

PDC
Center for Parallel Computers

Progress Report 1990-1991



The Cover Picture: The picture is a ray-tracing image of approximately 56,000 copper atoms, arranged in 8 identical cubes. Initially in a perfect cubic lattice, the copper atoms have been slightly disordered through a molecular dynamics simulation at a temperature of a few hundred degrees. The molecular dynamics simulation program was developed by Dr. Ole H. Nielsen at the Laboratory of Applied Physics at the Technical University of Denmark using the CM2 at PDC. This particular simulation was performed on the CM200 installed at the UNI-C computer center on the DTH campus. (Copyright © O.H. Nielsen, DTH. Reproduced with permission.)

The Cover Text: The text on the back cover is a testimony given recently to a U.S. Senate subcommittee.

PDC Paralleldatorcentrum
Royal Institute of Technology
S-100 44 Stockholm, SWEDEN
Telephone: +46-8-790 78 84, Telefax: +46-8-24 77 84
Email: info@pdc.kth.se

Publisher: Paralleldatorcentrum
Production: *Fredrik Hedman, Gert Svensson* (contents),
Per Hammarlund (contents & layout), *Jan Michael Rynning* (typesetting)
Editorial Committee: *Anders Lansner, Jesper Ooppelstrup, Lars-Erik Thorelli*
Linguistic revision: *Sven Westman*, Printed by: *Ekblads, Västervik, October 1992*
ISBN 91-7170-102-8

Foreword

The Center for Parallel Computers at the Royal Institute of Technology has now been active for two years. Our goal is to *stimulate research and spread information on the use of parallel computers*.

The center was formed in the beginning of 1990 to act as a focal point and national forum for both research on and use of parallel computers, including the new Connection Machine CM2. At the time we had a strong belief that massive parallelism would become an important technology, both in computer science and high-performance computing.

The Connection Machine, which in the beginning was considered a rather odd machine, is now used as a *general-purpose computer* which solves problems in several scientific and technical areas. This is due partly to an improved software and development environment, but more significantly to a better understanding of how to use massively parallel computers. Our user base is steadily expanding, with a majority of the new users aiming at solving real scientific and technical problems rather than just learning the new technique of parallel programming.

Problem solving on massively parallel computers often means that the user has to learn new methods and algorithms. However, most new users have little or no experience with massively parallel computers, making the transition from serial to parallel computers difficult. Optimally this could be taken care of by compilers, but we do not believe such automatic methods will be available in the foreseeable future. Hence, new users have to go through a learning process and as they become more experienced they can benefit from having a general forum in the area of parallel computing. The role of the Center for Parallel Computers is thus to provide help and advice for users and also to act as a catalyst. We are convinced that close relations between computer scientists and scientists from different application areas are crucial in order for this kind of user support to be successful.

The importance of massively parallel computers is no longer disputed. The supercomputers of the future will undoubtedly be massively parallel. In line with this, the majority of companies building supercomputers have announced massively parallel machines or plans to produce such machines. The importance of large-scale computing has also been emphasized by the High Performance Computing and Communication initiative in the USA.

In order to make it worthwhile for scientists and other users to invest work in porting their software to massively parallel machines, these machines must supply considerably better performance than sequential machines. Thus, it is important to keep the resources at the center competitive in order to maintain it as a flourishing meeting place for researchers from different fields.

“The role of the Center for Parallel Computers is ... to act as a catalyst.”

“The importance of massively parallel computers is no longer disputed.”

Our Connection Machine was recently upgraded to a CM200 including larger memory and a high-performance disk array. This will make it possible to solve larger problems and also to provide an even better development environment for a large, diverse user base. We are now planning to acquire a massively parallel machine of MIMD-architecture to complement the CM200. The architecture for this class of machines is today scalable up into the TFLOPS range and has a huge potential for the future.

The center strives to be a national resource for high-performance computing in scientific and engineering research, by providing access to state-of-the-art technology. Its creation was made possible by grants from Skandinaviska Enskilda Banken, FRN and NUTEK (formerly STU), which are gratefully acknowledged. Although the cost of equipment and operation is modest compared to conventional supercomputer facilities, we are dependent on continued national-level funding. This report shows that the first two years have been successful and resulted in significant research contributions in many fields. We therefore remain confident and enthusiastic in our approach to the future and the continued work with our present and future partners.

*“The center strives
to be a national
resource for
high-performance
computing in
scientific and
engineering research
...”*

The Board of the Center for Parallel Computers, January 1992

Contents

1	The Center for Parallel Computers	1
1.1	Background	1
1.2	Funding	1
1.3	Objectives	2
1.4	Organization	3
1.5	Hardware Resources	3
1.5.1	The Connection Machine	3
1.5.2	Other Hardware Resources	4
1.5.3	Upgrade of the Connection Machine	4
1.6	General Experience	5
1.7	Experience with the Connection Machine	6
1.8	Educational Activities	7
1.9	Summary of Projects	7
1.10	The Future	9
2	Neural Modeling And Computation	10
2.1	Implementing the SANS Algorithm	11
2.2	Multispectral Clustering using the CM2	11
2.3	Relaxation And Learning in Large Networks	12
2.4	Decorrelation And Sparsification	13
2.5	BIOSIM—Simulations of Realistic Neural Networks	13
2.6	A Neural Network Model of the Olfactory Cortex	15
3	Computational Fluid Dynamics	16
3.1	Return to Isotropy in Homogeneous Turbulence	16
3.2	Spectral Methods for Turbulent Flow	18
3.3	CFD Algorithms on Parallel Computers	19
3.4	High Order Finite Difference Methods for Hyperbolic Equations	19
3.5	Finite Volume Methods for Compressible Flow	20
3.6	The Characteristic Streamline Diffusion Method on the CM2	21
3.7	Unstructured Grid Methods for Compressible Flow	23
4	Biocomputing	24
4.1	Sequence Alignment on the CM2	24
5	Color Plates	27

6	Dynamical Systems	35
6.1	Harmonic Measure on a Fractal	35
6.2	Burgers' Equation with Scaling Initial Data	36
6.3	Interactive Analysis of Non-linear Mappings	37
7	Studies of Computer Architecture	39
7.1	Parallel Simulation on a SIMD Computer	39
7.2	Performance Evaluation of a Flow Control Algorithm	40
7.3	Performance Analysis of the CM2	40
8	Data Parallel Visualization And Picture Processing	41
8.1	Data Visualization—Volume Slicing	41
8.2	Data Parallel Algorithms for Picture Processing	42
9	Applications in Physics	43
9.1	Quantum Electronics: an Y-branch Switch for Electrons	43
9.2	High Temperature Superconductivity Models by Pair Correlations	45
9.3	Semiconductor Device Simulation—MINIMOS	46
9.4	Ground Vibrations Induced by Dynamic Surface Loads	46
9.5	Molecular Dynamics	48
10	Glossary	49
11	References	51
12	Index	55

1 The Center for Parallel Computers

This is a progress report of the Center for Parallel Computers (PDC: Paralleldatorcentrum) at the Royal Institute of Technology. It covers the period from the formation of the center in January 1990 to the end of 1991. The report describes the goals and the organization of the center as well as the user activities at the center during this time period.

An overview of the center follows in this section. User projects are described in Sections 2 through 9. The material dealing with the center as such has been written by the staff of PDC. Material describing user specific projects has been provided by the users as indicated in each section. The project descriptions have however been edited to fit into this report.

At the very end of the report there are three sections containing: a glossary of terms used in the text, an index of both terms and names, and a bibliography.

1.1 Background

Early in 1988 a group of scientists at the School of Computer Science and Engineering at the Royal Institute of Technology (KTH: Kungliga Tekniska Högskolan) applied for a grant to buy a massively parallel computer. A driving force behind the application was the belief that massive parallelism will become an important technology both in computer science and high-performance computing. Equally important, there was a need for the performance that could be delivered by such an architecture.

The computer market was surveyed and it was decided that Thinking Machines Corporation (TMC) offered the best choice with its Connection Machine system, CM2. Money was granted and during the fall of 1989 TMC installed an 8K Connection Machine CM2 at KTH.

At this time the idea came up to group together resources and activities around the CM2 and already existing parallel computers. Thus, what was to be called the Center for Parallel Computers was formed and inaugurated by Janne Carlsson, the President of KTH, on January 15, 1990.

In January 1991 PDC applied for an upgrade of the CM2 to a CM200. The application was successful and the upgrade was installed in December 1991.

1.2 Funding

The original grant of 10 MSEK for the CM2 was given by Skandinaviska Enskilda Bankens Stiftelse för Ekonomisk och Teknisk Forskning and the Swedish Council

for Planning and Coordination of Research, FRN. The grant of 4.72 MSEK for the upgrade came from FRN. The operational cost including staff has been covered by the Swedish National Board for Industrial and Technical Development NUTEK, formerly STU, and by the Royal Institute of Technology.

1.3 Objectives

Since the birth of computers, the capacity of even the most advanced and expensive computer systems has been regarded as a limiting factor by scientists in several areas. Nevertheless, these state-of-the-art computers have played a significant role in many scientific and technical areas.

The gigantic increase in capacity of computer systems is mostly related to advances in electronic devices like semiconductors. In this way the microprocessors of today are getting more and more powerful while retaining a reasonable cost-to-performance ratio. On the other hand, to produce processors with considerably higher speed by using special-purpose fast electronics is extremely expensive. As a consequence of this, one way to increase the speed of computer systems without extreme costs is to connect many standard, off-the-shelf processing units to form a parallel computer system. This solution seems to offer the best performance for the least amount of money. One difficult part of this approach is to build an interconnection network for inter-processor communication that is fast and general enough to support a multitude of different applications.

Almost all of the software of today has been designed for sequential computers. To write programs for a parallel system with hundreds or even thousands of processors requires other algorithms, programming methods and even new ways to attack a problem at a higher level. Programming these massively parallel systems is not a simple extension of programming a serial computer. Algorithms that work well on serial systems often turn out to perform poorly on parallel systems while out-of-date algorithms often contain a high degree of parallelism. Hands-on experience is necessary to learn how to “think in parallel”. The Center for Parallel Computers is meant to be a bridge between theory and practice by providing access to typical high-performance parallel computers and expertise on their usage, primarily to the Swedish academic research community.

*“... to be a bridge
between theory and
practice by providing
high-performance
parallel computers
...”*

1.4 Organization

The Center for Parallel Computers is headed by a board, which represents the research groups that funded the center and other Computer Science activities at KTH. Members of the board are:

Chairman	Lars-Erik Thorelli	Dean of the School of Computer Science and Engineering, Professor Computer Systems, TDS
Vice Chairman	Jesper Ooppelstrup	Director of research C2M2, NADA
	Fredrik Hedman	Application engineer, PDC
	Anders Hedin	Head of TDS
	Anders Lansner	Director of research SANS, NADA
	Yngve Sundblad	Chairman Computer Council, KTH
	Gert Svensson	Director, PDC

The center has a staff of three persons, or about two full-time equivalents. Fredrik Hedman works full time as Application Engineer, Gert Svensson half-time as project coordinator and Britta Svensson half-time as administrative assistant.

Johan Ihrén and Lars-Johan Liman from the TDS Systems Group administrate the UNIX front end computers and Per Hammarlund helps with the Connection Machine System.

1.5 Hardware Resources

PDC provides access to three classes of computer architectures: SIMD, MIMD with shared memory and MIMD with local memory. Most of the activities are however centered around the SIMD computer, the CM2.

1.5.1 The Connection Machine

The main resource of PDC is the Connection Machine. It is the only computer at the center that offers supercomputer performance. The Connection Machine at PDC is a SIMD computer with 8192 bit-serial processors. The machine is a “4 by 4” machine, meaning that it can be used as either an 8K machine or as two separate 4K machines—this improves the interactive user environment around the machine. The original machine had 256 Mbytes of main memory and 256 32-bit floating-point accelerators. The upgraded machine, the CM200 is also an 8K 4 by 4 machine. It has 1024 Mbytes of main memory, 256 64-bit floating-point units, and a 40% increase of clock frequency. The upgrade also included a Data Vault, i.e. a 10 Gbyte parallel disk array.

Included in the Connection Machine system is a Framebuffer. The Framebuffer is a high-speed graphical device that is connected in parallel to the Connection Machine. This makes it possible to display easily and quickly large amounts of data stored in the machine. The device is heavily used since it is possible to visualize a simulation as it is performed, e.g. a quantum-electronic simulation with the probability density of an electron color-coded. In the same room as the Framebuffer there are two graphical workstations, which offer users a convenient working environment when utilizing the Framebuffer.

The front end computers act as control processors in the CM2 system during program startup and execution. They are also the place for program development and compilation. The front end resources of the Connection Machine have been upgraded to meet the demands of a growing user base. The front end computers run the UNIX operating system and therefore provide a familiar user environment. The front end system is composed of two Sun computers, each having one Connection Machine interface. The two computers share disks, so users can employ either of the two machines. The center now has approximately 10 Gbytes of secondary disk space connected to the front ends. The front end computers are connected to the Swedish University Network and are easily accessible.

1.5.2 Other Hardware Resources

PDC has access to two other parallel computers. These are not meant for high-performance computing, but rather serve as typical examples of their respective architectures.

The Sequent Symmetry is a MIMD computer with a shared memory. The system at KTH has 10 Intel *iAPX386* processors connected by a common bus and memory. This method to build parallel computers is now well proven and has reached significant commercial success.

The transputer is a microprocessor specially developed to work in systems with many communicating transputer elements in parallel. The transputer microprocessor has communication channels implemented on the chip. Several network topologies can be implemented through re-connection. The experimental system at PDC has 4 transputers, runs the Trollius distributed operating system, and is connected to a Sun workstation.

1.5.3 Upgrade of the Connection Machine

In 1991 we applied for an upgrade of the Connection Machine's main memory and floating-point units. The motivation was the need to solve larger problems, be able to use the new CM Fortran compiler, and solve problems which demand higher precision.

Money was granted and negotiations with TMC started in the early summer. About this time TMC announced a model called the CM200 with a higher clock frequency. We were able to reach a favorable deal, which meant that we would get a CM200 with 1 Gbytes of main memory, 64-bit floating-point units, and a 10 Gbyte Data Vault—a parallel disk array.

Our experience of the upgrade is positive, e.g. a number of CM Fortran programs run 3–4 times as fast on the new machine after recompilation. The Data Vault will also enable us to configure our machine into an attractive combined production and development environment.

