Progress Report
1992

PDC
Center for Parallel Computers

Progress Report 1992

VETENSKAP OCH KONST

KTH

The Cover Picture: A rendering of a geometric model of the CM200 Connection Machine installed at KTH. Images sampled from different projects at PDC have been texture mapped onto the surfaces not occupied by LEDs. All the images are results from production runs at the CM200 at PDC. The rendering model was developed with the visualization software AVS by Johan Ihrén, PDC. (Copyright © 1993 Center for Parallel Computers.)

PDC Parallelldatorcentrum
Royal Institute of Technology
S–100 44 Stockholm, SWEDEN
Telephone: +46–8–790 78 84, +46–8–790 78 91
Telefax: +46–8–24 77 84
Email: info@pdc.kth.se

Publisher: Parallelldatorcentrum
Editors: *Fredrik Hedman*, *Per Hammarlund*, *Jesper Oppelstrup*
Typesetting & layout: *Jan Michael Rynning*, *Fredrik Hedman*
Linguistic Revision: *Sven Westman*
Printed by: *Ekblads, Västervik, May 1993*
ISBN 91-7170-127-3

Foreword

The Center for Parallel Computers at the Royal Institute of Technology has now been active for more than three years as a national forum for research on and use of parallel computers.

The continued positive trend from our first progress report is apparent: the diversity of projects in this second progress report is evidence that the Connection Machine is a general-purpose supercomputer. The total number of users have more than doubled during the past year. In particular, the growth in the number of projects from Physics-related areas is remarkable. Also noteworthy is the tendency towards production. In fact, around fifty percent of our projects are in a production phase. The Connection Machine is truly a national resource: half of the projects are pursued by groups outside of KTH and a number of groups from other countries.

A national resource

To achieve our general goal we believe it is of prime importance to provide the Connection Machine as an open and easily accessible resource to academia and industry. Time on the Connection Machine is provided free of charge for academia, and we intend to keep this policy as long as our funding allows – at the time of writing our funding from July 1993 is still unclear.

An easily accessible resource to academia and industry

However, our role is not only to provide hardware resources but also to offer help and advice to users and actively try to catalyze interest and activity in parallel computing. We are convinced that close relations between computing scientists and scientists from different application areas are some of the important advantages offered by a small organization such as PDC, and a prerequisite for continued progress in scientific computing.

The essential aspect of a supercomputer is its *solution capability for large and complex problems.* For many real-world problems with large data sets, this translates into solving a problem in reasonable time as compared to not solving it at all.

The development of massively parallel computers is application driven. Today, such machines provide higher performance on a number of important applications compared to more traditional vector computers. A new generation of massively parallel machines has been introduced since we acquired the Connection Machine. By acquiring such a machine we will be able to begin solving larger and more complex problems. This generation of computers has a tremendous potential and it is projected that on some applications a *sustained* TFlop/s will be achievable in three to four years. Efficient use of these computers also requires substantial investment in education and training. Consequently, we plan to strengthen the user support group at the center. Such expansion is necessary to properly support our growing academic user base and also to attract industrial projects. The need for an expansion of the resources at our center has also been pointed out in a recent evaluation by a group of international experts, working for the Swedish National Board for Industrial and Technical Development.

We plan to strengthen the user support group

With some notable exceptions, Swedish industry has been somewhat reluctant to use large-scale computing on supercomputers, in particular parallel supercomputers. This will hopefully change, and we intend to make a serious educational effort to demonstrate successful and cost-effective computing projects.

Large Swedish investments in education, training and supercomputer hardware are required

One of the backbones of the supercomputing infrastructure in Sweden, are the national high-performance networking projects. However, we would like to stress that to make the infrastructure balanced and complete, large Swedish investments in education, training and supercomputer hardware are required.

Thus, we are confident that the near future will see a rapid growth of the supercomputing community. We will do our best to provide continued parallel computing resources and expertise in this exciting development.

The Board of the Center for Parallel Computers, March 1993

Contents

1 PDC – Parallelldatorcentrum

This is the second progress report of the Center for Parallel Computers (PDC) at the Royal Institute of Technology (KTH). It covers the activities of the center during 1992. A brief overview of the center follows in this section. User projects are described in Sections 3 through 10. The last three sections contain a glossary, a bibliography and an index of terms and names.

1.1 Background

Massively parallel computers are important because they offer *solution capability for large and complex problems.* For many real-world problems with large data sets, this translates into solving a problem in reasonable time as compared to not solving it at all. Parallel computing however requires new ways to approach a problem and develop algorithms. This means that practical experience of parallel computing in many different areas is important.

The Center for Parallel Computers at KTH was established in January 1990 to act as a focal point and national forum for both research on and use of parallel computers, including the new Connection Machine. In December 1991 the Connection Machine has been upgraded to a CM200, including larger memory and a parallel disk array.

PDC is an interdisciplinary organization. Our goal is to *stimulate research and spread information on the use of parallel computers.* This is achieved by providing high-performance parallel computers and expertise on their use to the technical and scientific computing community in Sweden. We aim to reach professionals from academia and industry who have large and compute-intensive applications.

PDC is a national resource and as such we provide our services free of charge to academic users in Sweden. Industrial partners are also welcome to explore the possibilities of parallel computing.

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 Royal Institute of Technology, the Swedish National Board for Industrial and Technical Development, NUTEK and the Swedish Research Council for Engineering Sciences, TFR. Our past and present funding is detailed in the table below.

Grants kSEK	89/90	90/91	91/92	92/93
NUTEK	1500	1500	1785	473
KTH	100	300	380	880
TFR	0	0	0	300
Total	1600	1800	2165	1653

The relatively low cost during the fiscal year 1992–1993 is due to a one year warranty included in the upgrade of the CM.

1.3 Organization

The Center for Parallel Computers is headed by a board, with Professor Thorelli as the chairman and Dr. Oppelstrup as the vice chairman. The complete list of members is:

Björn Engquist	Professor of Numerical Analysis, NADA
Fredrik Hedman	Application Engineer, PDC
Anders Lansner	Director of Research SANS, NADA
Jesper Oppelstrup	Director of Research C2M2, NADA
Yngve Sundblad	Chairman of the KTH Computer Council
Gert Svensson	Coordinator, PDC
Lars-Erik Thorelli	Professor of Computer Systems, IT

From an administrative point of view, PDC is currently a project in the Department of Teleinformatics (IT, previously TDS). Teleinformatics has just relocated to new premises at Kista, as a consequence of the new organization at KTH. For this reason PDC will be transferred to the Department of Numerical Analysis and Computing Science (NADA) in April 1993. Professor Björn Engquist will be chairman of the board, starting May 1993.

We are currently establishing a scientific advisory committee, which will give us a broader insight in different application areas and provide more information about the need of different user groups and universities.

The center has a staff of four persons, or about three full-time equivalents: Fredrik Hedman, application engineer; Johan Ihrén, UNIX system manager and graphics specialist; Britta Svensson, administrative assistant; and Gert Svensson, project coordinator.

1.4 Educational Activities

Throughout the year several seminars on the use of parallel computers have been arranged. The Connection Machine has been used for courses in the M.Sc. program at KTH and CTH. A two-day introduction to the CM-system was given in October at Umeå University with about 15 participants. Two complete courses (3 KTH-credits each) on programming of massively parallel computers have also been arranged:

Course	Date	Students
Programming of MPP Computers	April 92	28
Advanced Programming of MPP Computers	May 92	14

1.5 Project Summary

Currently there are some 35 ongoing projects. The statistics collected in the table below stem from a rough classification of the contributions as new or continued, and whether in a production or development phase.

Area	New	Cont.	Production	Development
Neural Networks	2	2	3	1
CFD	3	3	3	3
Biocomputing	1	1	2	0
Physics/Geophysics	9	3	7	5
Chemistry	3	0	0	3
Numerical Analysis	3	2	2	3
Computer Science	2	1	1	2

The growth in the number of applications from Physics-related areas is quite remarkable. It is a strong testimony to the useful-

The machine is a national resource in reality and not only in theory

ness of parallel computers in basic science. The trend towards production is seen clearly: about 50 percent of the projects are in a production phase. The machine is a national resource in reality and not only in theory: more than half of the projects are pursued by non-KTH groups situated in Göteborg, Linköping and Uppsala; international groups come from Denmark, France, Russia, Switzerland and the USA.

In many of the projects under way, PDC has given substantial support by actually participating in the development of programs and in the development of algorithms.

1.6 Hardware Resources

PDC provides access to three classes of computer architecture: SIMD, MIMD with shared memory and MIMD with local memory. Currently most of the activities are centered around the SIMD computer, the Connection Machine.

The Connection Machine

The CM200 Connection Machine at PDC is a SIMD supercomputer with 8192 bit-serial processors, 256 64-bit floating-point processors and 1024 MByte of main memory. It can be used either as an 8K machine or as two separate 4K machines. Connected to the CM is a 10 GByte DataVault, i.e. a parallel disk array, which substantially improves the I/O performance of the system as a whole.

Included in the Connection Machine system is a framebuffer, which is a high-speed graphical device connected directly to the CM. This enables quick display of large amounts of data stored in the machine, and makes it easy to visualize a simulation as it is performed.

The two Sun front-end computers run the UNIX operating system; they are easily accessible via the Swedish University Network (SUNET). The front-end computers act as control processors in the CM system during user program's startup and execution, and they also host user's program development and compilation.

Other Hardware Resources

PDC has direct access to two other parallel computers, a Sequent Symmetry and a transputer based system. These are not meant for high-performance computing, but rather serve as typical examples of their respective architectures.

1.7 Software Environment

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. 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. The rapid evolution of compilers, scientific libraries, debuggers and system software on the CM200 is also important.

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, a subset of Fortran 90; C*, an extension of ANSI C; and *LISP, an extension of Common Lisp.

Designing scalable programs is simplified in the data-parallel programming model. It should be noted that the model can be implemented on both SIMD- and MIMD-machines, making it very attractive as a base for writing programs and designing algorithms.

1.8 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.

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

Many areas of Computational Science is currently undergoing a transition from 2D to 3D models; partly because of a specific need to produce more realistic models and partly because of the growth in capacity of supercomputers. To gain insight and understanding into complex problems characterized by very large data sets, a combination of advanced visualization systems and high-end graphics computers is needed. In collaboration with several groups at NADA, PDC is currently conducting an investigation of what systems to acquire.

Furthermore, PDC is planning to acquire a scalable massively parallel MIMD-computer with programming environment for integrated parallelism in the 93/94 timeframe. A proposal has been sent to FRN and the Knut and Alice Wallenbergs foundation. We will also participate in the Swedish High-Performance Computing Network project (SHPCNet). It is one of the prerequisites of the national supercomputing infrastructure in Sweden. However, we would like to stress that to make the infrastructure balanced and complete necessitates large Swedish investments in education, training and supercomputer hardware. It is vital that these hardware investments provide *solution capability*. This implies both a very large primary memory and competitive computational power. We view a MIMD computer as an important continuation and a natural extension to the work that has already begun on the CM2 and CM200.

