Report Number: WUCS-90-05

1990-01-01

NCUBE User Activity Academic Year 1988/89

Mark A. Franklin

This document summarizes usage and activities associated with the NCUBE computer system. The system is located within the Computer and Communications Research Center on the 3rd floor of Bryan Hall, School of Engineering and Applied Science, Washington University, St. Louis, Missouri. We begin with a brief review of the machine's hardware characteristics and then proceed to reviewing user activities.

NCUBE USER ACTIVITY
Academic Year 1988/89

Mark A. Franklin

WUCS-90-05

January 1990

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

NCUBE USER ACTIVITY
Academic Year 1988/89

This document summarizes usage and activities associated with the NCUBE computer system. The system is located within the Computer and Communications Research Center on the 3rd floor of Bryan Hall, School of Engineering and Applied Science, Washington University, St. Louis, Missouri. We begin with a brief review of the machine's hardware characteristics and then proceed to reviewing user activities.

The NCUBE multiprocessor contains 64 processor nodes connected in a hypercube configuration. Each processor has a VAX like architecture, runs at approximately 1 MIPS, and has its own local memory of 512 Kbytes. The maximum MIPS rate is thus 64 MIPS which is roughly 16 MEGAFLOPS. The machine is of the message based MIMD variety. After a message startup time of about 300 us, communications between processors proceeds at 10 megabit per second. The current configuration has a high performance graphics interface and monitor, and uses a SUN 3/150 (on loan from McDonnell Douglas) as the host processor. The system is accessible through Ethernet and most users interact with the machine remotely.

The current NCUBE user community consists of about 6 faculty and 11 graduate students. The NCUBE was also used in a new course "Parallel Architectures and Algorithms" which was taught in the Spring of 1989. A summary of user activities follows.

NCUBE USAGE SUMMARY		
Faculty/ Supervisor	Student/ User	Thesis/Project Title
W. Ball	D. Harker	Parallel Searching Algorithms
R. Dammkoehler	S. Karasek	Molecular Modeling Calculations
M. Franklin	E. Witte	Parallel Simulated Annealing Using Speculative Computation
M. Franklin	A. Bhatia	Cell Placement Algorithms for Multiprocessors
M. Franklin	R. Chamberlain	Parallel Algorithms for Mixed-Mode Simulation on MIMD Architectures
M. Franklin	EE520/CS579	Parallel Algorithms and Architectures
I. Katz	Y. Zhu	Parallel Implementation of the p-version Finite Element Method
M. Miller	T. Chen	Maximum Likelihood Estimation of NMR Signal Parameters with a Multiprocessor Computer
M. Miller	T. Schaewe	Reconstruction of Magnetic Resonance Images Using MLE
G-C Roman	K. Cox	Visualization of Concurrent Computation
G-C Roman	D. Gill	Implementing a Shared Dataspace Language on a Hypercube Multiprocessor

Faculty: William Ball
Student: David Harker
Project Title: Parallel Searching Algorithms
Group: Center for Intelligent Computer Systems (CICS)

CICS is using the NCUBE parallel computer to investigate the potential speedup of a search algorithm useful in many Artificial Intelligence applications. This algorithm is used to search trees in a best-first, cost-bounded iterative manner, in such a way that the first goal found is guaranteed to be the shortest-path solution. The algorithm was developed by V. Kumar and V. Rao at the University of Texas at Austin.

At the beginning of a search, one processor is given the initial start-state and begins searching, while the other processors ask for work from their immediate neighbors. Since the search space grows exponentially, each processor is eventually searching a disjoint region of the tree. When the cost-bounded iteration is over, all processors "synch-up" and determine the next cost-bound. The process continues until a goal is found, or a limit is exceeded (i.e. there is no goal).

On a small test problem (an instance of the 15-tile problem) speedup on up to 4 processors is very good (3.5 on 4 processors). On this small problem, using more than 4 processors does not help, since the search finishes before work can spread to more than 4 processors. We expect similarly good speedup on larger problems, which will be implemented when the search algorithm is fully debugged.

Faculty: Richard Dammkoehler
Student: Steve Karasek
Project Title: Molecular Modeling Calculations
Group: Molecular Modeling Group

The Drug Design Group (DDG) has experimented with the NCUBE computer to speed up our conformational search algorithm. Conformational search is used to model a molecule being twisted about its rotatable bonds. The resulting geometries are checked for valid inter-atomic distances, etc. All of the valid "conformations" are then recorded (usually as a set of angles, one for each rotatable bond).

