, "Plotix - A Graphical System to Run Cubix and Unix", Caltech report C^{3}P 285, 1986.

[Fox 88] Geoffrey C. Fox, Mark A. Johnson, Gregory A. Lysenga, Steve W. Otto, John K. Salmon, and David W. Walker, "Solving Problems on Concurrent Processors", pub. Prentice-Hall, Eaglewood Cliffs, New Jersey, 1988.

[Ho 88] Alex W. Ho, Scott Snyder and Douglas Chang, "User's Guide for PC-Cube, The IBM PC-based Hypercube", Caltech report C^{3}P 563, 1988.

[Ho 88b] Ho et al., "MAC-Cube, A Macintosh-based Hypercube," this volume, 1988.

[Ho 88c] Alex W. Ho and Scott Snyder, "CXLISP - A Concurrent XLISP Interpreter on the Hypercube", Caltech report C^{3}P 559, 1988.

[Kolawa 86] Adam Kolawa and Barbara Zimmerman, "CrOS III Manual", Caltech report C^{3}P 253B, 1986.

[Lee 86] Roger Lee, "Mercury I/O Library User's Guide, C Language Edition", Caltech report C^{3}P 301, 1986.

[Salmon 86] John Salmon, "Cubix: An I/O System for the Hypercube", Caltech report C^{3}P 285, 1986.

DISCLAIMER