We view a MIMD computer as an important continuation and a natural extension to the work that has already begun on the CM2 and CM200

An increase in funding for user support would greatly contribute to making present and future investments in hardware become even better used. In addition, to follow the strong international trend towards the establishment of the field “Parallel Computational Science”, we feel it is the right time to further strengthen our role as a national center. There are many good reasons for this:

- The overall development in the parallel computing area has been very rapid in the last couple of years. Both software and solution methods have now evolved to a point where parallel computers are not only a promising technology for the future, but can directly be applied to solve many large industrial and scientific problems.

- The interest for “high performance” computers has been boosted by the High Performance Computing and Communication (HPCC) initiative, already underway in the USA, and by the High Performance Computing and Networking (HPCN) project in Europe. Also, because of a growing insight into the present and future potential of large engineering and scientific calculations.

- The corresponding Swedish SHPCNet initiative with a computer network that initially will link universities with supercomputer installations. This is a first step towards realistic distributed supercomputing, meaning that there will be a large opportunity to efficiently execute demanding applications over the network.

- In Sweden, high-performance computing is less utilized – in industry especially – compared to most other highly industrialized countries. A large investment to develop the use of parallel computing would make it possible to put Sweden back as a leader in industrial computing.

- Introducing new techniques, such as parallel programming, takes several years and will require considerable efforts in education, training and programming. If we start now Sweden can be well prepared when parallel computing becomes dominant in high-performance computing.

Central to continue to operate without having to charge users

It is central to continue to operate without having to charge users, since, today, the time between inception of the idea to use a parallel computer in a particular project and its completion is often long. In the event that we would have to charge projects for the use of the machines it would mean that the turnaround time would be extended and the build up and transfer of knowledge would be slowed down. It would also discourage the necessary creativity, if each experiment and new idea a user would like to try on the machine would burden the individual project economically.

Crucial that we can secure stable and long-term funding

We would like to point out that PDC has funding only until July 1993. If we are to continue operating as a national resource, it is crucial that we can secure stable and long-term 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.

2 Color Plates

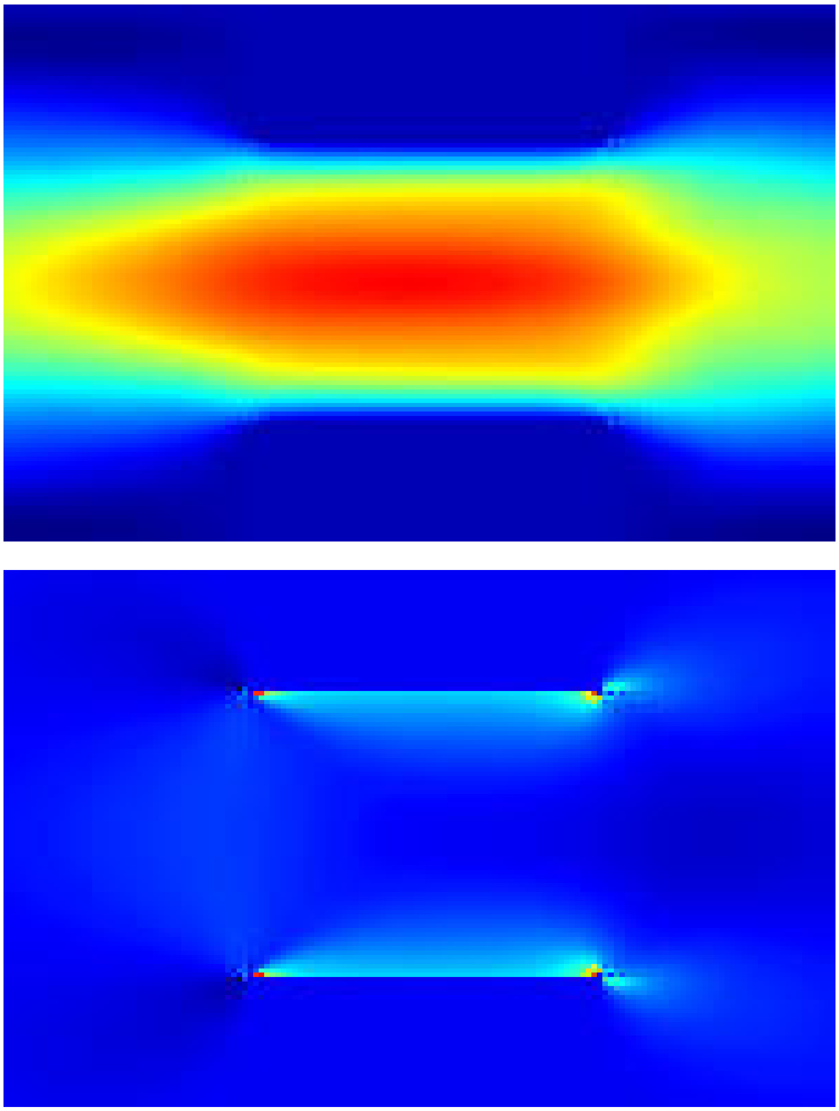

Figure 2.1. Contour plots of the velocity and normal stress component in the horizontal direction in a transient simulation of an UCM fluid in planar contraction. The velocity-field (top) is smooth and well resolved but the stress-field (bottom) shows singular behavior around the sharp corners; the stress components vary between extremal values in the region of just a few gridpoints from the corners. The problems do not disappear on a finer grid and this behavior together with numerical errors finally makes one of the eigenvalues of the stress tensor negative so that the computation breaks down. This seems to be one of the reasons for failure in visco-elastic simulations of this flow-geometry. (See Section 4.3 on page 34.)

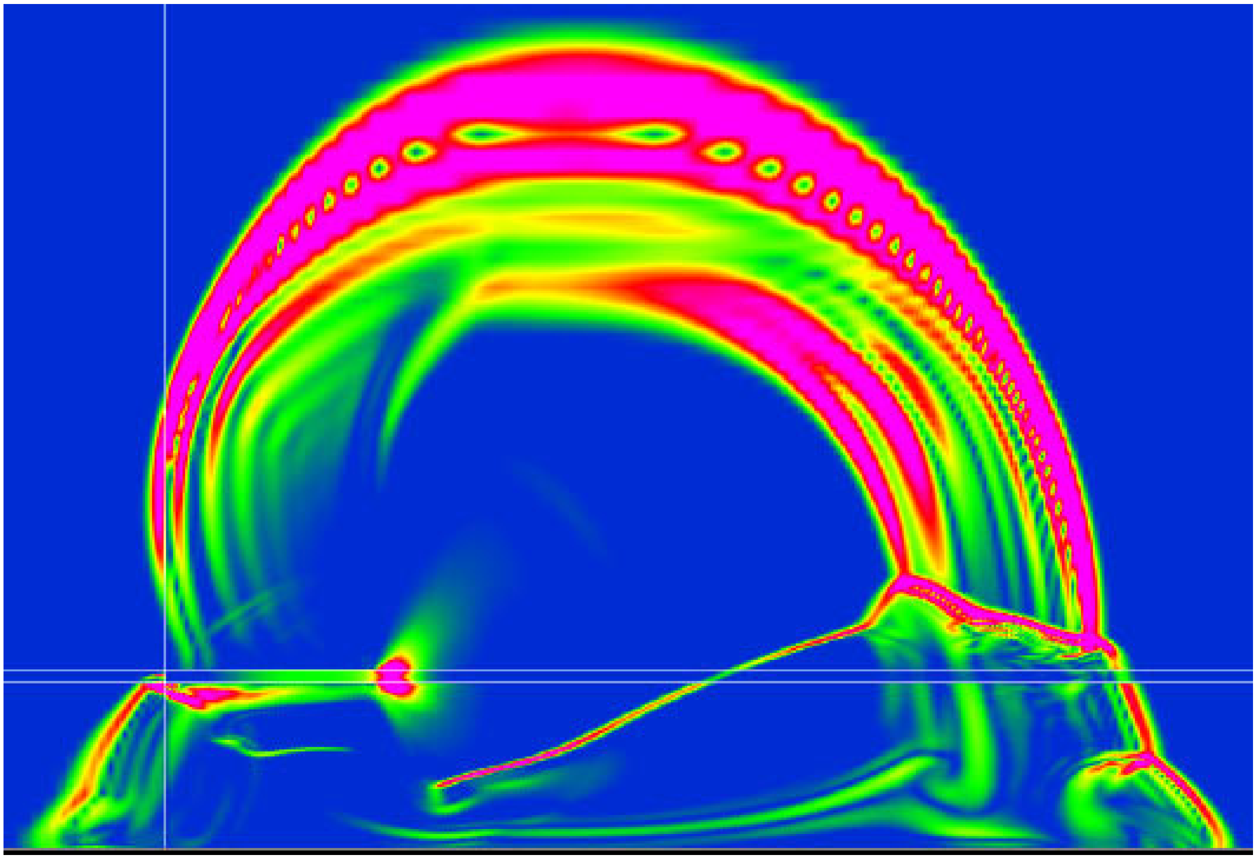

Figure 2.2. Blas-twave computations. In collaboration with FOA we computed the muzzle-blast from a recoil-free hand-held gun. The picture shows the computed pressure distribution. Note the complex interactions of the direct wave and waves reflected from the ground. The computation was done using 6 blocks of different size using a total of 85,000 points. (See Section 4.2 on page 30.)

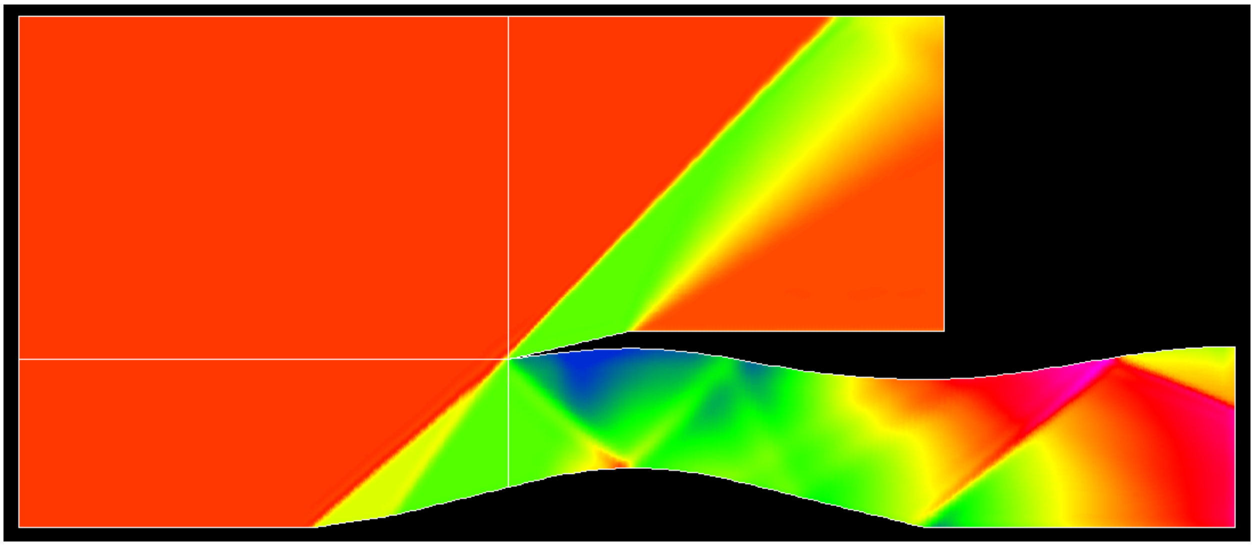

Figure 2.3. Supersonic computations using an air-intake geometry from SNECMA, France. The free-stream Mach-number is set to 1.865; the figure shows iso-mach contours. The shock system in the intake effects the thermodynamically efficient compression for the jet engine further downstream. The intake must be properly shaped for the compression to work over a range of flight-conditions. The computation uses four blocks of 125^2 each. (See Section 4.2 on page 30.)