This is a combinatorial problem. A typical "scan factor" is ten degrees, meaning that each bond is rotated through a circle ten degrees at a time, resulting in 36 possible configurations for each rotatable bond. If there are 15 rotatable bonds, the number of potential conformations is 36^15, or 2.2*10^23. Of course, there are many shortcuts which can cut this number down tremendously, but we still are left with problems which can run for days or even weeks.

One method of using the NCUBE is to create a set of subproblems by rotating the first bond through all of its allowed angles (there would be 36 subproblems in the above example). Then we can start 36 simultaneous searches on 36 nodes, with each search starting at the second bond. These searches are independent, so there is no communication between nodes, although each node must send back to the host any valid conformations that it finds. This node-host communication quickly becomes the bottleneck in problems which produce millions of conformations, so we are looking at some data compaction algorithms to overcome this problem. However, for problems which produce only a few conformations, and using a slightly more complicated variant of the node allocation just described (to make use of all 64 nodes instead of being limited to 36), we have achieved speedups of up to 62:1 compared to running the search on just one node.

Faculty: Mark A. Franklin
Student: Ellen Witte
Thesis Title: Parallel Simulated Annealing Using Speculative Computation
Group: Computer and Communications Research Center (CCRC)

Simulated annealing (SA) is a technique for obtaining approximate solutions to combinatorial optimization problems. Many combinatorial optimization problems are NP-complete and thus cannot be solved efficiently by known algorithms. SA is attractive for these problems because of its relatively simple applicability to the wide range of such problems. The drawback of SA is the long execution times, on the order of hours and days, required to reach good solutions for large problems. This has motivated the development of parallel algorithms to reduce computation time.

At the Computer and Communications Research Center we have taken a unique approach to parallelizing the SA algorithm. This approach uses a concurrency technique called speculative computation (SC) to extract parallelism from the serial algorithm. SC involves computing along several possible paths until it is known which is the correct path. Our parallel algorithm executes on an unbalanced binary tree of processors the shape of which changes as algorithm progresses.

The NCUBE is being used to implement this parallel version of the SA algorithm. Issues related to the implementation, include the mapping of unbalanced binary trees onto a hypercube architecture and the correctness verification of an asynchronous implementation. Speedups on the order of log2(N) are projected (N = number of processors). It is intended to apply this parallel optimization technique to the problem of load balancing (task assignment) MIMD machines such as the NCUBE.

Faculty: Mark A. Franklin
Student: Anil Bhatia
Thesis Title: Cell Placement Algorithms For Multiprocessors
Group: Computer and Communications Research Center (CCRC)

This project deals with the design and implementation of a parallel cell placement algorithm on a hypercube architecture. Effective solution of the cell placement problem is necessary to efficiently design large VLSI chips. The technique used to do cell placement is called Simulated Annealing (SA). This is a general combinatorial optimization technique that guarantees a global optimal solution if run for a long enough time.

Two parallel versions of the SA algorithm are being implemented. The first uses the work done by E. Witte (see above) and applies it to the cell placement problem. The second is discussed below. This implementation uses a parallel moves method to do placement by SA. A number of moves are performed in parallel on different processors and the results are combined during a synchronization phase. All processors start off with a local copy of a global solution from the host. Each processor is allocated some cells of the circuit and it is responsible for moving these cells and evaluating the resulting solutions. For a specified number of iterations, the processors compute without any communications between them. During the global update phase, each processor sends its solution, up a binary tree, to its parent in the tree structure. The host is at the root of the tree. The host then evaluates the solution, updates the global solution and also sends it to each node. The whole cycle then repeats until a stopping criterion is met at the host. Thus, the algorithm operates in two distinct phases. During the first phase, all nodes execute computational instructions with no communications between them. During the second phase, a binary tree is superimposed on the NCUBE nodes, and all nodes communicate with the host.

Preliminary experiments involving placing a 20 cell, 17 net circuit indicate that the speedup using the above algorithm is about 9 (over a single processor) when 16 processors are employed.

Advisor: Mark A. Franklin
Student: Roger Chamberlain
Thesis Title: Parallel Algorithms for Mixed-Mode Simulation on MIMD Architectures
Group: Computer and Communications Research Center (CCRC)

This project deals with implementing parallel mixed-mode simulation on a distributed memory machine of the NCUBE variety. Such a capability would be used to substantially speed up the design verification of digital systems. Simulation in this context is taken to include both logic and circuit simulation. Logic simulation refers to discrete time-based simulation at the gate or switch level. Circuit simulation refers to continuous simulation at the level of the differential equations that model the circuit components. Mixed-mode simulation is the combination of logic and circuit simulation in a uniform manner to that they can be executed as part of a single simulation task. Algorithmic alternatives of interest in the parallel implementation include time synchronization techniques (global clock or local clock) and circuit partitioning schemes (random, heuristic, and simulated annealing).