1.6 General Experience

The past year has been dominated by a rapid consolidation of the user community, where a number of research groups have reached excellent results. Currently there are some thirty ongoing projects which involve around forty regular users. Most of our users are situated in Stockholm, but we also have users from Uppsala, Göteborg, Linköping and Umeå as well as from Austria, Denmark, Finland, Switzerland and the USA. We have intentionally acted for a controlled expansion of our user base, enabling us always to provide good support despite the relatively small personnel resources of PDC. Given our experience so far, we foresee that the rate of expansion of our user base will continue to be high in the years to come.

The importance of massively parallel computers is no longer disputed. The supercomputers of the future are undoubtedly going to be massively parallel machines. Several manufacturers of supercomputers have also announced either new machines of this type or their intention to build massively parallel machines, e.g. CRAY, Alliant, and Convex. Also, the importance of large-scale computing has been accentuated by the High Performance Computing and Communication Initiative in the USA, which has attracted considerable interest and funding.

The possibility of *easy access* to powerful parallel computers is important both to stimulate the use of this new technique and because it enables scientists to attack important problems in science using these machines. Easy access means for the user to be able to access the machine without cost and without special security measures. A simple and helpful interactive environment, without long waiting times, enabling the user to try out new ideas quickly is especially important. We believe that we have succeeded in creating this kind of user friendly environment, and we are determined to continue this policy.

We feel that the organization of PDC as a small independent unit in close cooperation with different research groups has worked well. We would also like to stress the importance of being able to offer computers with enough capacity

“The possibility of easy access to powerful parallel computers is important ...”

to make worthwhile the effort for researchers in many different areas to use the machines. It can take a substantial effort to port a program to a new architecture; the gains therefore have to be sufficiently high.

1.7 Experience with the Connection Machine

SIMD computers such as the Connection Machine are especially successful in many scientific applications where one can represent the data set of the problem as a large number of identical elements; every element in the data set is then treated by its own processor. In a SIMD computer all processors execute the same instructions on their own local data. An example: by expressing the laws of nature for a system in local form one can model each portion of the system by assigning a data element to it and letting the processors “execute the same laws” for each element.

Experience shows that the Connection Machine can be successfully used in many more areas than was initially thought, the reason being the accumulation of knowledge of how to program a SIMD computer and the rapid evolution of compilers and system software on the CM2.

Scalability has become an important concept in the supercomputer field. By scalability one can mean different things:

- A hardware architecture which can be expanded within a wide range while remaining well balanced.
- The possibility to execute the same program on differently sized systems without changes to the program.
- The property of a program to execute efficiently on both small and large problems.

The CM2 was one of the first commercially available *scalable* computers. A CM2 or a CM200 can be acquired in sizes ranging from 4096 to 65,536 processors. Programs are independent of the size of the machine due to the use of so called virtual processors. The number of virtual processors can be much greater than the number of real processors in a machine. The mapping of the virtual processors to physical processors is automatically taken care of by system software.

In the Data Parallel Programming model, each data element is treated by its own processor. The Connection Machine has three complete programming languages which all support this model directly:

- CM Fortran, essentially a subset of Fortran90. The vector primitives in Fortran90 are used to implement the Data Parallel Programming Paradigm.

“The CM2 was one of the first commercially available scalable computers.”

- C* is an extension of ANSI C with integrated support for Data Parallel Programming.
- *LISP, built on Common Lisp, has been extended in the same way as C*.

Designing scalable programs is simplified in the Data Parallel Programming model, which can be implemented on both SIMD and MIMD machines. This makes it very attractive as a base for writing programs and designing algorithms.

“The Data Parallel Programming model can be implemented on both SIMD and MIMD machines.”

1.8 Educational Activities

The Connection Machine has been used for courses in the M.Sc. program at KTH and the Chalmers University of Technology. Many shorter seminars and four courses on the Connection Machine have been arranged:

Course	Date	Instructor	Stud.
Programming the Connection Machine	Apr 90	Adam Greenberg	59
Seminar on the new C* language	Oct 90	Mike Best	20
CM Fortran on the Connection Machine	Nov 90	Stephen Saroff	20
C* on the Connection Machine	Dec 90	James Frankel	53

1.9 Summary of Projects

The projects currently under way span a wide area of scientific disciplines: *Neural Modeling and Computation*, *Computational Fluid Dynamics*, *Biocomputing*, *Dynamical Systems*, *Computer Architecture Studies*, *Data Parallel Visualization and Picture Processing*, and *Applications in Physics*. Some of these projects have used the floating-point capabilities of the CM2 while others have profited from the bit serial hardware also present in the machine. A common theme for all projects is the large data sets used. Program and algorithm developments are the dominating activities, although some production runs have been done.

“A common theme for all projects is the large data sets used.”

The SANS group is interested in the design, theory, and application of Artificial Neural Network algorithms and in large-scale simulations of realistic neural networks. The neural-network algorithms have been successfully mapped to the massively parallel SIMD architecture of the Connection Machine. The computing power of the Connection Machine has allowed simulation of large networks, thus enabling more realistic applications and large simulation models.

The main activity at C2M2 is the investigation of mathematical and numerical problems associated with computation of solutions to partial differential equations. Most emphasis is put on the mathematical models for fluid flow, i.e. the

Navier–Stokes equations and their various simplifications. The Parallel Computational Fluid Dynamics group at C2M2 has set an easily defined goal: to assess the Connection Machine as a vehicle for 3D compressible flow computations with realistic geometries. Currently, the 8K CM performs approximately on a par with the CRAY X-MP. While this is acceptable, we feel that further advances can and must be made in this area.

In the area of Biocomputing, a program for protein sequence matching has been developed in collaboration with the Center for Biotechnology at Karolinska Institutet. The Connection Machine program performs a protein sequence matching with *interactive response* where the same operation takes hours on workstations.

Supercomputing is having a dramatic effect in the analysis of dynamical systems. It is now possible to attack large and complex systems, and in many cases the interesting properties of a system only emerge with a sufficient number of degrees of freedom. One type of systems contains many similar simple subsystems and is naturally well suited to data parallel modeling. Other types of systems have few degrees of freedom but parallelism can still be exploited by charting out parameter space with many parameter values running concurrently. The systems studied in the projects described here are of both types.

Parallel computers themselves are eminent examples of systems built from simple components which, due to the sheer number of components can exhibit complex behavior. Such architectures have been simulated and evaluated.

Image analysis is as an important application to which parallel processing can be applied. The Data Parallel Programming model is natural for digitized pictures. However, most algorithms have been developed for serial computers and new algorithms are needed to use massively parallel machines efficiently in this area. As a start, a representative set of operations used in image analysis has been reviewed for implementation on a data parallel machine. The creation of graphics to portray results of simulation and computing is crucial, and there is a need for easily adaptable programming environments which encourage experiments with new ideas for data visualization. We have attempted to use LISP as a high-level protocol in a client-server model for computing on the CM2 and visualizing on workstations.

The Data Parallel Paradigm is obviously well suited to computations on regular grids, and a number of other projects dealing with continua in such diverse fields as 3D elastic waves, semiconductor device modeling, high temperature superconductivity models, and quantum mechanical electron transport have been successfully undertaken. The majority of the applications considered deal with the solution of partial differential equations by grid-based methods and explicit time stepping procedures, or by simple iterative techniques for sparse systems.

In addition, a many-body simulation code for short-range molecular dynamics has been developed.

1.10 The Future

The supercomputers of the future have an enormous potential. They will have an ever increasing impact on society at large and in particular on computational sciences and engineering design calculations. Efficient use of these machines will require in-depth knowledge in many different fields, and PDC has the ambition to be both a meeting place and a focal point for people involved with these machines.

In line with this, PDC is planning to acquire a large MIMD computer in the 92/93 time frame and to participate in the Swedish High-Performance Computing Network project (SHPCnet). This will allow us to begin exploring the possibilities offered by high-speed networking and also enable us to spread knowledge of MIMD techniques nationally, as is already being done with our SIMD computer. An important requirement of this new MIMD computer, on top of the fact that it has to deliver competitive computational power, is that it should possess an integrated programming model to express parallelism, especially the Data Parallel Programming Model. We view a MIMD computer, which fulfills this requirement, as an important continuation and a natural extension to the work that has already begun on the CM2 and CM200.

“PDC has the ambition to be both a meeting place and a focal point for people involved with the supercomputers of the future.”

2 Neural Modeling And Computation

The SANS (Studies of Artificial Neural Systems) project at NADA, KTH is aimed at designing general-purpose adaptive systems with highly autonomous operation for technical applications. It is also engaged in realistic simulations of biological neural networks together with neurophysiologists at Karolinska Institutet [Lansner, 1991].

A neural network is a collection of summing units, neurons, that communicate with each other through directed connections of variable strength, synapses. Each unit sums the weighted input from other units and updates its own output accordingly. In feed-forward networks there are no loops in the connections between units; the computation flows from input to output. Recurrent neural networks on the other hand typically have only one layer and the output is connected back to the input. When a recurrent network is used, the input pattern is applied at its input and the network is then left to stabilize iteratively by using its output as new input—this process is usually referred to as relaxation.

Our interest in the CM2 is to investigate what can be gained from mapping our recurrent artificial neural network (ANN) algorithms to massively parallel hardware and also to do large-scale simulations of realistic neural networks.

Both for neurocomputing applications and for biologically realistic simulations the neural networks often have to contain many units in order to display interesting behavior. In the ANN case, large networks enable the researcher to escape from toy problems and turn to real world applications. In realistic simulations one can study the behavior of the network at the system level—for instance looking for emergent properties of large collections of neurons. The actual number of neurons needed may vary for different applications and simulations, but the computational power even of the high-end workstations often falls short.

It has been pointed out many times that sequential computers are quite inefficient for simulation of neural networks, as the networks themselves are inherently parallel. It is much more likely that the models can be implemented efficiently on parallel hardware. Actually, engaging in parallel implementation is important for a number of reasons. Clearly, it is always important to have access to high-performance computational resources. It allows us to investigate the *scaling properties* of our algorithms as problem sizes increase. In addition, it opens up the possibility of escaping from “toy problems” to some *real world applications*, as mentioned above. A further, perhaps less obvious reason, is that the use of parallel hardware puts relevant constraints on *algorithm development*. In this way we avoid devising inadequate solutions restricted by embedded sequential sections.

The projects that are described below are vital to our research; the application

“neural networks are
inherently parallel
...”

for funding of a massively parallel machine was written precisely to enable us to carry out these projects. In short, the fact that we have access to a massively parallel computer in-house, is a crucial factor for continuing our line of research and advancing into even more interesting applications.

2.1 Implementing the SANS Algorithm

Björn Levin

SANS, Royal Institute of Technology

At the heart of our artificial network applications is a recurrent Hopfield-like algorithm, “The SANS Algorithm,” for training and executing networks. The algorithm was designed by the SANS group at KTH and has been used extensively for simulations on workstations. Several different CM2 versions of this basic model have been developed. They all implement the same computations in fully connected recurrent networks, but differ in the way the task is mapped onto the CM2. The main purpose of this work has been to study how the algorithm is affected by parallel hardware, but it has also given the group a tool for fast simulations of large networks. These implementations can run fully connected networks of about 1000 units [Levin, 1990, Levin, 1991].

2.2 Multispectral Clustering using the CM2

Björn Levin

SANS, Royal Institute of Technology

As a result of earlier work, Section 2.1, on implementing the SANS algorithm on the CM2 we decided to look into implementations of smaller networks, i.e. networks with less than a hundred units, where the parallelism of the CM2 is taken advantage of by solving several problems simultaneously using a *large* number of neural networks all sharing the same connection matrix. (This is in line with the approach followed by Singer when implementing Back Propagation on the CM2 [Singer, 1990].)

The classification of satellite images supplied by the Swedish Space Corporation was chosen as a test problem, which is solved by running 16,384 small neural networks in parallel. No geometric information is used, e.g. the fact that roads are narrow elongated structures, and thus the classification task can be split into as many independent problems as there are pixels in the image. The pixels themselves are characterized by three intensity bands corresponding to different frequencies of light, each with 256 levels. Patterns are formed using around

*“using 16,384 small
neural networks in
parallel ...”*

10 sensors per band. To introduce the notion of surroundings the patterns in a 3×3 neighborhood are combined during learning. This is also done for the patterns taken as starting points during recall. So far, no manual classification or ‘ground truth’ is incorporated. As can be seen from Figure 7 on page 27, the result of the classification process is a drastic decrease in the number of different colors in the image, with each color representing a common class or ground type. The number of classes can be chosen by adjusting some parameters of the ANN algorithm.

As a measure of the quality of our results we compared them with a classification using a maximum likelihood approach (this classification was produced by the supplier of the images). The result was fair and has encouraged further investigation into the issue. Particularly interesting are different forms of automatic recoding schemes and the introduction of complex units.

2.3 Relaxation And Learning in Large Networks

Per Hammarlund

SANS, Royal Institute of Technology

Implementing the basic Bayesian model on the CM2 was initially explored in [Levin, 1990] as described in Section 2.1. Now we continue to implement the SANS algorithm with different mappings of the data on to the CM2. We also try to rewrite and modify the basic algorithm to fit the architecture of the CM2 better.

It is very often said that ANNs have inherent parallelism. In fact, there are numerous ways of achieving parallelization. With each parallelization there are a number of possible data mappings onto a specific architecture. In this project we have been looking closely at optimizing the usage of the CM2. Different possible parallelization strategies have been tried and evaluated. These implementations have in some cases involved writing microcode, CMIS, for the CM2 to get support for primitives that are not present in the instruction set of the machine and also to optimize instructions for the special cases of our implementation. This project has also involved optimizing the storage of very large, sparse activity patterns both on the CM2 and on the front end computer.

On an 8K CM2 with 256 Mbyte main memory we can run fully connected networks with 8K units and sparsely connected networks with up to about 32K units. In the fully connected case learning of a couple of hundred patterns of size 8K is a matter of seconds, and relaxation of one single pattern is well below one second. In the sparsely connected case, learning as well as relaxation is slower, but on the other hand the number of units one can use in the network is much bigger.

“there are numerous ways of achieving parallelization”

With the upgrade of the KTH machine to 1 Gbyte memory, we are able to run up to 16K fully connected units.