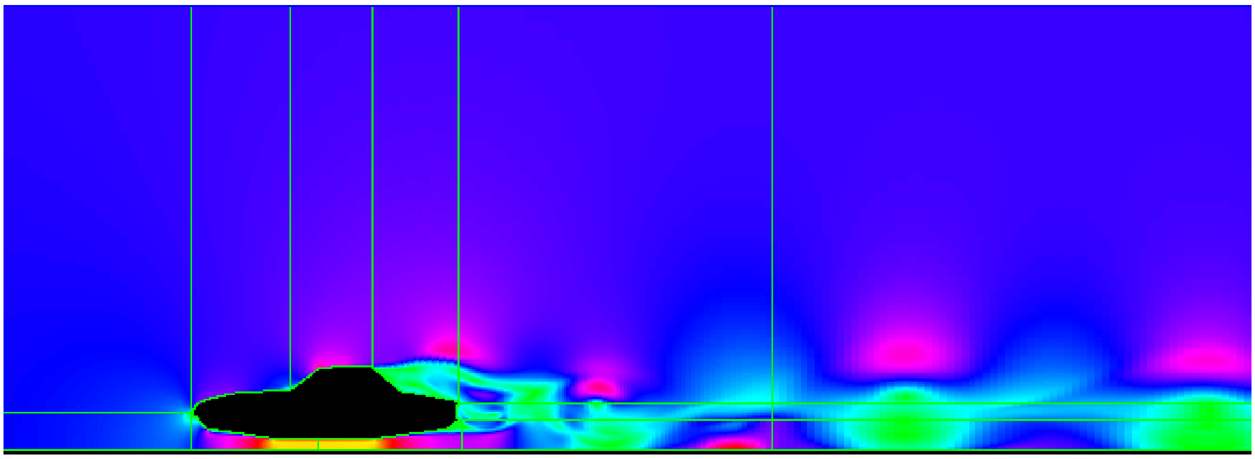

Figure 2.4. Contour plot of the horizontal velocity, computed for flow over the mid-section of a sports car. High velocities are represented in cyan/magenta and low values in green. The block boundaries are represented by lines; only 13 of the 16 blocks are shown in these plots. This computation uses 125^2 points in each block. (See Section 4.2 on page 30.)

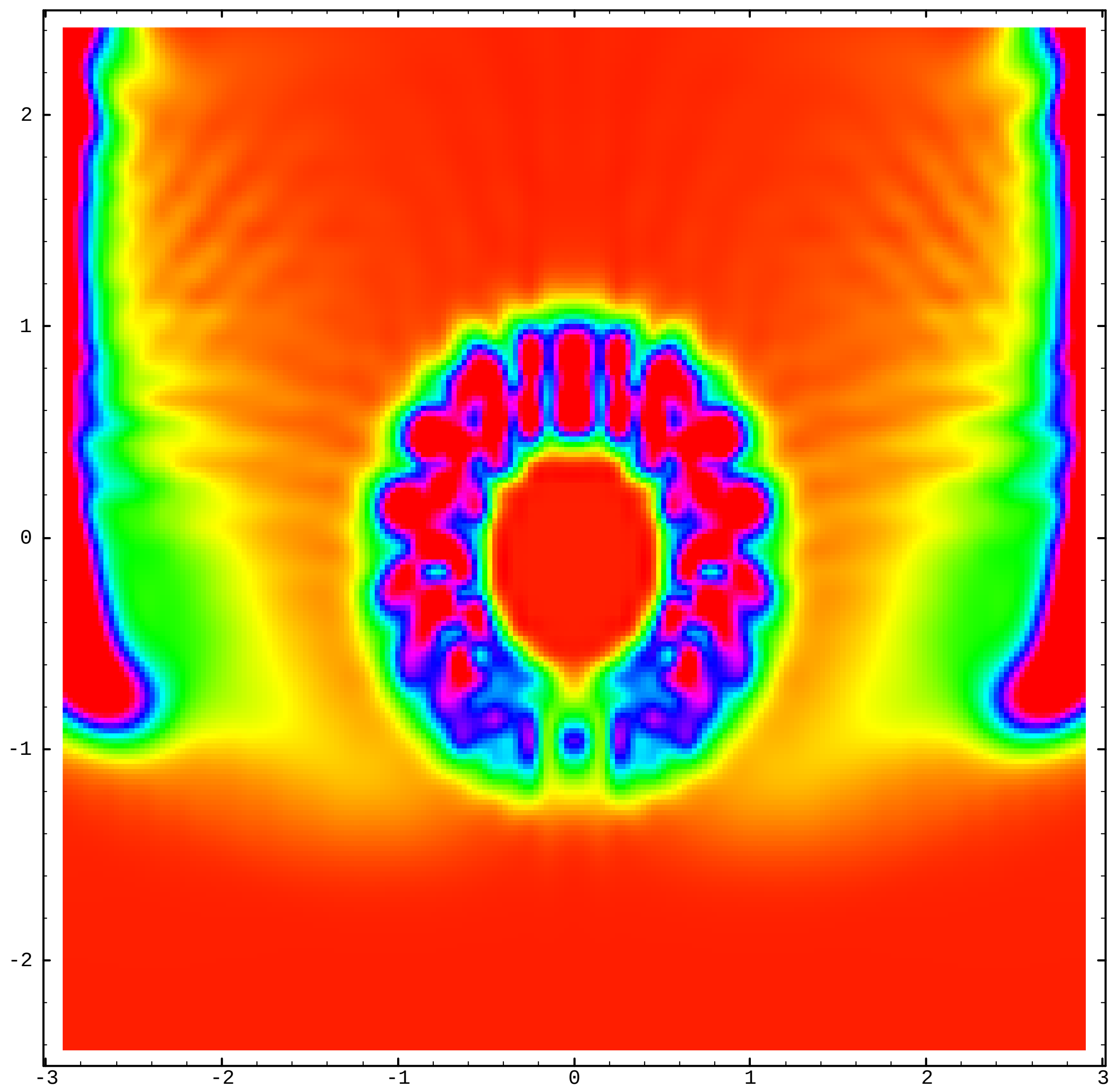

Figure 2.5. Contour plot of the reduced two-dimensional probability density of a 3D wavepacket describing the formation of a molecule by an Eley-Rideal reaction of two hydrogen atoms at a surface. It gives the probability to find the atoms in a relative position b perpendicular to the surface and with a relative distance a parallel to the surface. Note that the wavepacket has been defined for negative a by a reflection in the $a = 0$ axis. (See Section 5.2 on page 42.)

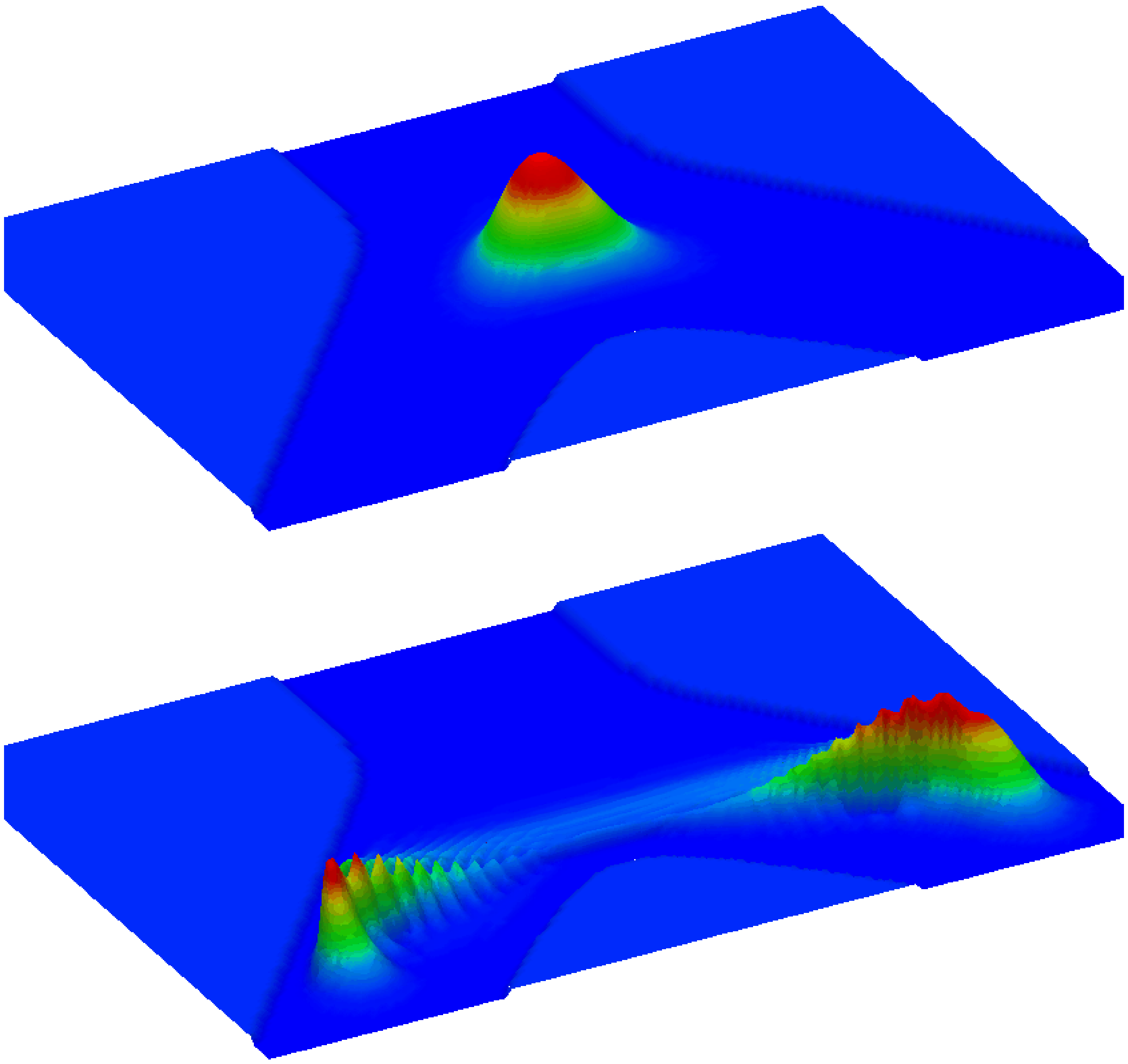

Figure 2.6. A simulation of an electron moving through a nanometer size Y-branch switch. The movement of the electron has been calculated using the Schrödinger equation. The height of the "bump" represents the probability of finding the electron at that point. With a potential difference of 0.4 V between the left and the right electrode, it is clearly seen how the electron, i.e. the current, primarily enters the right branch. (See Section 5.1 on page 41.)