Initial work in this area has focussed on developing analytic models of the various alternatives. After parameterizing these models appropriately, this will allow us to select the best approach for implementation on the NCUBE. Future work will focus on this implementation effort.

Faculty: Mark A. Franklin
Course Title: Parallel Architectures and Algorithms (CS579/EE520)
Groups: Electrical Engineering and Computer Science Departments

A new course was offered in the Spring of 1989 which covered the architecture of contemporary parallel processors and approaches to designing parallel algorithms. The course text was "Solving Problems on Concurrent Processors" by G. Fox, *et al.* This text emphasizes algorithms and usage of message based MIMD style machines of the NCUBE variety. While the emphasis was on this style machine and associated algorithm development, additional readings from the literature covered other approaches parallel architecture design. These include SIMD machines of the Connection Machine type, systolic architectures and vector machines of the Cray variety. Fourteen students were enrolled in the course.

At the algorithm level such topics as parallel matrix multiplication and addition, parallel solution to PDEs, parallel discrete event simulation and random number generation, load balancing and task assignment on MIMD machines, parallel optimization techniques, and graphics techniques for parallel processors, were discussed. Class time was also spent on the numerous operational details associated with programming the specific machine available.

The NCUBE was used as the base machine for a series of hands on homework assignments designed to acquaint students with various aspects of parallel algorithms development, programming and debugging. These assignments continued through the first half of the semester and included such problems as parallel addition strategies for large numbers of integers, parallel solution of Laplace's Equation solution, and parallel image thinning algorithms and graphics display on the NCUBE. During the second half of the semester, students worked on various projects involving parallel algorithm design and implementation. These included such topics as parallel sorting and searching, parallel solutions of ODEs and PDEs, parallel communications systems simulation, and load balancing on parallel processors.

Faculty: I. Norman Katz
Student: Yimin Zhu
Thesis Title: Parallel Implementation of the p-version of the Finite Element Method
Group: Systems Science and Mathematics (SSM)

The p-version of the finite element method is a new approach to finite element analysis developed at Washington University and the University of Maryland. It is currently available for serial computers in a commercially marketed code called PROBE, and in a Washington University based research code. Our current goals are to develop a parallel form of the the p-version method and implement it on the Ncube multiprocessor.

The approach currently being taken is to study rapidly converging iterative methods. A recently developed method of this kind is a textured algorithm which has been shown to have very favorable convergence properties when used to solve finite-difference approximations to Poisson's equation. The Poisson problem has been implemented on the Ncube and a textured algorithm for the p-version is now being formulated for future implementation. Studies are being made for various test cases including Poisson's equation on a rectangle. Speed-up curves are being developed for these cases.

Faculty: Michael Miller
Student: Tim Chen
Thesis Title: Maximum Likelihood Estimation of Nuclear Magnetic Resonance Signal Parameters with a Multiprocessor Computer
Group: Electronic Systems and Signals Research Laboratory (ESSRL)

Nuclear Magnetic Resonance signal consists of a collection of exponentially decaying sine waves. The amplitude, frequency and decay of the signal are valuable parameters provide the chemist with a varied set of information on the particular chemical sample under investigation. However, frequently NMR signals are often noisy making it difficult to pick out the parameters especially for those chemical components in small concentration. Hence the idea of using signal processing methods to aid the estimation of the parameters. A likelihood function which is the squared error between the estimated signal and the signal collected is created. The estimated signal can be improved at each iteration by adjusting the current estimate of the parameters so a smaller square error is obtained. The difficulty involved is that this is a highly non-linear maximization problem. Often 15-20 sinusoids are involved per NMR signal with 60-80 parameters to vary at each iteration of the estimation. The resulting computational complexity is enormous.

The solution approach taken uses the Expectation-Maximization (E-M) algorithm to break the problem up into many smaller subproblems. Using the E-M algorithm, an expected data set is created for each sinusoid and the parameters estimated separately. In this way, instead of a large maximization problem with 80 parameters, we have 20 maximization problems each with 4 parameters. The resulting problems are simpler and, by employing a multiprocessor machine, each processor can work simultaneously on a maximization involving a single sinusoid instead of having a serial machine plodding through all the sinusoids one by one. For an NMR signal with 20 sinusoids a 20 times speed-up is achievable using a parallel computer containing at least 20 processors. The 64 processor Ncube computer has been used in this problem successfully. It both has enough processors for most of these problems of interest, and each processor is sophisticated enough to be programmed to handle the maximization algorithm employed.