Even though the CM2 is a general-purpose massively parallel computer, it has its particularities. In the second part of this project we have allowed ourselves to make changes in the SANS algorithm to make it fit better the particular architecture of the CM2. This work is only possible with a good understanding of both the hardware and the mathematics of the SANS model. Using the understanding of the architecture, the algorithm has been rewritten to include only primitives which we know can be implemented efficiently.

2.4 Decorrelation And Sparsification

Björn Levin, Peter Sjögren

SANS, Royal Institute of Technology

We have started implementing a method for decorrelation and sparsification of patterns on the CM2. Decorrelation and sparsification are both means of reducing the memory requirements for the network connections by taking a priori information into account and neglecting weak couplings. Complementing the large recurrent networks mentioned above in Section 2.3 with this kind of pre-processing may open up interesting applications, for instance in the areas of information retrieval and molecular biology.

Peter Sjögren's Masters Thesis work on the CM2 implemented parts of these ideas [Sjögren, 1991]. The program was tried out on the problem of matching protein sequences, in cooperation with the Biocomputing Project (See Section 4) with the same aim running on the same CM2, but using completely different techniques. Despite these differences a quantitative correspondence was achieved with the test problem investigated. We are currently carrying out tests with a larger number of proteins to see how these results scale with increased problem size. We believe that our neural network approach to the protein matching problem has the potential to run very fast.

2.5 BIOSIM—Simulations of Realistic Neural Networks

Per Hammarlund, Björn Levin

SANS, Royal Institute of Technology

The BIOSIM program for the CM2 [Levin *et al.*, 90, Hammarlund *et al.*, 1991] is aimed at giving the researcher the opportunity to run very large, biologically

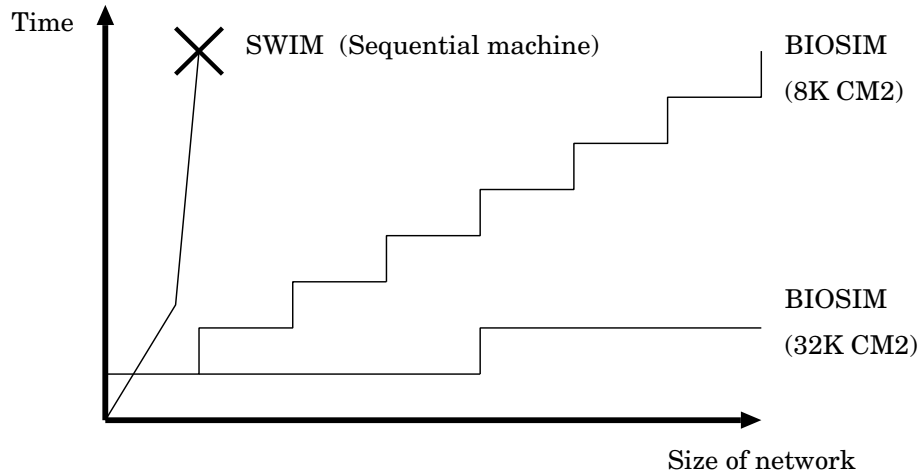


Figure 1: This figure shows the qualitative relation between the execution times of BIOSIM and SWIM (running on a workstation). We can see that the execution time of SWIM grows linearly as the number of simulated neurons grows—up to the capacity of the workstation that SWIM is running on. The execution time of BIOSIM grows in increments, as a step function, as the vp-ratio increases with increasing number of simulated neurons.

realistic simulations. On an 8K CM2, BIOSIM is capable of simulating one thousand cells and one million connections or 64 thousand cells and 256 thousand connections. Simulation times are typically in the order of minutes to one hour per second simulated time, using a time step of about $50 \mu\text{s}$. This makes BIOSIM one of the most, if not *the* most capable general-purpose simulator for large, biologically realistic neural networks.

BIOSIM is integrated with SWIM, an interactive package for definition and simulation of realistic neural network models [Ekeberg *et al.*, 1990, Ekeberg *et al.*, 1991a, Ekeberg *et al.*, 1991b] in such a way that the transition is easy from workstation simulation of smaller networks, or parts of the complete network to simulating the complete, full size network. The researcher need not bother with the particularities of the CM2, the interface is still SWIM and its specification language.

The BIOSIM program has been written in such a way as to take best possible advantage of the inherent SIMD-type parallelism in this type of simulation and also to make the most efficient use of the CM2’s floating-point hardware and fast communication network. Informally speaking, this implies solving differential equations of the same form in parallel, and also handling synaptic communication in parallel.

“a one hundred segment lamprey spinal cord model and a model of cortical associative memory ... have been simulated ...”

With the aid of the BIOSIM CM2 simulator, a one hundred segment lamprey spinal cord model [Lansner *et al.*, 1991] and a model of cortical associative memory comprised of 2000 cells and over 200,000 synapses [Lansner and Fransén, 1991] have been simulated. These are some of the largest simulations of their kinds. Figure 8 on page 28 shows an example of the cortical associative memory functions simulated. Figure 9 on page 30 shows a simulation of the lamprey spinal cord model.

At present, apart from internal use, a research group at the University of Illinois, Urbana, USA is using the simulator. We plan to improve the user interface, interactivity and the numerics of this simulator in the near future.

2.6 A Neural Network Model of the Olfactory Cortex

Hans Liljenström

SANS, Royal Institute of Technology

The project is a prototype for much of the work done at SANS. It integrates artificial neural network techniques with biological data and aims at understanding real neural networks, in this case the cortical parts of the brain. Several researchers have found oscillations in these regions. The goal of this project is primarily to investigate whether such cortical oscillations have any biological significance, e.g. for learning and memory. In the simulations we use a model of the olfactory cortex, the “odor brain”. It is used as a model system because it exhibits well characterized oscillations, in the 5 Hz as well as in the 40 Hz range, and has a relatively simple structure, which closely resembles that of neural network architectures used to model associative memory.

The complexity of the model network lies between those of simple Hopfield nets and detailed biologically realistic simulations. External inputs exist for excitatory and feed-forward inhibitory units. Network parameters are chosen within their physiological ranges. Network connection weights are adaptive, to allow learning and memory functions to be studied. Earlier computer simulations have shown that this model is capable of reproducing most observed and previously simulated dynamic behavior of the olfactory cortex. Implementing the model on a massively parallel machine yields a considerable gain in simulation time as well as network size. The layered 3-dimensional model architecture takes advantage of the data structures offered by the C* language. Both grid and general communication between the nodes, which ideally are represented by one processor each, are intended to be utilized optimally in the model.

“... investigate whether cortical oscillations have any biological significance for learning and memory.”

3 Computational Fluid Dynamics

“At present, the only road to progress in the understanding of turbulence seems to be through massive computer simulation ...”

Since the days of the first electronic computers the modeling of flows has been one of the major applications, indeed in many respects a pacing one. This is so because of the strong non-linearity of the equations governing the complex and fascinating patterns of flow and the intricate mechanism of turbulence. At present, the only road to progress in the understanding of turbulence seems to be through massive computer simulation supported by well-designed experiments in wind and water tunnels.

The Computational Fluid Dynamics (CFD) projects under way on the CM2 are using different methods, such as spectral (Sections 3.1, 3.2), finite difference (Sections 3.3, 3.4), finite element (Section 3.6), and finite volume methods (Sections 3.5, 3.7). While we cannot yet point to startling new discoveries made by CM2 computations, the importance of having accessible supercomputing resources has had an important impact on the work. It has stimulated cooperation with other groups, both in Europe and the USA, in particular with the Computational Fluid Dynamics group at Ecole Polytechnique Fédéral de Lausanne and the George Washington University group working with unstructured grids for CFD problems.

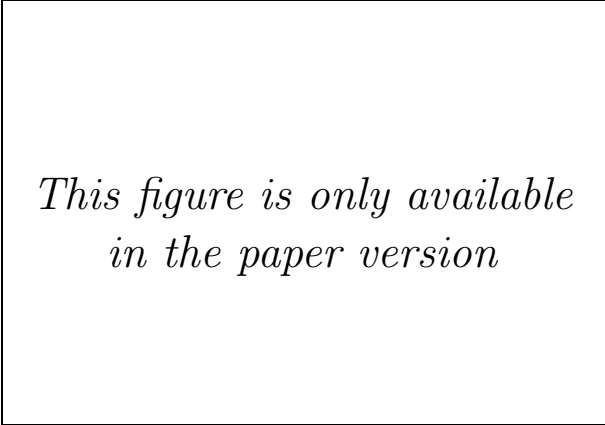
3.1 Return to Isotropy in Homogeneous Turbulence

Magnus Hallböck

Department of Mechanics, Royal Institute of Technology

With the advent of faster computers the experimental study of turbulent flow can be complemented by simulations of the time-dependent Navier–Stokes equations. It is possible to model flows with Reynolds numbers up to $O(1000)$, but this requires the largest computers available.

One of the goals of such a study is to create simpler models of turbulence which can be used in practical CFD. When the Navier–Stokes equations are time-averaged, the influence of the fluctuations appear in the equations for the average quantities as the so-called Reynolds stresses. These, in turn, satisfy transport equations where the sources are pressure-strain correlations.



*This figure is only available
in the paper version*

Figure 2: The Rotta coefficient $C1$ (defined in the text below) as estimated from statistics collected in computations of turbulent flow at various Reynolds numbers Re . A trend in $C1$ with increasing Re is evident. As indicated by the straight lines, this trend seems well established, but the computational experiments do not reveal any more detailed variations of $C1$ with Re .

An often used assumption is that the pressure-strain correlations are approximately proportional to the product of mean viscous dissipation and the off-diagonal parts of the Reynolds stresses,

$$\Pi_{ij} = -C1 \times \epsilon \times a_{ij}, \quad (1)$$

$C1$ is referred to as the Rotta constant, although it is not really a constant. In Figure 2 we have collected values of $C1$ computed from simulations of homogeneous turbulence at various Reynolds numbers. The initial anisotropy is created by setting the starting conditions suitably to correspond to a turbulent flow that has undergone an axial stretching, such as may be brought about by the contraction in a wind tunnel, see Figure 3. The following relaxation to isotropy is what the simulations model. As seen from the graph, simulations up to $Re = 100$ were performed on the 8K CM2 at KTH. Larger simulations were later run on 32K machines. It is also evident that, although one sees a trend in $C1$ with increasing Re , the results must be used with caution since there is little reason to believe that the relationship is so simple.

The 8K CM2 runs on a par with a one-processor CRAY X-MP on these problems, although precise comparisons are difficult. FFT routines are crucial for the efficiency, and real-to-complex FFT routines are currently under development (Section 3.2) which will halve the memory requirements and speed up the execution. [Hallbäck *et al.*, 1991]



This figure is only available in the paper version

Figure 3: Turbulent flow passing through a contraction. The turbulent small-scale flow is isotropic before the contraction. In the contraction, the large fluid volumes are stretched in the axial direction. When the eddies are stretched they become more unstable and break up, and the flow gradually returns to isotropic turbulence. In the computations, the stress components and pressure fluctuations are recorded and later correlated.

3.2 Spectral Methods for Turbulent Flow

Lars Malinowsky

C2M2, Royal Institute of Technology

Spectral methods give very high accuracy for PDE problems where the solution is smooth, and they have been the methods of choice for computing unsteady viscous flows with resolution at all scales, i.e. the direct simulation of turbulent flows. The methods rely on the Fast Fourier Transform, for which the Connection Machine Scientific Subroutine Library (CMSSL) has efficient routines using complex arithmetic. In the first phase of the project, the spectral method was implemented and tuned for the CM2 in CM Fortran. In the present phase, improvements to the FFT routines are being developed which will save memory and increase execution speed. These developments are done in microcode (CMIS) and will become part of CMSSL when completed. The high-resolution methods require 64-bit floating-point arithmetic precision, and with the upgrade to a CM200 we will be able to run 256^3 problems on the 8K machine at KTH at speeds expected to exceed those of the CRAY X-MP. [Malinowsky, 1991]

“With the upgrade to a CM200 we will be able to run 256 cube problems ...”

3.3 CFD Algorithms on Parallel Computers

Pelle Olsson

Department of Scientific Computing, Uppsala University

In this project we have studied the implementation of linear boundary conditions on a data parallel computer, such as the CM2. We propose explicit algorithms for the Euler equations and the Navier–Stokes equations. Higher-order finite difference methods particularly suited for data parallel computers have also been investigated.

For any multi-processor computers, and in particular data parallel machines, it is important that inter-processor communication is reduced as much as possible. Also, memory requirement must be considered, since the difference stencils must be stored locally in each virtual processor for maximum performance. We have developed stable 3rd, 4th, and 5th order accurate difference operators with a minimized amount of communication, and studied a class of difference operators which can be shown to satisfy an energy estimate, even if the boundary is not smooth. Thus, corners and edges are allowed in multi-dimensional domains.

On serial computers the time spent evaluating boundary conditions is negligible compared to the time needed to update the interior. This is in general not true for data parallel computers, where the implementation of the boundary conditions calls for particular attention, since concurrency, hence performance, can be lost if improper or ill-designed algorithms are used. We have also demonstrated that the boundary conditions can be viewed as a projection operator at each boundary point. The explicit structure depends on the boundary condition (inflow/outflow, solid wall, supersonic/subsonic flow) and the geometry. This projection operator is extended to the interior grid points. By allocating a projection array at each virtual processor (grid point), it is possible to embed the boundary conditions such that all boundary conditions may be concurrently updated. The conclusion remains true even for highly irregular boundaries. No conditionals are required for the boundary correction; in fact, the conditionals have been embedded in the projection array. This technique is particularly useful when computing steady state solutions, where the projection array can be initialized and then reused. Each boundary update is then performed as a matrix-vector multiplication. The present implementation runs at about 200 MFLOPS on a 512×512 grid.

3.4 High Order Finite Difference Methods for Hyperbolic Equations

Jennifer Chang

Department of Mathematics, UCLA

It has been conjectured that high-order finite difference methods should be competitive with spectral methods for most problems in CFD. A careful numerical comparison has been made between the fourth-order and sixth-order finite difference method and the pseudo-spectral method for some 1D and 2D hyperbolic problems. One common feature of these problems is that a smooth solution can become nearly singular as time evolves, and a thin shear layer or shock may develop. This makes it very difficult to resolve the profile accurately on a computational grid. The main conclusion of the study is that the fourth-order difference method can achieve a given error tolerance more efficiently than the pseudo-spectral method for this kind of problem. This is especially so on the Connection Machine, due to the special communication structure. For example, the fourth-order difference method runs about 30 times faster than the spectral method on a 256×256 grid, yet they produce roughly the same order of numerical error at this resolution level. This is very encouraging and brings new hope that we may be able to compute the incompressible Navier–Stokes equations much more efficiently.