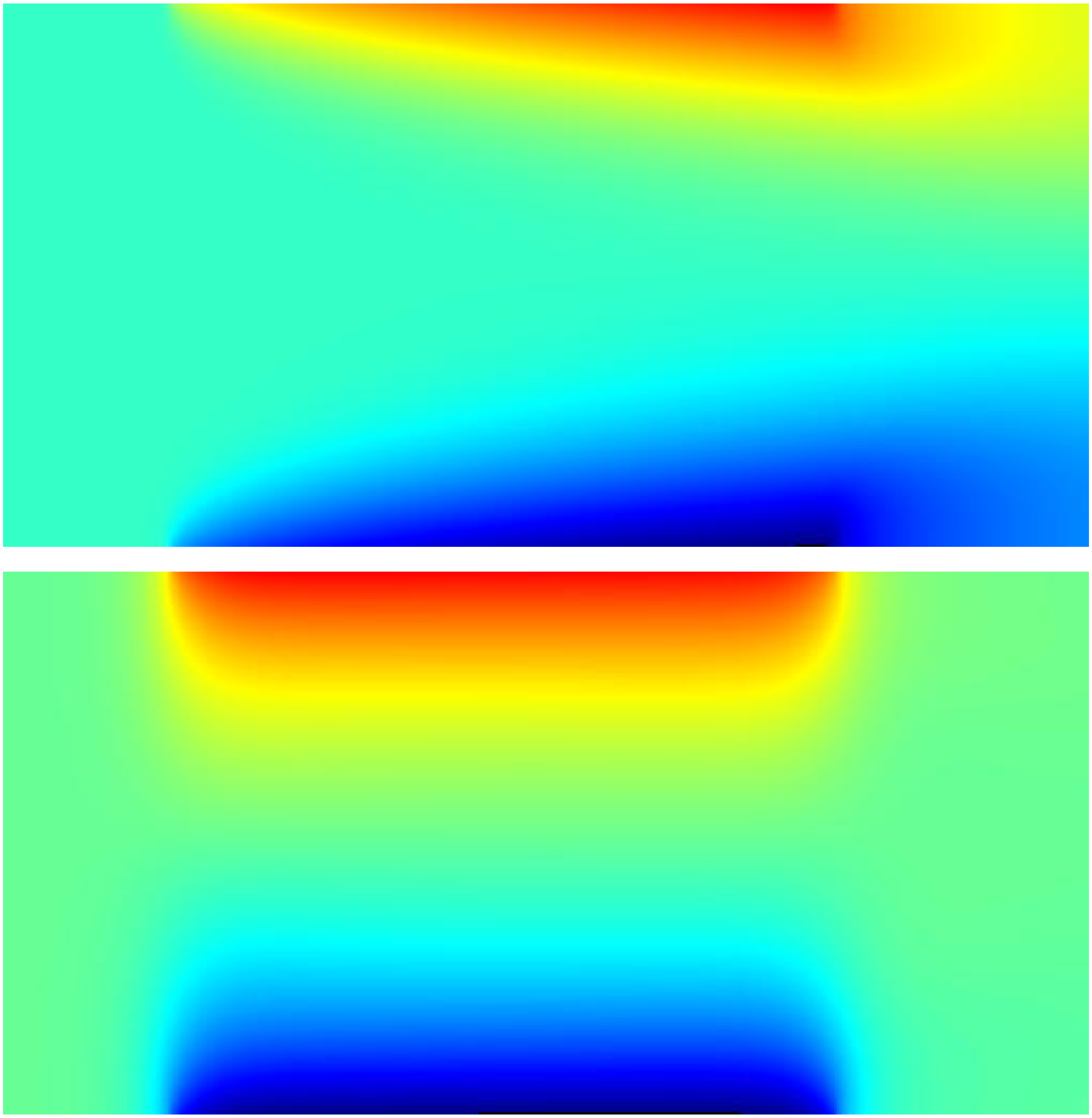

Figure 2.7. The concentration gradients (top) and electric potential field (bottom) for two dimensional forced convection flow in channel, with $Pe = 100$. (See Section 5.8 on page 52.)

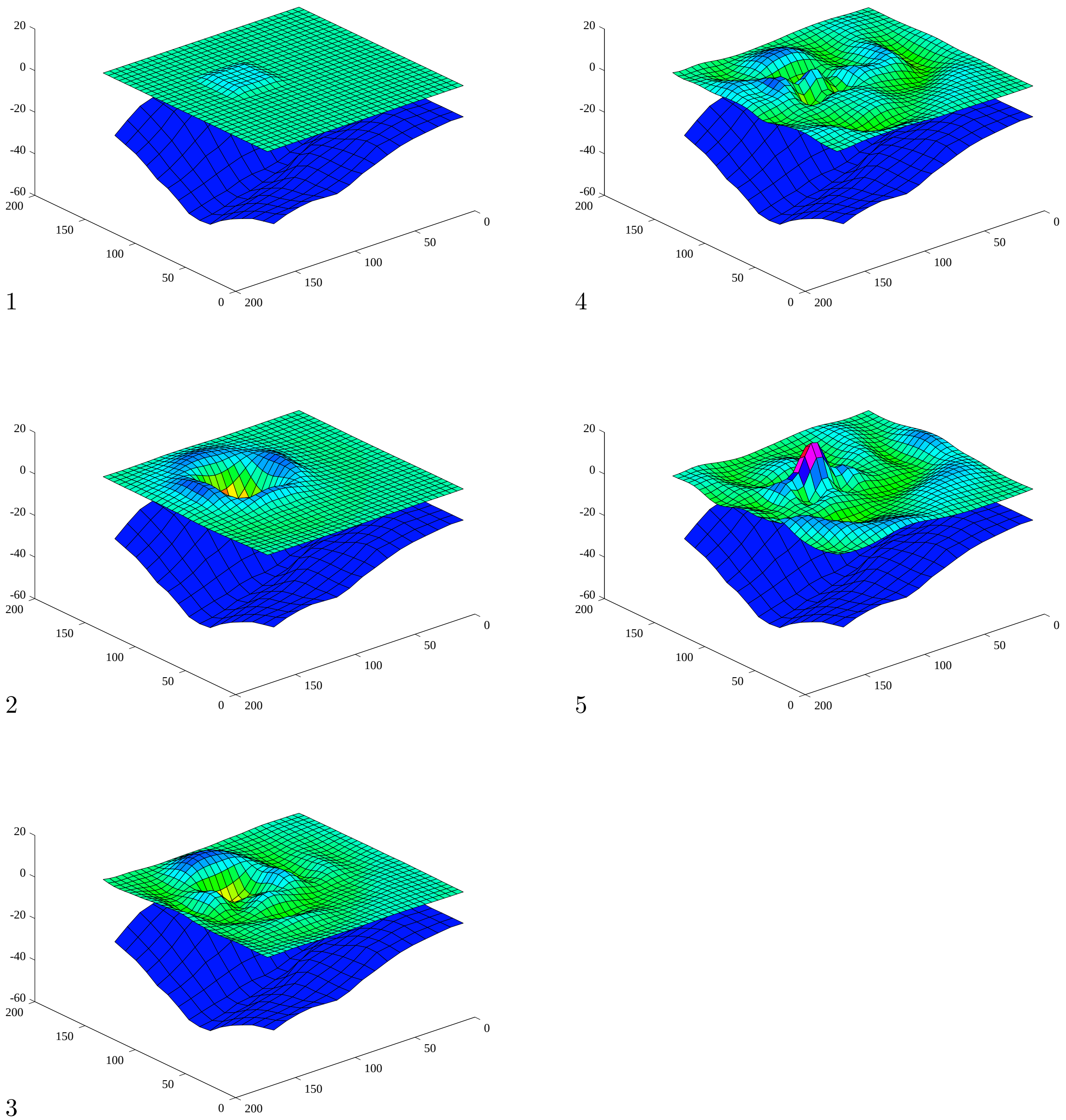

Figure 2.8. Plot of total displacement of the surface from time 0.068 s to time 1.87 s with a time difference of 0.45 s. The vertical displacement is magnified by a factor 2000. The unit of the plot is meters. Bedrock is plotted in blue color. (See Section 5.4 on page 46.)

Figure 2.9. An illustration of seismic 2D waveform modeling. It shows the evolution in time of the wavefield as the energy, injected as an acoustic point source, propagates and undergoes reflections and refractions from discontinuities in the acoustic properties of the model. The properties of the medium vary with depth. The pictures represent snapshots taken from the framebuffer as calculations proceed. (See Section 8.1 on page 63.)

This figure is only available in the paper version

This figure is only available in the paper version

This figure is only available in the paper version

3 Neural Modeling and Computation

Most work in the field of Neural Networks (NNs) is today done on standard sequential von Neumann machines. It has been pointed out many times that such sequential computers are quite inefficient for execution of NNs, as the networks themselves are inherently parallel. For this and other reasons it is important to implement the NN-algorithms and architectures on parallel hardware. Above all, this enables high computational capacity and therefore also the simulation of more realistically sized NNs. This allows us to investigate the scaling properties of our algorithms as problem size increases. It also opens up the possibility of escaping from "toy problems" to some real-world applications. A second, perhaps less obvious, reason for this implementation research is that the use of parallel hardware puts relevant constraints on algorithm development. In this way we avoid inadequate solutions with embedded sequential sections. The fact that we have convenient access to a massively parallel computer (an 8K CM200) at our institute further enhances our possibilities for conducting the projects described below.

Enables the simulation of more realistically sized Neural Networks

3.1 Protein Sequence Matching

Björn Levin, Anders Lansner
SANS, KTH

Higher-order units are frequently needed to improve the performance of our network models. These units can be regarded as "specialized sensors" or "feature detectors." In this project we have built such units by random formation followed by a selection based on different statistical measures. These measures have no knowledge of what type of data the primary sensors of the complex unit operate on. Here we are using a feed-forward network that leads from letter (character) sensors to units associated with documents. A Bayesian learning rule (SANS I, [Lansner and Ekeberg, 1989]) is used to calculate connection strengths and unit biases.

The methods have been applied to both *free text document re-*

trieval and *protein sequence matching* [Levin and Lansner, 1992]. Given a short or long description of the topic of interest, document retrieval is in this case defined as finding the documents with a content closest to that topic. Protein sequence matching consists of ordering a given set of protein sequences according to similarity with one target sequence. It should be noted that, as all of these algorithms are independent of the elements that the complex units are formed from, it is possible to use these methods in future systems in other problem domains. The work has been carried out using the CM and has resulted in a program that compares favorably with systems based on other search schemes. For instance, the retrieval of best matches to a protein sequence of length 200 in a database containing 20,000 sequences is accomplished in 3.5 seconds. The resulting ordering has been compared with the result from an "editing distance"-based system and found to agree quite well (See Section 6.1) [Wallin, 1991]. However, in that system, the same best-match search took about 10 minutes. Our system needs a "training" phase to form and select its complex units as well as to form the connections from sensors to documents. This phase currently takes around 1.5 hours for the above-mentioned protein data base. On the other hand, it only has to be done *once* per database and even allows fast addition of new documents, as long as the text is homogeneous. (See also Section 6.1 on page 56)

Retrieval of best matches to a protein sequence of length 200 in a database containing 20,000 sequences is accomplished in 3.5 seconds

3.2 Relaxation and Learning in Large Networks

Per Hammarlund, Anders Lansner
SANS, KTH

Implementing the basic Bayesian model on the CM2 was explored first using the SANS I in [Lansner and Ekeberg, 1989] and [Levin, 1990]. Here we continue to implement the SANS model with different mappings of the data to the CM2. We also try to rewrite and modify the basic algorithm implementing this model to better fit the architecture of the CM2.