Faculty: Michael Miller
Student: Tim Schaewe
Thesis Title: Reconstruction of Magnetic Resonance Images Using Maximum Likelihood Estimation
Group: Electronic Systems and Signals Research Laboratory (ESSRL)

The research undertaken concerns the development of a new method for the reconstruction of magnetic resonance images based on the maximum likelihood (ML) principle. The new method treats MRI image reconstruction as a parameter estimation problem, with the measured data modeled as a Gaussian process. An iterative expectation maximization (EM) algorithm has been derived which computes the ML solution and is suited for implementation on parallel computing architectures.

The EM algorithm has been implemented on the NCUBE and is currently being used to reconstruct images from simulated data. A single processor version of the algorithm was also implemented and used to calculate the speedup factor attainable with the parallel version.

The EM algorithm decouples onto concurrently solvable sub-problems in two different ways. The first corresponds to the reconstruction of the individual rows of the image, and involves no communication between processors. The speedup factor obtained from this decomposition of the problem is exactly equal to the number of processors assigned to the problem. Experimental results have been generated which support this claim.

The second form of parallelism relates to decomposition into separate sub-problems corresponding to the individual pixels within image rows. The implementation here involves some communication among processors, so the efficiency of the process is lower then the first method. A speedup factor of approximately 21 has been observed when the image row contains 64 pixels, with 64 processors assigned to the reconstruction. If one processor of a hypercube could be assigned to each pixel of a 64x64 image, these results indicate that a speedup of approximately 1,200 could be achieved.

Faculty: G-C Roman
Student: K. Cox
Thesis Title: Visualization of Concurrent Computations
Group: Concurrent Systems Group

The concurrent systems group is using the NCUBE to develop a transaction system. The transaction system supports a distributed database organized as a collection of tuples; its design and implementation is described in a recent technical report (WUCS-88-31) and is described briefly below. The transaction system will serve as a testbed and development base for the investigation of concurrent computations, using the shared dataspace model of concurrent computations (WUCS-88-34). In this role, the transaction system will also be used to investigate the visualization of concurrent computations (WUCS-88-27).

The overall design of the transaction system partitions the NCUBE nodes into two groups. One group is available for user processes; the other group supports a distributed database of tuples. A software package is provided whereby the user processes can access the database and insert, examine, modify, and delete single tuples. Tuples are accessed by content, and strong pattern-matching capabilities are built into the transaction system. Development work is underway to design and implement algorithms to permit more powerful, multiple-tuple transactions.

The transaction system will be used to investigate the shared dataspace model of concurrent computation. This model is characterized by the use of a single shared database through which all process communications occur; among languages using this paradigm are such expert system languages as OPS5 and OPS83, the Linda language, and Associons. It is our group's belief that the shared dataspace paradigm offers unique advantages for the visualization of concurrent computations, and we are conducting research in this area.

Faculty: G-C Roman
Student: Douglas Gill
Thesis Title: Implementing a Shared Dataspace Language on a Hypercube Multiprocessor
Group: Concurrent Systems Group

Shared Dataspace languages are a class of computer languages in which the primary means of communication among programs is a common content-addressable data structure known as a dataspace. These languages promise to be an effective vehicle for both large and small scale concurrent programming. Swarm is a particular shared dataspace language under development at Washington University. It is to be used for investigating the programming power of the paradigm as well as novel visualization techniques for understanding and debugging the complex behavior of highly concurrent programs.

Whatever the theoretical attractiveness of the shared dataspace paradigm and the Swarm language, their potential will not be realized unless acceptably efficient implementations can be demonstrated. Consequently, creating effective implementations is a key aspect of research and development into these languages. Such efforts fall into two broad categories: those targeting conventional general-purpose hardware and those attempting to match hardware with the particular features of a language (e.g. the Linda machine).

The purpose of this project is to demonstrate a prototype implementation of a significant subset of Swarm on a commercially available multiprocessing computer (the NCUBE-7). To date, a Swarm transaction system kernel for the NCUBE has been completed, as well as a minimal subset of the Swarm language embedded in C (referred to as Swarm/C). This project will significantly enlarge the set of features available in Swarm/C by permitting transactions consisting of multiple subtransactions and subtransactions accessing multiple tuples in the dataspace. The bulk of the language's query logic will be supported. Though it will fall short of a full implementation of the Swarm, this project will make available enough of its features to be used by those investigating concurrent algorithms, shared dataspace programming techniques, and program visualization methods.