3.5 Finite Volume Methods for Compressible Flow

Magnus Bergman

C2M2, Royal Institute of Technology

Per Wahlund

Scientific Computing, Uppsala University

“... the current industry standard for aerodynamic computations of aircraft configurations.”

The multi-block technology was introduced to handle complex geometry and is the current industry standard for aerodynamic computations of aircraft configurations. The computational grid is composed of a set of blocks, each with a structured grid. The blocks may adjoin in any fashion. Such a basic decomposition is suitable also for load balancing in a MIMD environment, but for SIMD architectures the unstructured communication patterns associated with the block boundaries require attention. The boundary-condition difficulties (Section 3.3) are compounded by the unstructured multi-block assembly. Our goal is to develop a 3D code for compressible viscous flow, which is optimized for the CM2 architecture but reasonably portable by virtue of implementation in Fortran90. The results to

date show that a 2D version runs on a par with or faster than one processor of the CRAY X-MP. On a simple block configuration, central parts of the code run at 250 MFLOPS on the 8K CM2 at KTH, and overall at 98 MFLOPS. The CM200 recently installed runs a somewhat modified code at more than 200 MFLOPS overall. [Sawley and Bergman, 1991, Wahlund, 1990, Bergman *et al.*, 1991]

3.6 The Characteristic Streamline Diffusion Method on the CM2

Claes Johnson, Peter Hansbo, Kenneth Eriksson

Department of Mathematics, Chalmers University of Technology

We have investigated a characteristic streamline diffusion (CSD) finite element method (FEM) applied to pure time-dependent convection model problems. These are prototypes of the incompressible and compressible Euler and Navier–Stokes equations of fluid dynamics. The main difficulty with these problems is to combine numerical stability with small added non-physical diffusion. Figure 4 shows the initial condition and the solution by CSD after pure convection in a rotational velocity field.

As may be inferred from Figure 4, the level of spurious diffusion is below plotting accuracy. The method is basically a standard Galerkin FEM combined with a small least-square term added to enhance stability. A particular feature is the use of finite elements both in time and space, continuous in space and discontinuous in time at discrete time-levels. The mesh is adaptively aligned with the characteristics, locally in time, during one or more time-steps. The mesh may be changed arbitrarily at discrete time-levels, in which case data is transferred from one level to the next through a built-in L_2 projection.

The main technical problem lies in the projection between non-matching grids. We have considered a mesh with fixed topology deforming over one or several time-steps, after which we return to the original mesh. The projection between the deformed and initial mesh has been performed with two different approaches:

- A method specialized for structured meshes, based on a subdivision of the domain into squares.
- A general method based on searching for the neighboring element closest to the sought interpolation point.

Preliminary conclusions show that for reasonably small time steps, ($CFL < 2 - 3$) the general search algorithm is less than one order of magnitude slower

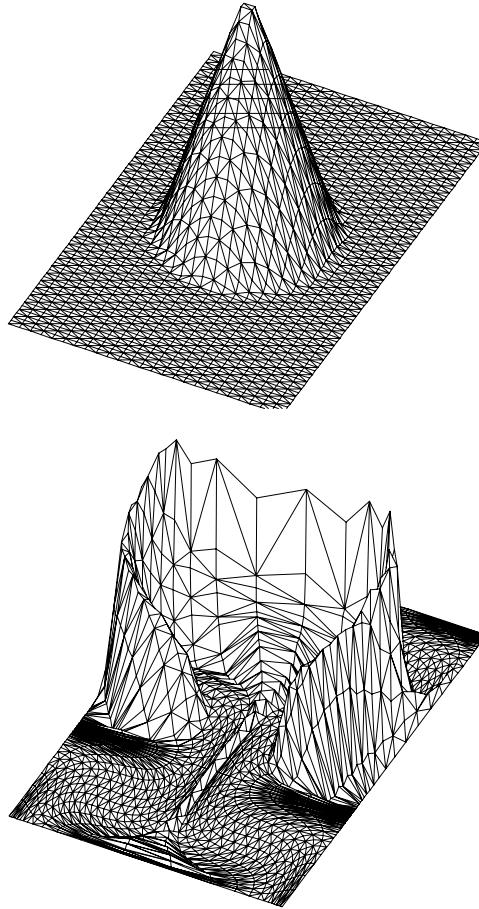


Figure 4: Advection of concentration profile in a rotational velocity field with a strong shear flow component. The upper figure shows the concentration at time zero. The advection process is computed by the streamline diffusion finite element method, and the field at a later time is shown in the lower figure. Note the good resolution of the sharp gradients without unphysical over- and undershoots.

than the specialized one, but for larger CFL (Courant, Friedrichs, and Lewy Condition) numbers there may be a difference of several orders of magnitude. This illustrates the basic difference between the two inter-processor mechanisms on the CM2: the fast grid-oriented NEWS communication and the slower general-purpose router communication. [Svennberg, 1991]

3.7 Unstructured Grid Methods for Compressible Flow

Rainald Löhner, Dorothee Martin

George Washington University

Magnus Bergman

C2M2, Royal Institute of Technology

A computational grid in which the connectivity of a computational cell with its neighbors varies over the grid is usually called an unstructured grid. Unstructured grids offer more flexibility in adaptive mesh refinement than structured grids do. Computing methods for such grids require storing and inquiring information about neighbor cells for each cell, and the resulting indirect addressing schemes are hard to parallelize efficiently on SIMD architectures. For this reason, much effort is devoted to creating automatic mesh generators for unstructured grids and to finding efficient ways of implementing the computational schemes on parallel machines.

In the project, a 2D inviscid flow solver for triangular grids was moved to the CM2 in CM Fortran. The computations for finite volume-like schemes usually work with data that are associated with nodes or cell faces, but for this implementation it was found that a node-cell edge structure would lead to better communication patterns. The redesign should be of benefit also on vector machines and machines with large cache memories. Presently, the first version of the code is working, but boundary conditions etc. remain to be properly recoded for the CM.

4 Biocomputing

Proteins and nucleic acids—the building blocks of the genetic material—can be thought of as sequences of characters, with a 20-letter alphabet for proteins and a 4-letter alphabet for nucleic acids. Currently, the amount of sequence data is growing by a factor of ten every five years, and this rate will increase as the Human Genome Project gets well underway.

With the rapidly increasing amount of sequence data there is a pressing need for faster and more sensitive means of scanning sequence data banks for similarities between newly determined and already published sequences. If a new sequence can be aligned with some of those already known, many years of experimental work can be avoided since the data of the known sequences can be used as a starting point for further investigations of the newly found sequence. [von Heijne, 1988, Coulson *et al.*, 1987, Argos *et al.*, 1991] Recent developments in computer architecture and hardware for parallel computing have led to vast increases in computational speed for certain classes of problems. Data bank scanning is well suited to parallelization, and more sensitive comparison algorithms may be executed in acceptable times.

“Currently, the amount of sequence data is growing by a factor of ten every five years, ...”

4.1 Sequence Alignment on the CM2

Gunnar von Heijne

Center for Biotechnology, Karolinska Institutet
Fredrik Hedman, Erik Wallin, Christian Wettergren
 PDC, Royal Institute of Technology

We have implemented the widely used Needleman–Wunsch sequence alignment algorithm on a CM2 computer with 8K processors. The Needleman–Wunsch algorithm is known to be one of the most sensitive methods available for protein and DNA sequence comparisons. [Argos *et al.*, 1991] Although slow on sequential machines, it is well suited to parallelization when large data banks are to be scanned.

Programming was done in C* to allow easy porting to future CM models. A user interface based on a client-server model which allows remote access to the CM2 has also been written. A typical scan of the *SwissProt* protein sequence Data bank (release 18; some 20,000 sequences, 7×10^6 residues) takes approximately 1 second per amino acid in the query sequence. The overall speed grows linearly with the size of the machine: a full, 64K processor CM2 should attain a speed of some 5×10^7 residue-residue comparisons per second. With a query sequence of 100 residues, this gives a theoretical turnaround time of about 10–15 seconds.

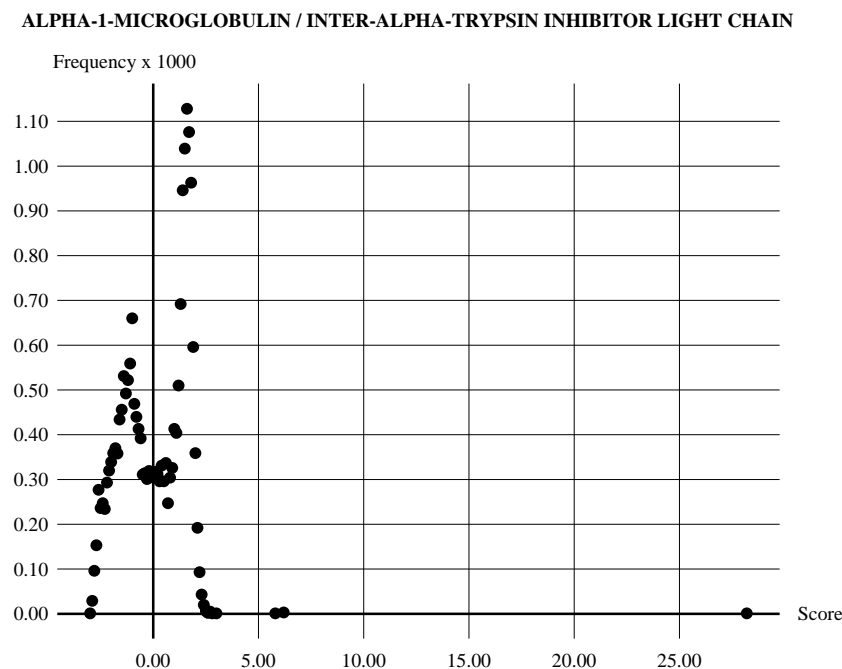


Figure 5: The normalized score distribution of a typical run with the whole database. The highest score, the one above 25.0, is that of the protein sequence against itself. The scores with high frequency below 5.0 represent statistical noise, they are most likely not evolutionarily related to the target protein. The interesting scores in this distribution are the two found at about 6.0.

This indicates that already today a full CM2 can search the very large databases of the future in reasonable time.

Our ambition is to make this service available to the Swedish molecular biology community via SUNET. [Andersson, 1991] The user interface, although functional, is not yet fully developed and must be improved. Also, we plan to implement one or two other protein sequence alignment algorithms (Section 2.4) on the CM2 before we start signing on new users. Nevertheless, we feel that we have already solved the basic problems and demonstrated the feasibility of our approach. [Wallin, 1991, Wettergren, 1991]

“Our ambition is to make this service available to the Swedish molecular biology community via SUNET.”

```

cmd> match 13371
wait...
  .
  .
  .
 0 M R S L G A L L L L S A C L A V S A G P V P T P P D N I Q V Q E N F N I S R I Y G K W Y N L A I G   50
  .
  .
  .
 0 ..... 0
  .
  .
  .
50 S T C P W L K K I M D R M T V S T L V L G E G A T E A E I S M T S T R W R K G V C E E T S G A Y E K   100
  .
  .
  .
 0 ..... 0
  .
  .
  .
100 T D T D G K F L Y H K S K W N I T M E S Y V V H T N Y D E Y A I F L T K K F S R H H G P T I T A K L   150
  .
  .
  .
 0 ..... 0
  .
  .
  .
150 Y G R A P Q L R E T L L Q D F R V V A Q G V G I P E D S I F T M A D R G E C V P G E Q E P E P I L I   200
  .
  .
  .
 0 ..... 0
  .
  .
  .
200 P R V R R A V L P Q E E E G S G G G Q L V T E V T K K E D S C Q L G Y S A G P C M G M T S R Y F Y N   250
  .
  .
  .
 0 .....K E D S C Q L G Y S Q G P C L G M F K R Y F Y N   24
  .
  .
  .
250 G T S M A C E T F Y Y G G C M G N G N N F V T E K E C L Q T C R T V A A C N L P I V R G P C R A F I   300
  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 24 G T S M A C E T F Y Y G G C M G N G N N F P S E K E C L Q T C R T V Q A C N L P I V R G P C R A G I   74
  .
  .
  .
300 Q L W A F D A V K G K C V L F P Y G G C Q G N G N K F Y S E K E C R E Y C G V P G D G D E E L L R F   350
  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 74 E L W A F D A V K G K C V R F I Y G G C N G N G N Q F Y S Q K E C K E Y C G I P G E A D E E L L R .   123
  .
  .
  .
350 S N   400

123 .. 171
Ok.
cmd>

```

Figure 6: The matching of the target protein sequence with the specified protein sequence. The two sequences are aligned with each other, with a match row in between. Each letter stands for one amino acid, a dot denotes a gap, and a vertical bar corresponds to matching amino acids in the two sequences. To keep track of the location within each sequence, the numbers of the first and last amino acid on each row are also shown.

5 Color Plates

Before clustering

This figure is only available in the paper version

After clustering

This figure is only available in the paper version

Figure 7: The pictures show the original satellite image (upper) and the classified result (lower), using the SANS neural network algorithm as a clusterer. In this experiment 65,536 identical networks are run independently, each looking at a 3×3 pixel area. The result contains 4 classes, but this number can easily be varied, letting the network split or join classes. (See Section 2.2 on page 11.)

Figure 8: A simulation of a cortical associative memory model. The network consists of 169 micro-columns (13×13), each of which is composed of 12 nerve cells (3×4). Within the micro-column 10 excitatory (positive) cells connect locally to one another. The 2 inhibitory (negative) cells (lower right) suppress the local excitatory cells. Between micro-columns a sparse connectivity projects from excitatory cells to excitatory or inhibitory cells. The ring shaped pattern is activated and the resulting firing of nerve impulses is monitored. Cell activity is color coded from purple, not active, via green to yellow, active. The first picture displays the activation of the cell assembly (pattern). The second and third pictures show how the activation spreads and peaks. Decay of activity starts in the fourth picture. The total time simulated was 300 ms. (See Section 2.5 on page 13.)