It is very often said that ANNs have inherent parallelism. In fact, there are numerous possibilities of how parallelization can be done. 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.

Have involved writing microcode for the CM2

These implementations have involved writing microcode (CMIS) for the CM2 to get support for primitives that are not present in the instruction set of the machine. This project has also involved optimizing the storage and handling of very large, sparse activity patterns both on the CM2 and on the front-end computer [Hammarlund and Lansner, 1992, Hammarlund *et al.*, 1993].

In the implementation of the fully connected case, one vp-set is used and each unit is mapped to a virtual-processor. In each virtual processor the incoming connections, i.e. the weights, are stored in an array. They can be stored either as floating-point values or as integers of adjustable length. Learning and relaxation can be done with only broadcast-type communication. In the sparsely connected case, two vp-sets are used, one for the units and one for a "compacted" array of connection weights. Here general communication is required between the two vp-sets.

On an 8K CM200 with 1 GByte main memory we can run fully connected networks with 8–16K units using low-resolution weights and sparsely connected networks with up to about 64K 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 takes well below one second. In the sparsely connected case, learning as well as relaxation is somewhat slower.

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 algorithm implementing the SANS I model, to make it fit the architecture of the CM2 better. 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 use only primitives for which we can implement efficient code. Together with the PDP group at the IT, work is also done in the area of languages suitable for describing these computations; most traditional parallel languages are not optimal for describing the sparse computations in parallel ANN implementations [Hammarlund and Lisper, 1992, Hammarlund and Lisper, 1993]. (See also Section 10.1.)

3.3 BIOSIM: a Backend to SWIM

Per Hammarlund, Björn Levin, Anders Lansner
SANS, KTH

The BIOSIM program for the CM2 [Levin *et al.*, 1990, Hammarlund *et al.*, 1991, Hammarlund *et al.*, 1992b, Hammarlund *et al.*, 1992a] is aimed at giving researchers the opportunity to run very large biologically realistic simulations. On an 8K CM200 BIOSIM is capable of simulating, for instance, 32 thousand cells and four million connections or 128 thousand cells and one million connections. Simulation-times are typically in the order of minutes to hours per second simulated time.

BIOSIM is integrated with SWIM, a simulator for workstation environments [Ekeberg *et al.*, 1993, Ekeberg *et al.*, 1990], in such a way that the transition from simulating smaller networks, or parts of the complete network, to simulating the complete, full size network is easy. The researcher need not bother with the particularities of the CM2; the interface is still SWIM and its specification language.

Take advantage of the inherent SIMD-type parallelism in this type of simulation

The BIOSIM program has been written to take advantage of the inherent SIMD-type parallelism in this type of simulation and also to make the most use of the CM2s floating-point hardware and fast communication network. Informally speaking, this implies solving the stiff, coupled differential equation of the same form in parallel, and also to handle synaptic communication in parallel.

Work is currently under way to improve the handling of the very large amounts of simulation data. One bottleneck of the current system is the sequential parsing and inheriting process of the object-oriented specification file. This is being implemented in parallel on the CM2 – inheritance of the object-oriented specification of a neural network will be done in parallel. Work is also done to analyze the simulation result in parallel on the CM instead of bringing large amounts of data to the front-end computer.

3.4 Hebbian Cell Assemblies

Erik Fransén, Anders Lansner, Hans Liljenström
SANS, KTH

Recurrent Neural Network (NN) models, so-called "attractor networks", can be regarded as mathematical instantiations of Hebb's cell assembly theory [Hebb, 1949] of cortical associative memory.

To investigate the biological relevance of these abstract models we have tried to make a network comprised of biological neurons operate as a content addressable memory (CAM) in much the same way as e.g. a Hopfield network. In particular, we have studied the ability of such a network after-activity as proposed by Hebb, and to perform pattern completion and reconstruction.

In the first study, nine neurons were simulated using the Spine simulator [Fransén and Lansner, 1990] running on the Apple Macintosh II. The following study, with a network of fifty neurons, using the simulator SWIM [Ekeberg *et al.*, 1991] for workstations, verified the earlier results: neurons modelled on spinal motor neurons are not able to support after-activity, whereas when using cortical pyramidal cell models, the network function can be quite similar to that of a recurrent ANN [Lansner and Fransén, 1992]. The operation of this network is now studied when effects from various neuro modulators are applied. In this way the different operations of the network can be enhanced or inhibited. A larger assembly size for the motor neuron network has been tested, showing preserved functional features. The assembly operations were shown to be robust when adding geometry-dependent axonal time-delays up to 10 ms [Fransén *et al.*, 1993].

A more elaborate network model of a cortical "microcolumn" is now under study. In the first step two new cell types have been developed. One version with 50 columns (750 neurons) has been used to explore features of the column model. Tests with a larger network (2000 neurons and 200,000 synapses) have been done with the BIOSIM simulator for the CM2.

4 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is the art and science of computational modeling of fluid-flow phenomena under various physical conditions. The mathematical models are variants of the non-linear Navier–Stokes equations which can have very complicated solutions. Numerical simulations of these problems are therefore a necessary complement to the theoretical and experimental analyses. CFD is also pursued as an engineering discipline because it leads to better products and to savings in the design process.

Make progress by constructing computationally efficient methods and by exploring the possibilities offered by massively parallel computers

Most CFD simulations today are made with discretizations which are not sufficient to completely resolve all interesting physical scales. What is lacking is both computational power and physical memory size. However, it is still possible to make progress by constructing more efficient numerical methods and by exploring the new possibilities offered by massively parallel computers.

The CM is routinely used for both pilot studies and production runs

The CM is now routinely used for both pilot studies of a problem and for production runs with programs already developed. The method development projects under way in the CFD area explore several different directions: methods for unstructured meshes (Section 4.1), adaptive finite-element methods (Section 4.4), finite-volume methods (Section 4.2) and high-order finite-difference methods (Section 4.6). The more application-oriented projects cover non-newtonian fluid flow (Section 4.3) and the initial stages of a study of detonation wave phenomena (Section 4.5).

The CFD projects have also inspired international contacts with groups working at EPFL, Switzerland and CERFACS, France.

4.1 Compressible Flow Computations using Unstructured Meshes

Lars Bomholt, Rémi Choquet, Pénélope Leyland
IMHEF, EPFL, Switzerland

Massively parallel computers of SIMD-type have been very successful in solving partial differential equations on structured grids. The use of unstructured meshes, however, has many advantages for modeling as well as for computation, as good solutions can be obtained at a relatively low cost using conceptually simple adaptive mesh refinement in critical areas. Unfortunately, the implementation of such codes is not straightforward. The data structures are generally more complex than for programs using structured grids, and the communication patterns associated with unstructured meshes can lead to low performance.

The aim of our project is to study the problem of efficiently implementing such codes for compressible flow simulations on massively parallel machines. As a practical example, two different types of numerical schemes for solving the 2D compressible Euler equations have been implemented on the CM200 in Fortran 90 using the data-parallel approach.

The first scheme is an explicit/implicit scheme of pseudo-Newton type. It is based on a conservative formulation of the equations and involves flux balance estimations at element interfaces and the resolution of linear matrix subsystem by an iterative method. The second code uses an entropy-variable formulation in a fully non-linear implicit scheme, presenting a complete range of large sparse matrix manipulation techniques as well as the resolution of matrix subsystems. [Bomholt *et al.*, 1992, Bomholt and Leyland, 1992]

A successful implementation of finite-element codes on a fine-grained, massively parallel machine requires communication to be reduced as much as possible, via adapted data structures, algorithms, and machine dependent means such as communication compilation and mapping.

Explicit/implicit code

The underlying numerical scheme is based on an explicit step with a two-step predictor-corrector scheme to compute a residual b. In the explicit version, the time-step for the flow field is directly computed from b. In the implicit scheme, the time step is computed from a linear system of equations $Ax = b$, where the sparse matrix A is the sum of a linear approximation of the operator and a diagonal mass matrix. Space discretization is by triangular $P1$ finite-elements.

Due to its generality and simplicity this scheme forms an ideal basis for studying various aspects of an implementation. The implicit version is also used to compare different iterative methods for solving the sparse matrix equations. Two methods are currently implemented; one is the standard Jacobi relaxation, the second a preconditioned GMRES (generalized minimal residual) method.

In the implementation on the CM200, both element and node data structures are used. The node and element data are partially replicated at elements and nodes, respectively, in order to avoid unnecessary communication. This isolates the communication operations between the node and element data structures (gather and scatter) and allows a better optimization. The remaining elemental and nodal computations do not require any further communication and can be fully executed in parallel. Analogously, an additional node-neighbor data structure is necessary for the assembly and the storage of the sparse matrix A in the implicit scheme.

For communication, such as the scatter and gather operations both the FastGraph communication compiler and the CMSSL communication primitives provided on the CM200 have been used. Both involve a preprocessing step, in which the communication patterns are optimized and stored for subsequent use. Whereas the CMSSL routines allow a quite straightforward implementation of the typical finite-element gather and scatter operations, FastGraph is more general and is therefore more complicated to use. The preprocessing time with FastGraph is considerably longer, but the performance slightly better.

Communication is further optimized by techniques for mapping the data structures onto the distributed memory of the computer. Finding the optimal mapping is, in principle, an NP-complete problem, but with efficient heuristic algorithms a comparably good mapping is found in a reasonable time. For a fine-grained machine like the CM200, simulated annealing and similar deterministic algorithms based on pairwise exchange with optimization of a global cost function have been used with good results. Quite surprising is the efficiency of a simple randomized mapping, which yields almost as good performance at a much lower preprocessing cost.

The following performance results have been obtained by solving the sample problem (25,025 nodes, 49,152 elements) on an 8K CM200. Randomized mapping and the communication primitives from CMSSL version 2.2 have been used.

The following tables show the time spent in the different parts of the program and the true floating-point performance in 32-bit

MFlop/s. The first table shows the relations for the evaluation of the residual b, which represents most of the computation to be performed in the explicit version. The second table shows the relations for a matrix vector product Ax, which is the dominant type of operation in the iterative matrix solvers. The main obstacle to higher performance is the time spent in communication.

The main obstacle to higher performance is the time spent in communication

Program Part		Relative Time in %	MFLOPS
Communication	Gather	24	
	Scatter	38	
Computation	Bulk	24	400
	B.C.	14	3
Total		100	96

Program Part		Relative Time in %	MFLOPS
Communication	Gather	88	
Computation	MatVec	12	680
Total		100	82

A serial Fortran 77 version of the same code has been generated for traditional sequential computers; it has also been fully vectorized for CRAY vector computers. The CPU times for solving a problem on an 8K CM200 and on one processor of a CRAY Y-MP are currently about the same.

Implicit code

The second scheme is a non-linear and implicit. It uses an alternative formulation of the Euler equations. The system is symmetrized with a change to entropy variables; thus the weak formulation automatically enforces the second law of thermodynamics. To improve stability, the Petrov-Galerkin formulation introduced by Hughes is employed. At each time-step of the scheme, a non-linear system of equations needs to be solved. The implementation uses the non-linear Generalized Minimal Residual method (GMRES), an iterative algorithm based on Krylov-subspace projection methods that is very efficient for large problems. It involves a matrix-times-vector assembly, performed directly on the Connection Machine; smaller, dense sub-problems are solved on the front-end by QR-factorization and resolution of successive triangular problems. This method avoids the use of global matrices, which contributes to a reduction in global memory requirements.

Each mesh point and all its neighbors are contained in a local data structure. As a consequence, the element integrals can be evaluated without further communication. This implementation has proved to be slow, because excessive redundant operations were found to be more time-consuming than scatter operations. A second method was thus implemented, using an element-by-element data structure. Then the scatter/gather routines found in CMSSL could be more extensively employed. In the method the most expensive operation was the node-to-node communication when the solution was updated after each step of the GMRES algorithm, whereas in the optimized version this was no longer the case. However, compared to the explicit/implicit code described in section 4.1, it is the actual resolution process of the assembled element integrals which become time-consuming.

An experimental platform for testing non-linear algorithms and their implementation on parallel machines

The first implementation is now completed, but many optimizations remain to be tested. This code is more of an experimental platform for testing non-linear algorithms and their implementation on parallel machines than a candidate for production simulations. The discretization technique involves element-by-element discretization of the governing equations, which is particularly adapted to SIMD machine architectures. The extension of such methods to MIMD machines, using domain decomposition techniques, could be even more promising. In particular, the domain-decomposition approach could act as a preconditioner, enhancing the efficiency of the numerical non-linear Newton type scheme.

4.2 Multi-Block Methods for Compressible Flows

Magnus Bergman
C2M2, KTH
Per Wahlund
TDB, Uppsala University
Mark Sawley, Jon Tegner
IMHEF, EPFL, Switzerland

A study of the parallel computation of 2D/3D, inviscid/viscous, compressible flows using structured meshes is being undertaken as a collaboration between PDC/C2M2/KTH and IMHEF/EPFL. The aim of this project is the development of efficient solvers that are portable over a range of parallel architectures.

In order to compute flow in complex geometries using structured meshes, multi-block methods are often employed. In such methods, the flow region is divided into a number of sub-domains (blocks) based on geometrical considerations. Structured meshes are constructed for each of the sub-domains. Such methods, which retain the possibility of using relatively simple numerical algorithms, also lend themselves in a natural way to parallel computation.

Two different forms of parallelization can be distinguished: fine-grain parallelism at the mesh cell level, and coarse-grain parallelism at the block level. Depending on the programming models available, either or both of these forms can be employed. The data-parallel programming model employed by the CM200 is better adapted to fine-grain parallelism, while a control-parallel programming model is generally employed on multi-processor computers. More recent parallel computers, such as the CM5, provide several different programming models. In the present study, both data- and control-parallel models are being studied. [Sawley and Bergman, 1991, Sawley *et al.*, 1992, Sawley, 1993]

The basic code employed in the project was originally designed as a general code for several different computer architectures. The code solves the time-dependent Euler equations (for inviscid flows) or Navier–Stokes equations (for viscous flows) using a finite volume method based on spatial central differencing. Both second- and fourth-order artificial dissipation are added for stability reasons. The system of discretized equations are integrated in time using an explicit five-stage Runge-Kutta scheme, using either global (time-accurate) or local time-stepping.

In an initial phase, the application of the data-parallel approach has been studied for single-block calculations. Both a Fortran 77 version (optimized for vector computers) and Fortran 90 versions (optimized for SIMD computers) of the code have been developed with the goal of examining both the performance and portability of the code on different computer systems. A detailed performance study has been undertaken for a simple 2D inviscid flow problem, using a wide range of computational mesh sizes (and hence, problem sizes). Timings have been obtained for CM200/CM5 and MP-1/MP-2 computers of different size, as well as for a series of CRAY vector computers and a (serial) high-power HP 700 work-

The 8K CM200 performed the computation at 265 MFlop/s, a factor of 1.4 times faster than a single processor CRAY Y-MP

station. Some of the performance values obtained from this study are presented in Figure 4.1. These results reveal that for medium to large problem sizes, the CM200 performs at similar or higher speeds than a single-processor vector computer. For example, the 8K CM200 performed the computation at 265 MFlop/s for the largest mesh considered (limited by the available memory), a factor of 1.4 times faster than a single processor CRAY Y-MP, while a 32K CM200 performed at 1.06 GFlop/s, 2.0 times faster than a single-processor CRAY C90. For small problem sizes, however, the performance of the CM200 is significantly lower, due to the fact that not all the processors are active. (It should be noted that the performance values quoted here for the CM200 are for 32-bit precision only; this was found to be of sufficient accuracy for the present flow problem.) Further details concerning this study can be found in [Sawley and Bergman, 1993].

The Fortran 90 versions optimized for the CM200 and MP-1, differ slightly because of the different syntax of the data distribution directives and the different relative efficiencies of shifting data; the CSHIFT intrinsic is more efficient than array syntax on the CM200. Nevertheless, both version could be compiled without modification using the NAG Fortran 90 to C compiler/translator, while only minor modification (to eliminate currently unavailable intrinsic functions) was needed to compile the code using the CRAY fpp pre-compiler. The Fortran 90 code could therefore be employed to undertake the computations on either a workstation or a vector computer, albeit at a reduced efficiency compared to that obtained using the Fortran 77 code version.

Two different extensions of the basic code to compute more complex flows using the multi-block approach have been implemented on the CM200. The first computes the flow in individual blocks using the data-parallel approach, with each block being computed in a sequential manner.

For the second extension, the basic task of mapping a set of blocks to memory has been studied for both SIMD and MIMD architectures. An incremental one-pass algorithm for the packing problem in 2D has been developed. Several different criteria for the successive placement of the blocks are employed, with blocks being processed in order of descending size. For the problem of mapping a small number of blocks to a larger number of processors, a subdivision algorithm for the 3D case is currently being studied in collaboration with CERFACS.

An implementation that employs a combination of data-parallel and control-parallel methods has also been investigated using the CM200. [Sawley, 1993] While the CM200 is usually considered to be a "pure" SIMD computer, the parallel processing unit is divided into a small number of partitions, e.g. 2×4K for the 8K CM200 at KTH. Each partition can communicate with the front-end via a separate sequencer, and thus independent jobs can be executed on the machine in parallel. Communication and serial data-transfer between the partitions can be made via the front-end, but more efficient transfer is possible via the DataVault. Such a configuration has been employed to execute the multi-block code described above. Each block is computed in a data-parallel manner on a partition, with the transfer of data between partitions being undertaken using simple message-passing style commands (involving barrier synchronization). Unfortunately, for the CM200, even using the parallel route via the datavault, the data-transfer rate was found to be insufficient to allow good performance to be obtained for the multi-block computations. Nevertheless, it is expected that such a combined data/control parallel method may lead to performance advantages for the next generation of distributed-memory parallel computers, such as the CM5. (See Figures 2.2, 2.3, 2.4 on pages 14 and 15.)

4.3 Non-Newtonian Fluid-Flow

Fredrik Olsson, Jacob Yström
C2M2, KTH

Many fluids such as blood, dyes, and yoghurt have complicated relations between stresses and strains. They are usually called non-newtonian fluids. The elastic mechanisms in visco-elastic fluids at solid boundaries are not well known, and one has observed a special type of vortices close to sharp corners, the so-called lip vortices, not seen in newtonian fluids . For visco-elastic fluids, the Upper Convected Maxwell (UCM) model relates the stress tensor to the strain-rate tensor by a relaxation model,

$$\lambda d\tau/dt = \eta\gamma - \tau$$

where λ is the relaxation time, or material memory, γ the strain rate, τ the stress tensor, and η is a viscosity coefficient. It reduces to the familiar Newtonian linear relation in the limit of zero relaxation time. (This conjecture has not yet been completely proven.) The UCM is a prototype of most models in practical use for visco-elastic fluids. Computations with these models are notoriously difficult and often give solution blow-ups. We have shown, however, that the Cauchy problem for an UCM fluid is well-posed and hence short-time numerical calculations should be possible, if the adequate solid-wall boundary conditions can be applied. [Olsson and Yström, 1993]

Computations with these models are notoriously difficult and often give solution blow-ups

Flow in a Contracted Channel

Computations on a rectangular contraction were performed using the CM200. The geometry used is of great interest both from the theoretical point of view and in applications: the coating of paper and the dispensing of dairy products both use flow geometries that are contracted channels. In the paper application, it is of interest to control the swelling of the fluid after the contraction, and in the dairy application the stresses themselves must be monitored lest the product quality be impaired. Resolution of the extremely high stress gradients at corners requires a large number of grid points. The CM simulations were done with sharp corners. The blow-up does not go away with increased spatial resolution. This indicates that the (unphysical) sharp corners induce breakdown of solutions. We are currently computing the dependence of the flow, in particular the stress gradients, on the corner radii. (See Figure 2.1 on page 13.)

4.4 Adaptive Finite-Element Methods

Kenneth Eriksson, Peter Hansbo, Claes Johnson
Department of Mathematics, CTH

The main objective of our work is to investigate principles for the implementation of adaptive finite-element methods on massively parallel architectures such as the Connection Machine (CM). Specifically, we have considered the implementation of the streamline diffusion (SD) method, which is a general finite-element method for hyperbolic-type problems such as convection-diffusion problems and the Euler and Navier–Stokes equations of incompressible and compressible flow. Our work so far has focused on SD methods for compressible flow on unstructured grids in 2D and SD methods for the neutron transport equation. We first give a brief description of the SD method, and then turn to parallelization aspects of the above items. For a complete account, see [Hansbo and Johnson, 1992] and [Eriksson *et al.*, 1992].

The SD method is a modified Galerkin method based on space-time finite-element discretization with piecewise polynomial basis functions, discontinuous in time and continuous in space. The SD method contains the following two modifications of the standard Galerkin method: a weighted least-squares modification giving control of the residual of the finite-element solution, and an introduction of artificial viscosity depending on the local residual and the local mesh size. The residual is obtained essentially by plugging in the discrete solution into the continuous equation. The role of the modifications is, roughly speaking, to increase stability without sacrificing accuracy.

We have done an implementation on the CM200 of the space-time oriented SD method for time-dependent compressible flow on unstructured grids in 2D. In an unstructured finite-element code, two primary data sets are needed: elements and nodes. Typically, each element and each node is independently associated with a processor. The primary operations in each time-step for a time-dependent problem concern formation of element-level matrices and residual vectors and solution of a global system of equations.

The coupling of elements and nodes is usually represented by an array `ITRI` such that `ITRI(i,j)` contains the global node number of node `i` in element `j`.

Information must be passed from nodes to elements in a gather operation and from elements to nodes in a scatter operation. On an unstructured mesh this requires general communication. The gather and scatter operations are supported through the CMSSL library. It contains routines that help the programmer set up a static communication pattern, which allows efficient communication as long as the mesh is not changed.

SD methods for the neutron transport equation

We have conducted a study of the parallelization of numerical methods for the neutron transport equation. This equation serves as a model for the kinetic equations of gas dynamics, such as the Boltzmann and Vlasov equations. These models play a fundamental role in physics and astrophysics, and their solution requires massive computational work because of the seven independent variables x, v, t, where x is a space coordinate, v is velocity and t is time. Thus even a moderate number of degrees of freedom in each variable will make the total number of degrees of freedom very large, and only massive parallelization appears to be able to provide the required computational power. We have studied different parallelization strategies, e.g. parallel over x and sequential over v and vice versa, for numerical handling of the neutron transport equation. We have completed an implementation in a model case, where the SD method is used to solve the transport equation for each given velocity, i.e. parallel over x and sequential over v.