Dia 16

1

This figure is only available in the paper version

Dia 10

2

This figure is only available in the paper version

Dia 09

3

This figure is only available in the paper version

Dia 01

4

This figure is only available in the paper version

Multisegment simulation.

This figure is only available in the paper version

Figure 9: Simulation of the neuronal network in the lamprey spinal cord responsible for swimming. Here we simulate 20 spinal segments with a total of 120 realistically modeled cells and 240 synapses. 100 segments, consisting of 600 such cells and 1200 synapses, have been simulated. The intra-cellular potential is color coded, ranging from purple (-70 mV) to yellow ($+35$ mV, spiking). The horizontal axis shows time. Spinal segments 1 (top) to 20 are shown along the vertical axis. The time lag seen is responsible for the wave like swimming motion characteristic of the lamprey and other fish. (See Section 2.5 on page 13.)

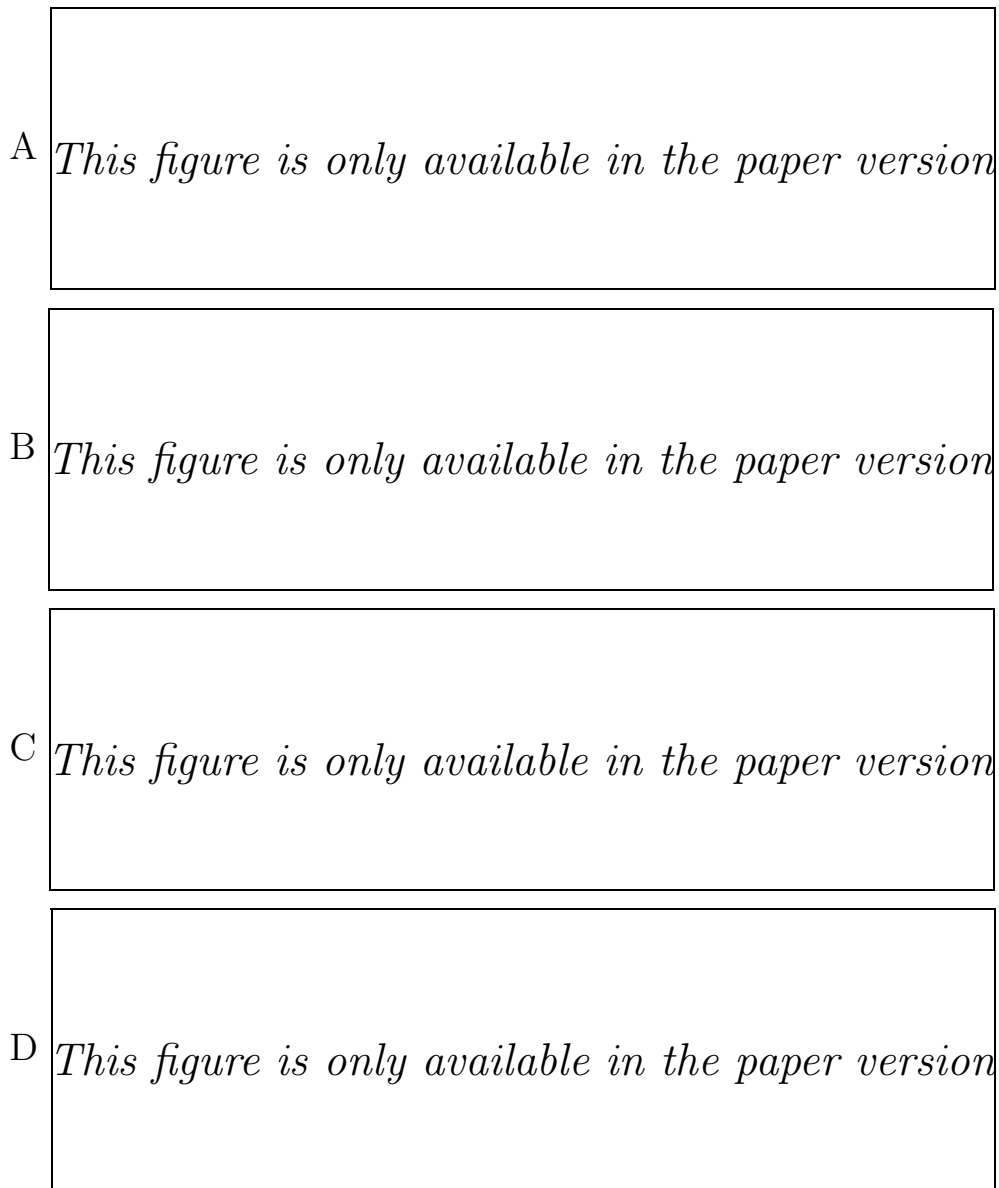


Figure 10: The picture series on this page shows the temporal evolution of a steady flow field in a channel subjected to supersonic inflow from the left. Red corresponds to high density, and blue to low density. The shock emanating from the foot of the ramp develops quickly and is very stable (A,B). The next noticeable feature is the expansion fan where the flow turns at the convex corner (B). After reflection at the upper wall, the shock interacts with the expansion fan (C). The wave character of the developing flow is obvious, as is the growth in strength of the reflected shock until it eventually hits the lower wall again (D). (See Section 3.5 on page 20.)

Density, after a long while...

This figure is only available in the paper version

Figure 11: Steady Mach 3 Flow in Converging Channel. The prediction of flow in air intakes for air breathing supersonic propulsion systems relies on simulations such as this one. This is a typical supersonic flow in a straight channel: Large regions of parallel flow separated by slanted shocks and expansion fans. The quality of the numerical solution improves with decreasing cell size. The CM allows the use of very fine grids in reasonable times, and this solution is well resolved. This particular computation ignores the boundary layers which form along the walls, and thus also the interaction between the layers and the shocks. These effects can usually be ignored in a converging channel such as this, but are important in diverging channels where separations may occur. (See Section 3.5 on page 20.)

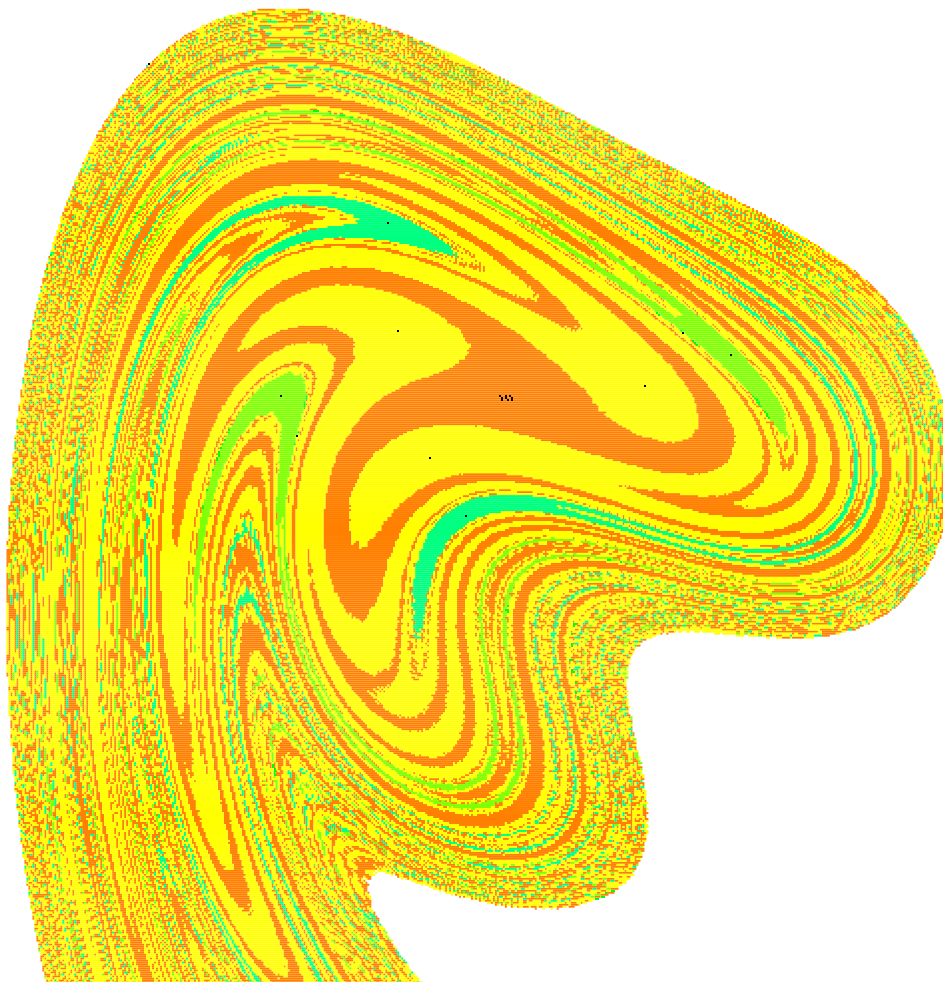


Figure 12: The effect of iterating a mapping; different colors indicate different asymptotic behavior. The complete turnaround time, including computation on the CM2 and post-processing of data for a 1000×1000 map is about one minute. (See Section 6.3 on page 37.)

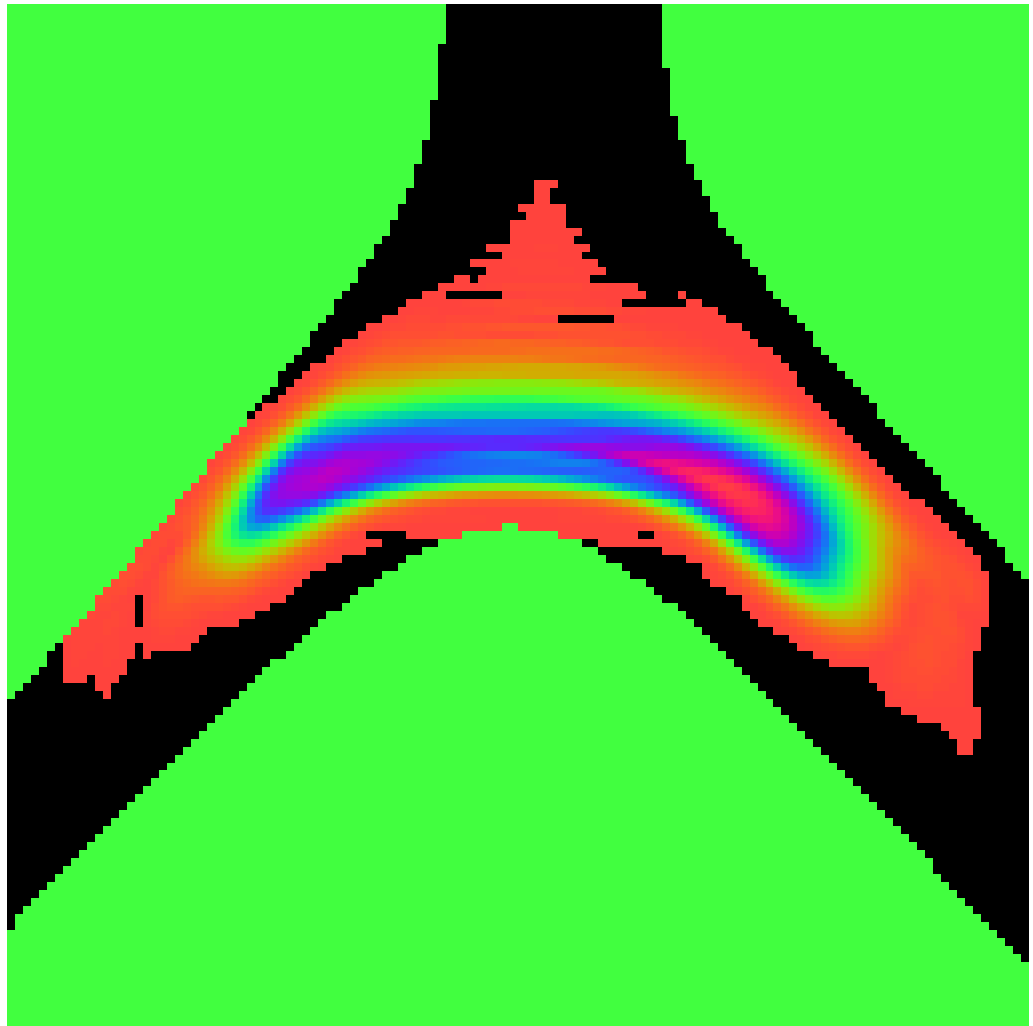


Figure 13: A simulation of an electron moving through a nanometer size Y-branch switch. The movement of the electron has been calculated using the Schrödinger equation. The color represents the probability of finding the electron at that point. With a potential difference of 0.4 V between the left and the right electrode, it is clearly seen how the electron, i.e. the current, primarily enters the right branch. (See Section 9.1 on page 43.)

6 Dynamical Systems

Supercomputing is having a dramatic effect in the analysis of dynamical systems. It is now possible to attack large and complex systems, and in many cases the interesting properties of a system only emerge with a sufficient number of degrees of freedom.

Often the goal of academic exercises is to discover patterns of behavior that can be understood in terms of simple models. One type of system contains many similar simple subsystems and is naturally well suited to data-parallel modeling. Other types of systems have few degrees of freedom, but parallelism can still be exploited by charting out parameter space with many parameter values running concurrently. This may require doing many simulations to assess effects of parameter variations. Post-processing of the raw data into graphical representations which capture the patterns is another key to scientific advance in the field.

The systems studied in the projects described here are of both types. Burgers' equation is a widely used simplified model for fluid flow, and the simulations performed here (Section 6.2) indicate new answers to questions on statistical properties of solutions after long times. The simulations of the forced spring-damper models (Section 6.3) reveal long-time behavior, and the computations of harmonic measures of fractal sets (Section 6.1) are of interest in growth processes as well as from a purely mathematical point of view.

6.1 Harmonic Measure on a Fractal

Erik Aurell

Department of Mathematics, Royal Institute of Technology

The harmonic measure on a set is the mathematical description of the following physical situation: the set is held at constant electric potential; an enclosing large container is grounded, then the harmonic measure on the surface (or in two dimensions, the boundary curve) is proportional to the local electric field normal to the surface. This describes many different problems of large current interest, such as electrodeposition, diffusion-limited aggregation and viscous fingering. Especially interesting, because they arise naturally in growth processes, are harmonic measures on fractal sets. These are also of intrinsic interest in complex iteration theory, in potential theory, and in other branches of mathematics.