We give a numerical example concerning neutron transport in 2D with initial angular flux equal to 1.0 in a disc of radius 0.25 for all angles, and with given source equal to zero. The computational domain, $(0,1) \times (0,1)$, was divided into 90×90 elements, and the time-step was set to $k = 0.03$. In Figure 4.2 we show part of the mesh and the scalar flux after one time-step, using 40 discrete angles. In Figure 4.3 we show the scalar flux after 10 time-steps, using 20 discrete angles (left) and 40 discrete angles (right).

Conclusion

Efficient implementation of finite-element methods on the CM system is possible

Our research so far indicates that efficient implementation of finite-element methods on the CM system is possible on fixed unstruc-

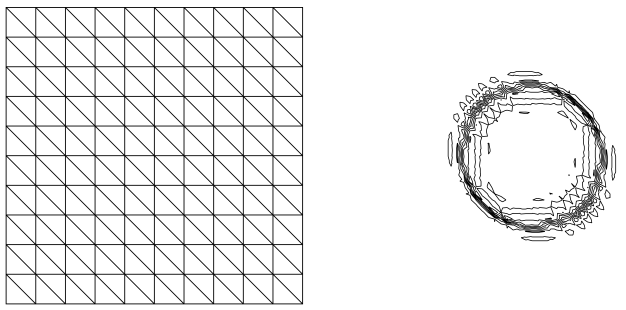

Figure 4.2. Part of the mesh and scalar flux after one time-step.

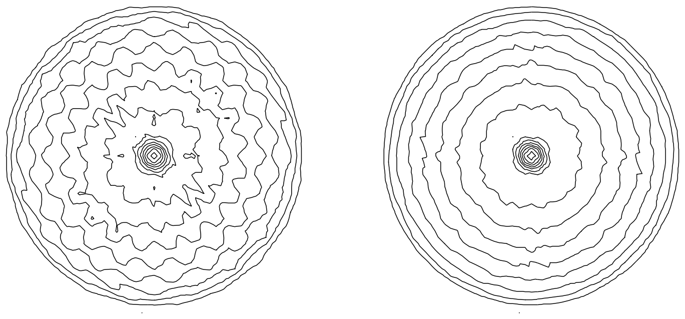

Figure 4.3. Scalar flux after 10 timesteps, using 20 discrete angles (left) and 40 discrete angles (right).

tured grids. In our future work we plan to concentrate on problems related to dynamic adaptive mesh modification.

4.5 Detonation Waves in 2D and 3D

Björn Sjögreen
CERFACS, Toulouse, France
Jean-Christophe Thil
University of Grenoble, France

The Euler equations in gas dynamics are routinely solved for the simulation of gas flows. We have studied the flow of a gas in which an exothermic chemical reaction is taking place. When a shock wave is sent into a premixed reactive gas, the gas is ignited by, and a reaction zone develops behind the wave. This is called a detonation wave.

Similar phenomena can be found in the flow past a hypersonic vehicle during reentry from orbit. The extreme heat in such a flow dissociates the air molecules. Furthermore, it has been proposed to use scramjet engines with a standing detonation wave to propel aircraft at extremely high velocities. The numerical simulation of these phenomena is therefore of great interest.

It turns out that there are numerical difficulties associated with this problem, due to the very large reaction rates, which give rise to a very thin reaction zone. If this zone is not resolved by the numerical computation, incorrect wave velocities are obtained. We have previously studied this phenomenon [Engquist and Sjögreen, 1991, Engquist and Sjögreen, 1989] and tried to design a better numerical method. We now plan to extend these results to a problem with a larger number of chemical reactions.

In the beginning of 1992, a code to simulate two-dimensional detonations with one reaction was implemented on the Connection Machine. In the spring of 1992 the code was extended to include a hydrogen-oxygen model. This work was done for a master's thesis.

Our experience with CM Fortran is overall positive. Some peculiarities have been noted, however. As an example, an array with four quantities at each grid point is best declared `u(4,ni,nj)`. The code with declarations `u(ni,nj,4)` is about 50 times slower. This is unsatisfactory. Compilation and linking is somewhat slow: compiling and linking the entire code takes about half an hour. Still, the ease of access to the CM is a very big advantage, which greatly eased the work of implementing and debugging the code.

The ease of access to the CM is a very big advantage

Performance on an 8K CM200 is around 100 MFlop/s in the computationally intense parts of the code. The program has been run on a CRAY X-MP at 50 MFlop/s, on an Alliant FX80 at 10 MFlop/s; and a similar, but three-dimensional computation on a 32 processor *i*PSC/2 hypercube ran at 8 MFlop/s. So far, we have been able to run the two-dimensional one-step chemistry program on the CM. The multi-component chemistry is under development.

4.6 Data-Parallel High-Order Difference Methods

Pelle Olsson
TDB, Uppsala University

In this project we have studied how fluid-flow equations, such as the Euler and the Navier–Stokes equations, can be solved on a data-parallel computer. In particular, we have derived numerical methods that employ the inherent parallelism of the mathematical-physical problem.

Data-parallelism offers a more natural approach to programming than does the sequential paradigm

In many ways, data parallelism offers a more natural approach to programming than does the sequential paradigm. The typical problem of proper loop ordering very often has little or no bearing at all on the real problem; the physical problem evolves concurrently over the whole domain. The key issues in parallelizing are: communication, concurrency, and memory usage. In general, there is a trade-off between these properties. To reach a high degree of concurrency, or to reduce communication, one may have to use more memory. This type of memory usage has no counterpart on a serial computer. In the context of fluid-flow equations the considerations above pertain to boundary conditions, difference stencils, and coordinate mappings onto the computer. The findings of this research are presented in a doctoral dissertation submitted to the Department of Scientific Computing, Uppsala University [Olsson, 1992]. We give a brief summary below.

To be of practical importance, a numerical method must be *stable*, that is small perturbations cannot be allowed to grow without control. Stability of the numerical method can only be achieved if the original mathematical-physical problem is cast into a form, which suppresses uncontrolled growth. It is shown how this can be done for linear symmetric hyperbolic and parabolic systems in several space dimensions. Existing stability results usually rely on the assumption of smooth domains. We extend these results to include the case when the boundary is non-smooth [Olsson, 1991]. As an illustration we consider the linearized Euler and Navier–Stokes equations, including the energy equation.

Since finite-difference methods are employed, the domain of definition of the discretized PDE is assumed to be diffeomorphic to an n-dimensional hypercube. These domains have the benefit that they can be efficiently mapped onto the computer topology by existing parallel compilers, such as the CM Fortran compiler. We show how to derive *stable* semi-discrete schemes of arbitrary order of accuracy using one-sided difference stencils at the boundaries. The effects of the boundary conditions are taken into account. We refer to [Olsson, 1991] for details.

It is demonstrated that the boundary conditions may be viewed as projection operators. This observation forms the basis for the implementation on parallel computers of SIMD-architecture. On a serial computer, the time spent evaluating the boundary conditions is negligible compared to the time needed for updating the interior points. This may not be the case on a parallel computer. Assuming that the number of boundary operations is of the same order of magnitude at each boundary point, the simplest implementation leads to a total execution time for enforcing the boundary conditions proportional to the number of boundary portions that must be treated separately. Clearly, a highly irregular boundary may reduce the inherent concurrency. We have shown, however, that the boundary conditions for parabolic and hyperbolic PDEs may be concurrently evaluated, even if the boundary is non-smooth. We provide an explicit example how this can be done using CM Fortran in case of the Euler and the Navier–Stokes equations [Durand and El Dabaghi, 1991, Olsson, 1991]. The best performance is achieved if gradient-free boundary conditions are used. We prove that such side conditions indeed yield stable numerical approximations of the Euler and the Navier–Stokes equations.

The current implementation of a fourth order Euler solver using eighth order artificial viscosity runs at about 300 MFlop/s on the CM200

We have also derived boundary-modified, semi-definite artificial viscosity operators of arbitrary order of accuracy. The viscosity operators are presented in a form that is particularly well-suited for the implementation on data-parallel computers. Two CM Fortran arrays per space dimension are needed, independently of the order of dissipation. The current implementation of a fourth-order Euler solver using eighth-order artificial viscosity runs at about 300 MFlop/s (double precision) on the CM200.

5 Applications in Physics

The number of projects in this area has increased considerably; both those doing production runs and those that are in a development phase. Some of the projects study field problems by numerical solution of conservation laws such as the Poisson, Schrödinger or wave equations, for which the CM200 is ideal. It is interesting to note that there are many different types of studies, such as eigenvalue analyses of very large collections of small matrices, a model for turbulent flow made up of particle-like vortices in a quad-tree structure, and Monte Carlo (MC) simulations of spin glasses. We note in passing that simple variants of MC are ideal for the CM200 and that we have seen fewer MC-projects than expected.

The ongoing projects in production come from a variety of fields: supersymmetry in condensed matter (Section 5.10), ground vibrations (Section 5.4), hierarchical models of turbulence (Section 5.5), quantum surface phenomena (Section 5.2), high temperature superconductivity (Section 5.3) and quantum electronics (Section 5.1). Application programs and methods are being developed in areas such as: smooth particle hydrodynamics (Section 5.6), disordered magnetic systems (Section 5.7), evolutionary models (Section 5.9) and electrolytic flow (Section 5.8).

5.1 A Quantum Electronic Y-branch Switch

Thomas Palm
Microwave Engineering, KTH

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, which has been shown to have some advantages over existing transistors. [Palm *et al.*, 1993]

In the first experiments this device is supposed to be implemented using a so called modulation-doped split-gate structure. My model is split into two parts: one semi-classical, describing the distribution of the electric potential and background charges in the structure, and a more exact, fully quantum-mechanical description of the conduction electrons. Both steps rely heavily on the FFT routines in the CMSSL library.

The advantage of a numerical approach

In order to verify the model, I have also simulated other components such as an Aharonov-Bohm interferometer and a point contact. The advantage of a numerical approach is that many different components are very simple to simulate using the same code. (See Figure 2.6 on 17.)

The program is written in C*. This incurs a rather heavy speed penalty over working in CM Fortran which is better supported by Thinking Machines. Nevertheless the advantages of a better language is repaid in reduced development time. In my case simulations rarely take more than a minute per data point, or 20 minutes in total.

5.2 Quantum Wavepacket Studies

Mats Persson
Department of Applied Physics, CTH
Bret Jackson
Department of Chemistry, U. of Massachusetts, Amherst, USA

A wide variety of important surface phenomena like catalysis, surface chemical reactions, plasma-wall interactions, and growth of materials, can only be understood from an analysis of elementary dynamical processes at surfaces. Examples of such processes are sticking, desorption, diffusion, tunneling through and climbing over reaction barriers and gas-surface energy transfer. The rapid advances and developments in experimental methods for the study of such processes like molecular beam scattering, time-resolved laser spectroscopy in the femtosecond domain, and scanning tunneling spectroscopy makes this field a timely subject for theoretical studies.