I have used the Connection Machine to calculate the harmonic measure on simple fractals in the plane, by simulating in parallel the motion of a large number of random walkers absorbed on the fractal. Fractals are usually characterized

as having noninteger, or fractal, dimensions. The purely geometric layout of the fractal specifies one quantity, called the capacity dimension, which does not depend on the measure, and which may be adjusted continuously between zero and two for a set confined to the plane.

Other dimensions can be thought of as scaling exponents of the moments of the measure, and probe, qualitatively speaking, how much lumped together, or far from uniform, the measure is. The numerical results on these dimensions (e.g. Hausdorff dimension of the measure), are converged to better than one percent.

This is not quite as good as one would like, since a theorem of Lennart Carleson states that the Hausdorff dimension of the harmonic measure on a disconnected fractal set is strictly less than one, and in the parameter region where it could be larger, because the capacity dimension is tuned to be larger, I cannot distinguish the results from being precisely one. It would be interesting if one could estimate numerically how much the dimension is less than one, since this is difficult to do analytically, and it could help in understanding how dense parts of a fractal are screened away from the Brownian walkers.

In other parameter regions (where the capacity dimension is already less than one) I observe an almost complete absence of multifractal behavior; the measure is nearly uniform on the fractal. The behavior is therefore very close to a screening-transmission transition, and this appears not to have been known.

The speed of the Connection Machine allows one to get good estimates of, and even semi-rigorous bounds on, the measures by overcoming Monte Carlo noise by brute force. A problem is instead the lack of double-precision arithmetic on the present floating-point units, which puts a limitation on the resolution of the computation. With only a finite number of levels in the fractal, boundary effects become important in the construction of the measure, and these are difficult to estimate. With the new 64 bit floating-point units installed, I will be able to repeat the calculation, and obtain results to greater accuracy.

“The speed of the Connection Machine allows one to get good estimates ... by brute force.”

6.2 Burgers' Equation with Scaling Initial Data

Erik Aurell, Department of Mathematics, Royal Institute of Technology

Zhen-Su She, Princeton University

Uriel Frisch, Observatoire de Nice

Burgers' equation balances a non-linear convective term against a dissipative term and was initially proposed as a one-dimensional model of turbulence. It is a reasonable and simple model for shock formation in a compressible medium, especially if the shocks are small. There are however very direct applications to the diffusion of a phase of a complex amplitude with modulus fixed, or almost

fixed. This is a common situation in nonlinear physics; it describes such diverse processes as slowly burning flame fronts, propagation of neural pulses and, in general, wave propagation in one direction where the phase velocity depends linearly on the amplitude.

The authors have studied the solutions of Burgers' equation at vanishing viscosity with scaling, non-smooth, initial conditions. [She *et al.*, 1992] This describes the evolution of smooth, but random, initial data if the scales of time and space one considers are sufficiently long. The reason for choosing random, non-smooth and scaling initial data is computational efficiency: with smooth initial data everything is practically known except the behavior at large scales and after long times, which is what we were interested in.

We found, on a basis of large-scale numerical simulations and heuristic arguments, that with initial energy spectrum $E(k) \sim k^{-1-2h}$, $0 < h < 1$, i.e. random-walk initial data, all statistical properties are indistinguishable from those of an extremely simple fractal model with one free parameter (h). The result was quite surprising, and predicts for instance a particular scaling form of the number of small shocks, smaller than a given threshold value. If $-1 < h < 0$, i.e. white noise, the results are very different, and they imply that both shock locations and Lagrangian regular points [She *et al.*, 1992], are isolated and finite in number per unit length. Although physically the most natural one, we have not studied this case in detail yet, and at the moment one should just note that the results are qualitatively different from those obtained with Brownian motion initial data.

6.3 Interactive Analysis of Non-linear Mappings

Arne Nordmark

Department of Mechanics, Royal Institute of Technology

The study of dynamical systems of low dimension which exhibit chaotic behavior has received wide attention after the pioneering works by Lorenz and others. The current project is a study of second-order mechanical spring-damper systems with non-linear limiters and periodic forcing. By analytical pre-processing, the behavior of the systems can be related to a non-linear mapping of a plane, representing two of the parameters, onto itself.

Detailed study requires massive computer resources, and the CM2 is an ideal tool for charting out parameter spaces because of the easily exploited parallelism. The graphical rendering of the result is conveniently taken care of by a local workstation. This division of labour is a prime example of using the CM2 as a very powerful workhorse and leaving the less demanding tasks to a local workstation. The combination of the two resources gives an environment which is interactive

“Detailed study requires massive computer resources, and the CM2 is an ideal tool for charting out parameter spaces because of the easily exploited parallelism.”

for a class of problems that would otherwise entail long batch runs.

Figure 12 on page 33 shows the effect of iterating a mapping; different gray shades indicate different asymptotic behavior. The complete time for computation on the CM2 and post-processing of data for a 1000×1000 map is about one minute, and the whole process is responsive enough to be called interactive. The results of the study will be presented at *Svenska Mekanikdagarna 1992*.

7 Studies of Computer Architecture

Parallel computers themselves are eminent examples of systems built from simple components, which due to the sheer number of components can exhibit complex behavior. The design of such architectures must encompass dimensioning and control of the communication network. Different combinations of these and other parameters can be studied via simulation.

Parallel simulation is aimed at reducing the execution time of large and complex simulation models. Massively parallel SIMD computers can be used to speed up execution of time-consuming simulation programs. A parallel simulator built for the CM2 is reported in Section 7.1. The idea that some critical messages are of vital importance to overall system performance is discussed in Section 7.2, where also a type of flow control which will recognize prioritized messages is proposed. In Section 7.3, finally, the object of the performance study is the Connection Machine itself.

7.1 Parallel Simulation on a SIMD Computer

Rassul Ayani, Boris Berkman
TDS, Royal Institute of Technology

A parallel simulator based on a synchronous simulation scheme has been developed on the Connection Machine [Berkman and Ayani, 1991]. Multistage Inter-connection Networks (MIN) of up to 64K inputs have been simulated. Several experiments with both symmetric and asymmetric workloads were performed [Ayani and Berkman, 1991b]. We compared performance of the parallel simulator with that of a sequential simulator developed on a SPARCstation [Ayani and Berkman, 1991a]. Various parameters, such as message population, timestamp distribution, network size, and routing policy, were studied with respect to the performance. Several metrics were used to measure performance.

The experiments allowed characterization of some of the features of the parallel simulation scheme and some specific features of the Connection Machine. The experimental results demonstrate speed-up factors of over 2000 for some sets of parameters. At the same time the experiments revealed several shortcomings of the simulator developed on the CM2. First, it has a low efficiency (25% in the best case). This is probably a characteristic of SIMD computers, since very often only a portion of the processors are active. The bottleneck for parallel simulation on the CM2 was the processing of message queues and the cost of communication. The size of the main memory, 256 Kbits per physical processor, was an important factor limiting the possibility of simulating large MINs. The speedup reported

in [Ayani and Berkman, 1991a, Berkman and Ayani, 1991] could be seen as the best case due to the regular structure of the benchmark and the used parameters. A network with a feedback loop, such as a torus, would probably produce a lower speedup.

7.2 Performance Evaluation of a Flow Control Algorithm

Handong Wu, Lars-Erik Thorelli, Abdel-Halim Smai
TDS, Royal Institute of Technology

The interconnection network (ICN) can be a bottleneck in a parallel computer. The main design criteria of an ICN are the throughput (the number of messages delivered per time unit) and the latency (the time it takes for a message to be sent from source to destination). The choice of allocation strategies, the flow control algorithm, the bandwidth of the physical links, and the buffer resources at the nodes are crucial for performance.

In some systems, the latency of critical messages has a significant effect on the whole system performance. We propose a scheme that can take into account different classes of messages and also give critical messages a higher priority than others. The result can be expressed as an analysis of network latency and throughput of different classes of messages.

We have studied this flow control algorithm by simulation, using a time driven simulator constructed on the CM2. The proposed scheme has been evaluated on a two-dimensional mesh ICN with 128×128 nodes, where messages are continuously generated from each node. Whenever a message is injected into the network or a message has arrived at a destination, we collect statistics about latency and throughput. The main advantage of using the CM2 and programming in C* is the flexibility in constructing the simulator [Wu *et al.*, 1992].

7.3 Performance Analysis of the CM2

Jukka Helin
Tampere University of Technology

The performance evaluation process for a massively parallel distributed memory SIMD computer has been described generally. The performance in basic computation, basic communication, and computation with communication has been analyzed. A practical performance evaluation and analysis study has been done for the CM2 and conclusions about its performance have been drawn [Helin, 1991].

8 Data Parallel Visualization And Picture Processing

Early on, image analysis was recognized as an important application to which parallel processing could be applied. Real-time applications in robotics etc. require massive computing power, and the Data Parallel Programming model is natural for digitized pictures. A successful application of neural network models to a classification problem was described in Section 2.2; in Section 8.2 a representative set of operations used in image analysis is reviewed for implementation on a data parallel machine.

The creation of graphics to portray results of simulation and computing is crucial, and there is a need for easily adaptable programming environments which encourage experiments with new ideas for data visualization. Section 8.1 describes an attempt to use LISP as a high-level protocol in a client-server model for computing on the CM2 and visualizing on workstations.

8.1 Data Visualization—Volume Slicing

Magnus Persson

Digital Equipment Corporation

Peter Siklosi, Per Hammarlund, Lars-Johan Liman

NADA, Royal Institute of Technology

Our work, which is partially supported by DEC, is an outgrowth of a degree project in which the LISP dialect SCHEME was used to provide a very user-friendly interface, SCIX, to X-Windows. The project has developed a pilot implementation, SPEX, of the 3D graphics standard PEX on top of SCIX, and a high-level protocol for the communication of data between a graphics workstation running SPEX and a compute server, say the CM2. The target application is to slice multi-dimensional data sets residing on the CM2 into surfaces which are transmitted to the workstation for rendering and further post-processing. At present, a more hard-wired test code implements a parallel variant of the *marching cubes* algorithm ordinarily used in volumetric data exploration and displays the results on the workstation.

“The creation of graphics to portray results of simulation and computing is crucial ...”

“... data on the CM is transmitted to the workstation for graphical rendering ...”

8.2 Data Parallel Algorithms for Picture Processing

Thomas Johansson

Center for Image Analysis, Uppsala University

“... how different architectures affect the way algorithms should be implemented ...”

Image analysis algorithms are often implemented on special computer architectures optimized for the needs in image analysis. For this reason it is interesting to investigate how efficiently more general-purpose parallel computers can be used for image analysis and how their different architectures affect the way algorithms should be implemented. A project at the Center for Image Analysis, University of Uppsala, compares implementations of some of the most common and time-consuming image analysis algorithms on different parallel architectures. Characteristic algorithms from most of the branches of image analysis were studied with regard to parallel implementation. The implementations were done in C* because of its facility to support one-bit data and records.

The obvious way of mapping images to the CM2 is to map one processor to each pixel. This is practicable because of the CM2 Virtual Processor facility. Parallel programs on CM2 run independently of the number of physical processors. The input image is not always the only data structure we use in image analysis algorithms. For example, in object operations or feature extraction from the objects we only want to map one processor to each object pixel and skip the pixels which are not included in any object.

One of the most important operations is for each pixel to gather information from its neighborhood pixels. This operation is quite time-consuming compared to other computations. A variant which does two one-dimensional gatherings and builds records of the results of the first for subsequent use in the second was found to be superior.

“Most of the effort on the implementations was spent on operations which are not easy to formulate as parallel algorithms.”

Filter, fractile and edge operations show similar characteristics and are well suited for the CM2 as long as the vp-ratio is low (vp-ratio = number of virtual processors/number of physical processors). As an example of performance, an 8K CM2 takes 304 ms for a 2D FFT (forward) on a 512×512 image. Set operations and morphological binary operations are efficient on the CM2 because the images are stored as bitplanes. Comparison of two bitplanes takes less than 1 ms even if the vp-ratio is high.

Most of the effort on the implementations was spent on important binary operations which are not easy to formulate as parallel algorithms. As an example, we mention *labeling* which in this context means finding the connected regions of the image and numbering them starting from one. Serial implementations of these algorithms have recursive character and are not easily parallelized. A more detailed description of the implementations is given in [Johansson, 1991].

9 Applications in Physics

The classical field problems of physics are described by partial differential equations. The fluid-flow models discussed in Section 3 are outstanding examples, but wave-propagation models are also prominent models in classical physics. Such models describe vibrations, radiations, and the wave aspects of sub-atomic particles. It stands to reason that a diversity of such models have been successfully implemented on the CM2.

In fact, the CM2 is used as a numerical Schrödinger laboratory for the analysis of nanometer electronic devices and, when solving the continuity equations of current and electric charge, also as a device simulator for μm -scale semiconductor devices.

In addition, many-body problems are important models for both atomic lattices and molecules on the microscopic scale and for vortex flows and galaxies on the macroscopic scale. The n -body problem with large n requires sophisticated algorithms which use simplifications for the long-range interactions in order to avoid the $O(n^2)$ complexity. An example using short-range interactions is furnished by the lattice dynamics simulation of copper atoms described in Section 9.5.

9.1 Quantum Electronics: an Y-branch Switch for Electrons

Thomas Palm

Microwave Engineering, Royal Institute of Technology

With the current trend in miniaturization of electronic components it is inevitable that we will soon reach the point where the wave nature of the electron will have to be taken directly into account. While this causes problems for some components like the conventional transistor, it also creates an opportunity for making new devices.

The wave nature of electrons is quite similar to that of photons. It is therefore natural to use optics as a source of inspiration for possible components. At the Department of Microwave Engineering we are developing an electronic Y-branch switch, corresponding to the optical Y-branch which has proven very useful for switching networks in optical communication.

Figure 14 shows the Y-branch implemented using modulation doping. The current is trapped in a thin horizontal sheet at the boundary between the GaAs and AlGaAs layers and further restricted to a Y-shape by a negative potential applied to the electrodes at the surface. By changing the potential on the electrodes one can control which of the two legs of the Y the current will enter. This

Figure 14: The Y-branch implemented using modulation doping. The current is trapped in a thin horizontal sheet at the boundary between the GaAs and AlGaAs layers and further restricted to a Y-shape by a negative potential applied to the electrodes at the surface. By changing the potential on the electrodes one can control which of the two legs of the Y the current will enter. This is in contrast to the transistor where one instead controls the amount of current.

*This figure is only available
in the paper version*

is in contrast to the transistor where one instead controls the amount of current.

The behavior of this switch is found by directly solving the Schrödinger equation for an electron entering through the base. The planar structure of the device makes a two-dimensional solution possible, and the whole calculation process for a 128×128 grid in the Y-branch, including computation of the self-consistent electric potential in the 3D structure, takes about one minute. Spectral methods are used for solving the PDEs, and extensive use is made of the FFT routines in the CM scientific subroutine library.

Figure 13 on page 34 shows an electron halfway through a biased Y-branch. The colors represent the probability for finding the electron at that point. With a potential difference of 0.4 V between the left and right electrode, it is clearly seen how the electron, i.e. the current, primarily enters the right branch.

Quantum-interference components require coherent electron transport, i.e. the electrons must not scatter while in the device. This requires using very pure semiconductor crystals and, currently, low temperature. Together with the required nanometer size this has prevented building these devices until very recently. We are currently cooperating with the University of Lund in building a prototype Y-branch switch.

*“... the whole
calculation process
for a 128×128 grid
in the Y-branch, ...,
takes about one
minute.”*

9.2 High Temperature Superconductivity Models by Pair Correlations

Stellan Östlund

Department of Physics, Chalmers University of Technology

The search for materials which are superconducting at high temperatures, i.e. have high critical temperatures T_c , is intense in laboratories all over the world. The theoretical models of superconductivity were initially proposed by Nobel laureates Bardeen, Cooper, and Schrieffer (BCS) and have formation of pairs of particles at their hearts. In this project, the particular pairs of electrons, d-wave pairs, are distinguished by having a particular angular momentum. The project has used the CM2 intermittently but very intensely. The machine calculations have been used to guide analytic developments of tractable approximations for particular problems, and the speed, ease of access and relative ease of programming have been deciding factors in the use of the machine. Note that, for this kind of use, only total turnaround time is of importance, and one is quite willing to trade programming efforts for machine time. As an example, we have been using the Connection Machine to do numerical simulations of models of high- T_c materials where we investigated a d-wave pairing mechanism. After choosing a trial wave function with adjustable parameters, we proceed to minimize the total energy by adjusting the parameters. This is a very difficult minimization problem that fortunately was perfectly suited to be solved on the Connection Machine. The minimization involved twelve variables on each site of a 64×64 grid.

With the numerical solution at hand, we found that the solution was of sufficiently high symmetry, with d-wave pairing, so that the resulting equations could be solved more or less analytically with simpler methods. We checked our analytical results against the numerical minimization. The d-wave pairing hypothesis was strongly supported by the original numerical solutions and gave us confidence that the answer was correct.

This project shows an important use of the machine. It is often necessary to gain familiarity with a problem by intermittently “sketching” out numerical solutions. The analysis proper can then be attacked by alternative means once an intense numerical session has put us on the right track. Sometimes we use the machine intensively for a while and then start working on the problem using other methods. It is a use of the machine which is just as important as heavy numerical simulations but, averaged over a longer period of time, does not tax the machine resources heavily. However, without a numerical facility of the type at PDC we would not have gained enough insight into the correct answer to proceed to the solution.

“It is often necessary to gain familiarity with a problem by intermittently “sketching” out numerical solutions.”

“... without a numerical facility of the type at PDC we would not have gained enough insight ...”

This example stresses the importance of having the machine readily available to a wide variety of researchers who on the average do not do typical “number crunching” or simulations, but who occasionally need a very fast machine to solve specific problems in areas that are not necessarily numerically intensive.

9.3 Semiconductor Device Simulation—MINIMOS

Otto Heinrichsberger

Technische Universität, Wien

Computation of currents and electric fields in semiconductor devices require solution of 3D strongly non-linear conservation equations. The MINIMOS package, developed at the Institute for Microelectronics at the Technical University of Vienna with DEC support, is used in many academic institutes and also commercially. In a joint project with DEC and PDC, Dr. Heinrichsberger has implemented the most intensive numerical part of MINIMOS on the CM at KTH in CM Fortran. Most of the computation time is spent in solving large symmetric and unsymmetric sparse systems of linear equations by the pre-conditioned conjugate gradient method. The CM implementation of the symmetric solvers runs very efficiently, outperforming a CRAY 2 implementation. The unsymmetric solvers must use a parallelizable preconditioner, hence incomplete LU-factorization which is otherwise the method of choice is ruled out. The parallelizable alternative obtained by red-black ordering gives slower convergence, resulting in overall poorer performance than the CRAY 2. The main conclusion is that parallel implementations of device simulators is a viable option, but more research must be directed to finding efficient parallelizable preconditioners for the unsymmetric systems. [Heinrichsberger, 1991]

*“the CM
implementation of
the symmetric
solvers ran very
efficiently,
outperforming a
CRAY 2 ...”*

9.4 Ground Vibrations Induced by Dynamic Surface Loads

Marcus Berglund

C2M2, Royal Institute of Technology

Computations of the dynamic response of elastic structures to various loadings are usually and conveniently carried out by general-purpose finite element packages. For 3D solid structures the simulations are costly, and the computational models have to be simplified. However, if the structure is of sufficiently simple geometry and homogeneous in the sense that it can be modeled by one kind of finite elements, the CM2 can handle 3D problems with reasonable resolution.

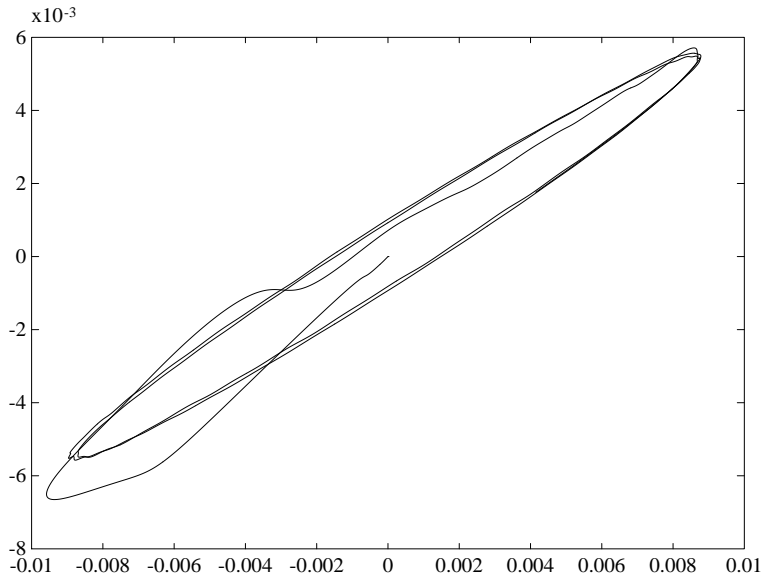


Figure 15: Dynamic response in a 3D elastic structure. Horizontal displacement is plotted as a function of vertical displacement. The displacements converge quickly toward a steady, periodic solution when a periodic driving force is applied. The transients decay very rapidly because most of the boundaries are non-reflecting.

The current computation of 3D elastic waves is a joint project with the Department of Soil and Rock Mechanics at the Royal Institute of Technology to study the vibrations induced in the ground by dynamic loads on the surface. Earlier studies have been limited to 2D models by the computing resources. This is a very crude approximation to reality since the decay of the wave amplitude with distance is $1/r$ in 2D, rather than the correct $1/r^2$. The finite difference technique was chosen in this case because the constant coefficients of the problem allow the coefficients of the computational stencils to be precomputed and stored, reducing substantially the amount of arithmetic necessary. The extensive code for computing the coefficients was produced with the help of the symbolic algebra package *Mathematica*. The program is being validated against known analytical solutions and 2D finite element calculations. The *Lissajous graph* in Figure 15 shows how the displacements tend toward the steady, periodic solution when a periodic driving force is applied. The transients decay very rapidly because most of the boundaries are non-reflecting.

9.5 Molecular Dynamics

Ole H. Nielsen

Laboratory of Applied Physics, Technical University of Denmark

*“... nature herself is
massively parallel.”*

The n -body method is a powerful tool for studying large, complex systems of bodies, especially when coupled with state-of-the-art computers. In the n -body method, called Molecular Dynamics, the time evolution of systems is studied with given forms of the interactions between the constituent parts, for example atoms or molecules.

Massively parallel computers such as the CM200 are ideally suited for solving this problem, which involves thousands or even millions of identical atoms that behave according to the laws of physics; in this respect, *nature* herself is massively parallel. Atomic or molecular interactions may be classified according to whether they are long-range (electrostatic) or short-range (quantum-mechanical) in nature, and the computational methods are chosen accordingly. In the present project, interactions in the physical model (Effective Medium Theory) extend to about the fourth atomic neighbors.

A CM Fortran code for short-range molecular dynamics was developed on the CM2 at KTH, dividing the computational box geometrically into sub-boxes, each of which is assigned to a processor on the CM200. By using regular communication in fixed patterns, the interatomic interactions are calculated very efficiently. The stochastic rearrangements of atoms inherent in a finite-temperature system are treated by regular communication as well. The production code is compute-bound, with less than 10% of the time being spent in data communication. In case of ideal load balancing, the code has an effective speed of about 1000 MFLOPS on an 8K CM200, but in real cases only about half of that speed is attained, due to some processors being inactive some of the time.

The current project studies the crystal-to-liquid phase transition in mesoscopic-sized copper clusters, including effects such as facetting, premelting, solid-liquid coexistence, and critical nuclei. In a preliminary study, the cluster appearing on the cover of this report was simulated with increasing values of the temperature, leading finally to the complete melting into a liquid droplet.

10 Glossary

CFD, Computational Fluid Dynamics Computational Fluid Dynamics is the study of flow phenomena by computational methods. In fluid dynamics, like in many other branches of natural science, the traditional physical methods of theoretical analysis and experimental observations can now be complemented by computational experiments. Present advances in computer power and algorithm design allow serious efforts both in basic science, such as the understanding of the properties of the Navier–Stokes equations, and in practical applications to aerospace engineering.

CFL, Courant, Friedrichs, and Lewy Condition In a classic paper from 1928, Courant, Friedrichs, and Lewy give for the first time an important condition for stability of a numerical time-stepping scheme. Let the velocity be U , the cell size δx , and the time step δt . The CFL-number is then

$$CFL = \frac{U\delta t}{\delta x}$$

and the condition

$$CFL \leq 1$$

states that the velocity must be so small that no wave travels more than one cell per time step.

FFT, Fast Fourier Transform Any signal can be seen as the superposition of harmonics with different frequencies. The set of amplitudes of the different frequencies is the *Fourier Transform* of the signal. Multi-variate functions can be decomposed by treating one variable after the other, and we then speak of multi-dimensional Fourier Transforms. When the decomposition is performed on a sampled signal we call it a *Discrete Fourier Transform* (DFT). The actual computation of the DFT can be seen as a complex matrix-vector multiplication where the matrix elements are roots of unity and the vector is the set of N samples. The many symmetry properties of the matrix allows one to perform the calculation in only $O(N \log N)$ arithmetic operations as compared to $O(N^2)$ required for a general matrix multiply, and these algorithms are therefore called *Fast Fourier Transforms* (FFT). FFTs are used extensively in scientific computations. They perform filtering operations in image and signal processing applications, such as speech recognition and synthesis and computed tomography, and they make spectral methods for solving differential equations computationally tractable.

FLOPS, Floating Point Operations per Second A measure of numerical performance of a computer.

SHPCnet, The Swedish High-Performance Computing Network The initial intention is to connect the three Swedish supercomputer centers Linköping (CRAY), Stockholm (CM200) and Skellefteå (IBM 3090) with 34 Mbit/s links. This is a first

step towards *realistic distributed supercomputing*. Apart from a number of projects that aim directly at the distributed possibilities, it is also reasonable to foresee a more efficient sharing of existing computer resources among high-performance users in Sweden.

HUGO, Human Genome Project The Human Genome Project aims at creating a database of the human genome, the DNA, that will for instance aid research in medicine. Collecting this information is a formidable task that includes work and development in many different areas—from biotechnology to computer science. As a first step the laboratory has to find the sequences; much of this work has been automated. The sequences are stored in a database. When the information on the sequences has been collected, researchers can start using it. New techniques have to be created for maintaining and using this database.

MIMD, Multiple Instruction Multiple Data Computer The term “Multiple Instruction Multiple Data” describes a specific parallel computer architecture. In a MIMD computer the processing elements (PEs) have their own code and can all perform different instructions on their local data. The MIMD programming model is more general than the SIMD model. Recently there has been a revival of the MIMD computer as the microprocessors have become very powerful—this can be seen in the CM5 from Thinking Machines and the Paragon from Intel.

SIMD, Single Instruction Multiple Data Computer The term “Single Instruction Multiple Data” describes a specific parallel computer architecture. In a SIMD computer instructions are broadcast to all processing elements (PEs) from the control processor. The PEs all perform the *same* instruction on their own local data. One reason for building computers like this is that the architecture reduces the complexity of the PEs—they do not have to have local code and hardware for parsing this code. The fact that all PEs perform the same instruction is a restriction in the programming model; a SIMD computer can however perform any operation a MIMD computer can, it only increases computing time by a constant factor. A SIMD computer will usually have more PEs than a MIMD computer.

11 References

- ANDERSSON, B. (1991). Det var BELLMAN, HUGO och några forskare *Ny Teknik* **46**: 20–21. 25
- ARGOS, P., VINGRON, M., AND VOGT, G. (1991). Protein sequence comparison: methods and significance. *Protein Engineering* **4**: 375–383. 24, 24
- AYANI, R. AND BERKMAN, B. (1991a). Parallel Discrete Event simulation on SIMD Architecture. Submitted to the Journal of Parallel and Distributed Computing. 39, 40
- AYANI, R. AND BERKMAN, B. (1991b). Performance of a Synchronous Simulation Scheme on a Massively Parallel Computer. In: *Proc. of the European Simulation Multiconference*, Copenhagen, Denmark. 39
- BERGMAN, M., WAHLUND, P., AND VOS, J. (1991). Implementation of a 2D Multi Block Euler Solver on the Connection Machine. In: *Third Annual Conference on Parallel CFD*, Stuttgart. To be published by Vieweg Verlag. 21
- BERKMAN, B. AND AYANI, R. (1991). Parallel simulation of multistage interconnection networks on a SIMD computer. In: *Proc. of the Parallel and Distributed Simulation Conference*, Anaheim, CA. 39, 40
- COULSON, A. F. W., COLLINS, J. F., AND LYALL, A. (1987). Protein and nucleic acid sequence database searching: a suitable case for parallel processing. *The Computer Journal* **30**: 420–424. 24
- EKEBERG, Ö., STENSMO, M., AND LANSNER, A. (1990). SWIM. A Simulator for Real Neural Networks. Technical Report TRITA-NA-P9014, Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, Stockholm, Sweden. 14
- EKEBERG, Ö., STENSMO, M., TRÅVÉN, H., LEVIN, B., HAMMARLUND, P., AND LANSNER, A. (1991a). Tools for Biologically Realistic Simulation of Neural Networks. In: *Artificial Neural Networks, Proceedings of ICANN-91*, edited by K. M. T. Kohonen, O. Simula, and J. Kangas, Espoo, Finland. North-Holland, Amsterdam, p. 1439–1442. 14
- EKEBERG, Ö., WALLÉN, P., LANSNER, A., TRÅVÉN, H., BRODIN, L., AND GRILLNER, S. (1991b). A computer based model for realistic simulations of neural networks. I: The single neuron and synaptic interaction. *Biological Cybernetics* **65**: 81–90. 14

- HALLBÄCK, M., LARSSON, T., MALINOWSKY, L., AND JOHANSSON, A. (1991). Numerical Simulation of Return to Isotropy in Homogenous Turbulence—Reynolds Number Dependence. In: *1st European Fluid Mechanics Conference*, Cambridge. 17
- HAMMARLUND, P., LEVIN, B., AND LANSNER, A. (1991). Biosim—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. In: *Artificial Neural Networks, Proceedings of ICANN-91*, edited by K. M. T. Kohonen, O. Simula, and J. Kangas, Espoo, Finland. p. 1477–1480. 13
- HEINREICHBERGER, O. (1991). Implementation of MINIMOS on Parallel Machines. In: *2nd Copper Mountain Parallel Computing Conference*, February 1991. 46
- HELIN, J. (1991). Performance Analysis of the CM2, a Massively Parallel SIMD Computer. Submitted for acceptance to *Concurrency, Practice and Experience*. 40
- JOHANSSON, T. (1991). Data Parallel Algorithms for Picture Processing. Technical report, Center for Image Analysis, Uppsala University. To appear. 42
- LANSNER, A. (1991). SANS, Studies of Artificial Neural Networks—Annual Progress Report. Technical Report TRITA-NA-P9126, Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, Stockholm, Sweden. 10
- LANSNER, A., EKEBERG, Ö., WADDEN, T., TRÅVÉN, H., FRANSÉN, E., GRILLNER, S., WALLÉN, P., AND BRODIN, L. (1991). Examples of Realistic Neural Network Simulations. In: *Artificial Neural Networks, Proceedings of ICANN-91*, edited by K. M. T. Kohonen, O. Simula, and J. Kangas, Espoo, Finland. North-Holland, Amsterdam, p. 1431–1434. 15
- LANSNER, A. AND FRANSÉN, E. (1991). Modeling Hebbian Cell Assemblies Comprised of Cortical Neurons. In: *Artificial Neural Networks, Proceedings of ICANN-91*, edited by K. M. T. Kohonen, O. Simula, and J. Kangas, Espoo, Finland. North-Holland, Amsterdam, p. 1747–1750. 15
- LEVIN, B. (1990). Implementation of the SANS algorithm on the CM2. Technical Report TRITA-NA-P9011, Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, Stockholm, Sweden. 11, 12

- LEVIN, B. (1991). Implementation of the SANS algorithm on the CM2. In: *Artificial Neural Networks, Proceedings of ICANN-91*, edited by K. M. T. Kohonen, O. Simula, and J. Kangas, Espoo, Finland. North-Holland, Amsterdam, p. 1473–1476. 11
- LEVIN, B., HAMMARLUND, P., AND LANSNER, A. (90). BIOSIM—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. Technical Report TRITA-NA-P9021, Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, Stockholm, Sweden. 13
- MALINOWSKY, L. (1991). Spectral Methods on the Connection Machine. Master's thesis, Royal Institute of Technology. TRITA-MECH-91xxxx, to appear. 18
- SAWLEY, M. AND BERGMAN, M. (1991). Computational Fluid Dynamics on Massively Parallel SIMD Computers. *EPFL Supercomputer Review*, Lausanne, Switzerland. 21
- SHE, Z., AURELL, E., AND FRISCH, U. (1992). The Inviscid Burgers' Equation with Initial Data of Brownian type. *Comm. Math. Phys.*, In press. 37, 37
- SINGER, A. (1990). Implementations of Artificial Neural Networks on the Connection Machine. Technical Report RL 90–2, Thinking Machines Corporation, Cambridge. 11
- SJÖGREN, P. (1991). Extraction of Significant Features in long bitstring, implementation on the Connection Machine. Master's thesis, School of Electrical Engineering, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 13
- SVENNBERG, U. (1991). An effective data parallel implementation of the CSD method. Master's thesis, Chalmers University of Technology, Department of Mathematics. In Swedish. 21
- VON HEIJNE, G. (1988). Getting sense out of sequence data. *Nature* **333**: 605–607. 24
- WAHLUND, P. (1990). Parallel Numerical Computations. Working paper. 21
- WALLIN, E. (1991). Optimized sequence matching on the CM2. Master's thesis, Royal Institute of Technology. 25
- WETTERGREN, C. (1991). Development of a fast parallel sequence alignment-package, the serial parts. Master's thesis, Royal Institute of Technology. 25

- WU, H., THORELLI, L.-E., AND SMAI, A.-H. (1992). Flow control support for efficient realisation of the Extended Data Flow Architecture model. *Microprocessing and Microprogramming* 175–178, Volume 34. 40

12 Index

- AlGaAs, 44
- Alliant, 5
- ANN, *see* Neural Networks
- Atomic Lattices, 43
- Aurell, Erik, 35, 36
- Automatic Mesh Generators, 23
- Ayani, Rassul, 39

- Bardeen, 45
- BCS, 45
- Berglund, Marcus, 46
- Bergman, Magnus, 20, 23
- Berkman, Boris, 39
- Biocomputing, 8, 24–25, 26
- BIOSIM, 13–15
- Burgers' Equation, 35

- C
 - ANSI C, 7
 - C* (C star), 7, 24, 40, 42
- Carleson, Lennart, 36
- Carlsson, Janne, President of KTH, 1
- Center for Biotechnology, 8, 24
- CFL, *see* Courant-Friedrichs-Lewy Condition, 49
- Chalmers University of Technology, 7, 21, 45, 53
- Chang, Jennifer, 20
- Characteristic Streamline Diffusion, 21
- Classification
 - Maximum likelihood method, 12
 - Satellite images, 11
- Client-Server Model, 8, 24, 41
- CM Scientific Subroutine Library, 18
- CM5, 50
- CMIS, 12, 18
- CMSSL, *see* CM Scientific Subroutine Library
- Common Lisp, 7
 - *Lisp (star lisp), 7
- Communication Networks
 - Message Passing, 39
- Computational Fluid Dynamics, 8, 16, 16–23, 49
- Computer Architecture, 39–40
- Connected Regions, 42
- Continuity Equations, 43
- Convex, 5
- Cooper, 45
- Cortical Oscillations, 15
- Courant-Friedrichs-Lewy condition, 22
- CRAY, 5, 8, 17, 18, 21, 46, 49
- Crystal-to-Liquid Phase Transition, 48
- CSD, *see* Characteristic Streamline Diffusion

- Data Parallel Programming
 - Model, 6–9, 41
- Data Vault, 3, 5
- Data Visualization, 8, 41
- Department of Soil and Rock Mechanics, 47
- DFT, 49
- Diffusion-Limited Aggregation, 35
- Digital Equipment Corporation, 41, 46
- Discrete Fourier Transform, 49
- Dynamical Systems, 35–38
 - Burgers' Equation, 35
 - Fluid Flow, 35
 - Fractals, 35
 - Harmonic Measure, 35

- Hausdorff Dimension, 36
- Random Walk, 37
- Spring-Damper Systems, 37
- Ecole Polytechnique Fédéral de Lausanne, 16
- Elastic Structures, 46
- Electrodeposition, 35
- Electronic Switch
 - Y-Branch, 43
- Electrostatical Interaction, 48
- Emergent Properties, 8, 35
- Eriksson, Kenneth, 21
- Euler Equations, 19, 21
- Extended Data Flow Architecture, 54
- Fast Fourier Transform, 17, 18, 42, 44, 49
- FEM, *see* Finite Element Methods
- FFT, 49
- Finite Difference Methods, 16
- Finite Element Methods, 16, 21
- Finite Volume Methods, 16
- Finite Difference Methods, 47
- Flame Fronts, 37
- Fluid Flow, 35, 43
 - Compressible Viscous Flow, 20
 - Unstructured Grids, 23
- FORTTRAN
 - CM Fortran, 4–7, 18, 23, 46, 48
 - Fortran90, 6, 20
- Fractals, 35
- Framebuffer, 4
- Frisch, Uriel, 36
- FRN, 2
- Front End Computers, 4
 - Sun Microsystems, Inc., 4
- GaAs, 43, 44
- Galaxies, 43
- Gallium Arsenide, *see* GaAs
- George Washington University, 16, 23
- Graphics, 37, 41–42
 - 3D, 41
 - Image Analysis, 8, 41
 - Post-processing, 35, 41
- Hallbäck, Magnus, 16
- Hammarlund, Per, 3, 12, 13, 41
- Hansbo, Peter, 21
- Harmonic Measure, 35
- Hausdorff Dimension, 36
- Hedin, Anders, 3
- Hedman, Fredrik, 3, 24
- Heijne, von, Gunnar, 24
- Heinrichsberger, Otto, 46
- Helin, Jukka, 40
- High Performance Computing and Communication Initiative, i, 5
- High-Level Protocols, 8, 41
- HUGO, *see* Human Genome Project
- Human Genome Project, 24, 50
- IBM
 - IBM 3090, 49
- Ihrén, Johan, 3
- Intel Paragon, 50
- Inter-Processor Communication, 19
- Johansson, Thomas, 42
- Johnson, Claes, 21
- Karolinska Institutet, 8, 10, 24
- Lagrangian Regular Points, 37
- Lansner, Anders, 3
- Levin, Björn, 11, 13
- Liljenström, Hans, 15
- Liman, Lars-Johan, 3, 41
- LISP, 8, 41
- *Lisp (star lisp), 7
- Lissajous Graph, 47
- Löhner, Rainald, 23

- Lorenz, 37
- Malinowsky, Lars, 18
- Marching Cubes, 41
- Martin, Dorothee, 23
- Mathematica, 47
- Message Passing, 39
- Microcode, *see* CMIS
- MIMD, 4, 9, 20, 50
 - Local Memory, 3
 - Shared Memory, 3, 4
- MIN, *see* Multistage Inter-connection Network
- Miniaturization, 43
- MINIMOS, 46
- Molecular Dynamics, 48
- Monte Carlo, 36
- Multi-Block Technology, 20
- Multistage Inter-connection Network, 39
- n*-body Simulation, 9, 43, 48
- Navier–Stokes Equations, 8, 16, 19, 21
- Needleman–Wunsch Sequence, 24
- Neural Networks, 7, 10–15
 - Biologically realistic simulations, 13
 - Cortical Associative Memory, 15
 - Decorrelation, 13
 - Description, 10
 - Hopfield, 11
 - Lamprey Spinal Cord, 15
 - Olfactory Cortex, 15
 - Propagation of Neural Pulses, 37
 - Relaxation, 10
 - Sparsification, 13
- Neurophysiology, 10
- Nielsen, Ole H., 48
- Nordmark, Arne, 37
- Nucleic Acid, 24
- NUTEK, 2
- Olsson, Pelle, 19
- Oppelstrup, Jesper, 3
- Optics, 43
- Östlund, Stellan, 45
- Palm, Thomas, 43
- Paragon, 50
- Partial Differential Equations, 7, 43
- Particles Differential Equations, 44
- Persson, Magnus, 41
- PEX, 41
- Physics, 43–48
- Protein, 24
- Protein Matching
 - Neural Networks, 13
- Protein Sequence Matching, 8
- Quantum Electronics, 4
- Quantum-interference components, 44
- Quantum-Mechanical Interaction, 48
- Radiation, 43
- Reynolds Numbers, 16
- Robotics, 41
- Rotta Constant, 17
- SANS, 7, 10–13, 15
- Scalable Computing, 6
- SCHEME, 41
- Schrieffer, 45
- Schrödinger, 43
- SCIX, 41
- Screening-Transmission Transition, 36
- Sequent Symmetry, 4
- She, Zhen-Su, 36
- Shocks
 - Formation, 36
- SHPCnet, *see* Swedish High-Performance Computing Network
- Siklosi, Peter, 41

- SIMD, 3, 6, 7, 9, 23, 39, 40, 50
- Sjögren, Peter, 13
- Skandinaviska Enskilda Banken, 1
- Smai, Abdel-Halim, 40
- Sparse Systems, 46
- SPEX, 41
- Spring-Damper Systems, 37
- Stencils
 - Precomputed, 47
- STU, 2
- Sun Microsystems, Inc., 4
- Sundblad, Yngve, 3
- SUNET, *see* Swedish University Network
- Superconduction, 45
- Svensson, Gert, 3
- Swedish High-Performance Computing Network, 9, 49
- Swedish Space Corporation, 11
- Swedish University Network, 4, 25
- SWIM, 14
- Switching Networks, 43

- Technical University of Denmark, 48
- Technical University of Vienna, 46
- Thorelli, Lars-Erik, 3, 40
- Transputer, 4
- Trollius, 4
- Turbulence, 16–18, 36

- University of Illinois, Urbana, 15
- University of Lund, 44
- University of Uppsala, 42
- UNIX, 4

- Vibration, 43
- Viscous Fingering, 35
- Vortex Flows, 43

- Wahlund, Per, 20
- Wallin, Erik, 24

- Wave Propagation, 43
- Wettergren, Christian, 24
- Wind Tunnel, 17
- Wu, Handong, 40

- X Windows Interface, 41