In many cases a quantum-mechanical description is needed, and the associated complexity often makes it necessary to perform computer simulations of detailed models. We have an ongoing project where we study such processes using pseudospectral methods for *time-dependent propagation of multi-dimensional wavepackets* on the CM. In these methods the multi-dimensional wavefunction is represented on a grid, and in each time-step the action of the kinetic part of the Hamiltonian on the wavefunction is handled by an FFT of the wave-function. The limiting factors in the computations are then determined by the primary memory needed to represent the wave function and the speed of the FFT subroutines. A parallel machine like the CM200 is found to be very well suited to handle these factors.

A parallel machine like the CM200 is found to be very well suited to handle these factors

At the moment we are studying a prototype catalytic surface reaction: the formation of hydrogen molecules by the recombination of an incoming hydrogen atom with an adsorbed hydrogen atom. This reaction is of prime interest in understanding the formation of hydrogen molecules in interstellar space and also for plasma-wall interactions in fusion reactors. Recent experiments have suggested that the reaction follows an Eley-Rideal mechanism, where the incoming atom reacts directly with an adsorbed atom.

We have earlier performed two-dimensional wavepacket studies of this process in a restricted collinear configuration, where we were able to show that this kind of reaction can produce the observed high vibrational excitations of the molecules formed [Jackson and Persson, 1992]. However, this study was done on a SPARC workstation and a CRAY and, due to the limited primary memory on the machines available to us, could not be extended to a less restrictive model yielding full rovibrational distributions and reactive cross-sections. The large primary memory of a CM200 and also its high speed allows us to study a more realistic model. This model includes all six degrees of freedom of the two atoms and uses a potential energy surface which effectively reduces, by introduction of three conserved quantities, the six-dimensional problem to a three-dimensional in curvilinear coordinates. We have developed an efficient method to handle the kinetic part in these coordinates.

The large primary memory of a CM200 allows us to study a more realistic model

We are now obtaining results for reaction probabilities and internal state and translational energy distributions of the formed molecule. In Figure 2.5 on page 16 we show a snapshot of the reduced two-dimensional probability densities of the three-dimensional wavepacket in the reaction zone. As shown in Figure 5.1, the

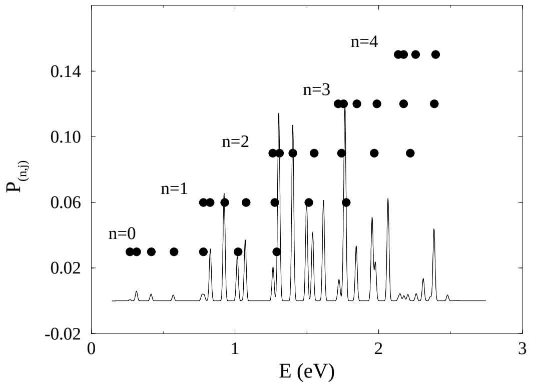

Figure 5.1. Internal rotational and vibrational state distributions of a para-hydrogen molecule formed in an Eley-Rideal reaction at a surface. The heights of the peaks, $P_{(n,j)}$, give the relative probabilities for finding the molecule in rotational states j and vibrational states n. The solid circles give the energy positions of the different rotational states $j = 0, 2, 4, \ldots$ in a definite vibrational state.

produced molecule ends up in highly excited vibrational and rotational states. The large amount of rotational excitations are due to the reaction only taking place for a narrow range of impact parameters. A quantum effect of this steric constraint results in a distribution that is much broader than the result from a classical treatment.

The original Fortran 77 code for the propagation of the wavepacket has been straightforward to transfer to CM Fortran, in part due to its simple structure. The code consists of one part that initializes the incident wavepacket on the grid and runs on the front-end with minor modifications of the code. The time-consuming part is the discrete FFT in the time-propagation. This routine has been replaced by the versatile CMSSL routine, and we have followed the standard recipe for its optimization. We have enjoyed the possibility of using Bellman interactively in the development and debugging phase.

We have enjoyed the possibility of using Bellman interactively in the development and debugging phase

Presently one time-step takes about 1.3 second for a grid size of 128^3 in single precision, and the final state analysis requires about 4000 time-steps. The plot of the reduced two-dimensional probability densities in Figure 2.5 takes about 20 minutes, while the final state internal energy distribution of the formed hydrogen molecule shown in Figure 5.1 takes less than 2 hours. This allows us to make rather extensive studies of the “topological” effects of different potential energy surfaces on the reaction. The large primary memory size of a CM200 also makes it feasible to study the isotope effect on the reaction by considering also deuterium. In this case the larger mass makes it necessary to have a grid size of 256^3. The representation of the wavefunction on such a grid requires a memory space of about 128 MByte for the array in double precision, and we need at least 5 such arrays.

In summary, the computational study of dynamical processes at surfaces using multi-dimensional quantum wavepackets is very computationally demanding and also well-suited for a parallel machine like Bellman. For instance, a calculation using a more realistic potential energy surface requires a higher-dimensional wavepacket, up to six dimensions, which would require a machine with much larger primary memory and speed. The availability of Bellman has made it possible for us to go beyond the simplified two-dimensional models for quantum dynamics to more realistic models. For instance, recent molecular-beam experiments have shown most surprisingly that a weakly van der Waals bonded hydrogen dimer, $(H_2)_2$, survives the collision with a surface, and we are now in a unique position to understand the quantum dynamics of this process by using our program developed on Bellman.

The availability of Bellman has made it possible for us to go beyond the simplified two-dimensional models for quantum dynamics to more realistic models

5.3 Order Parameters in High Temperature Superconductors

Fabian Wenger, Stellan Östlund, Anders Eriksson
Institute for Theoretical Physics, CTH

The phase transition of a material to a superconducting state can be described by an order parameter which measures the tendency of electrons to form pairs. Since the discovery of new superconducting materials with unusually high transition temperatures by Müller and Bednorz in 1986, theorists try to determine suitable order parameters with the correct symmetry properties. Guided by analytical considerations we were led to investigate the case of electrons paired in a state with d-wave symmetry.

To make use of existing experimental data, we calculated the low-lying excitations of the system and determined the resistivity of a tunnel junction consisting of two pieces of superconducting materials separated by a thin insulating region.

Due to the appearance of singularities, a very fine mesh of data points is crucial to get accurate integrals that determine the resistivity. The CM allows an accurate high-speed calculation of these expressions, which greatly enhances the predictive power of the theoretical analysis. We have used the CM as an everyday calculational tool which has provided us with immediate and accurate

We have used the CM as an everyday calculational tool results. [Wenger and Östlund, 1993]

In the future we will continue our investigation in a direction that will be possible only because we have developed a routine on the CM that diagonalizes thousands of small Hermitian matrices in a small fraction of a second.

5.4 Simulation of Ground Vibrations in 3D

Marcus Berglund
C2M2, KTH
Sigurdur Erlingsson
Department of Soil and Rock Mechanics, KTH

A problem of general interest in soil dynamics

Solving the 3D elastic wave equation in the time-domain is of growing interest in engineering practice. This project uses computations of 3D elastic wave propagation to study the vibrations induced in the ground by dynamic loads on the surface. This problem is of general interest in soil dynamics with all kinds of site responses as application, such as earthquakes, machine vibrations, pile drivings and blastings.

The finite-difference technique was chosen in this case because of the large memory of the CM200 and because the simple geometry allows the coefficients of the computational stencil to be precomputed and stored, which substantially increases the performance of the program.

The case studied here dates from 1985. At a rock concert in a large sports arena "Nya Ullevi", Göteborg, Sweden, the audience of 60,000 jumped periodically to the rhythm of the music. This rhythmic motion lead to violent vibrations resulting in structural damage.

Earlier studies have been limited to 2D models by the computing resources. The 2D model is a very crude approximation to reality, since the decay of the amplitude with distance is $1/r$ in 2D, compared to $1/r^2$ in 3D. Also the fact that the bedrock under the stadium is very irregular, see figure 5.2, makes the 3D effects important.

Figure 2.8 on page 19 shows the total displacement of the surface at different times when a periodic vertical driving load of 3000 Pa is applied on part of the surface. A maximum velocity of 18 mm/s was calculated under the structure, which should be compared with the maximum limit of 20 mm/s, given by the German

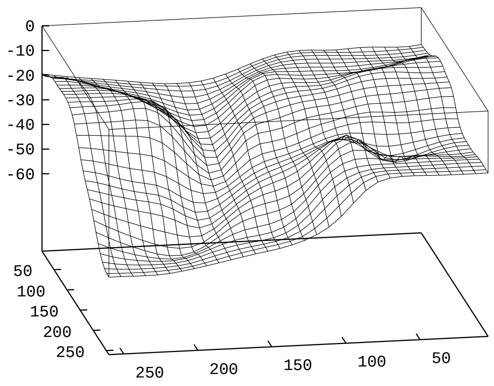

Figure 5.2. Geometry under the sports arena "Nya Ullevi" where bedrock is approximated with splines. The length scale of all three axis is meters.

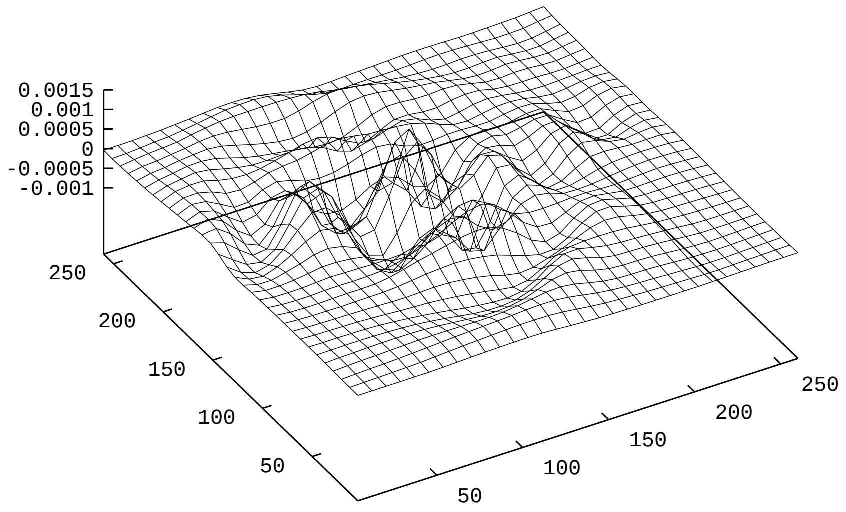

Figure 5.3. Plot of displacements of the surface at time t=1.25 sec. A time-periodic vertical load of 3000 Pa is applied on a part of the surface. Every second point is shown.

building code DIN 4150. With speeds above this limit serious damage starts to occur. [Clayton and Engquist, 1977, Berglund and Erlingsson, 1992, Erlingsson and Berglund, 1992]

5.5 Hierarchical Model of Turbulence

Erik Aurell
Department of Mathematics, Stockholm University
Fredrik Hedman
PDC, KTH
Peter Frick, Vladislav Shaidurov
ICMM, Perm, Russia

The understanding of turbulence in fluid dynamics is one of the main problems in classical physics. In our view, turbulence is a state of matter, analogous to the critical state in phase transitions in statistical physics, which may arise in a great variety of physical systems, as soon as the time-scales of inertial forces are much shorter than those of viscous forces. In that case, the system has time to revolve many times before viscous effects become important, and the resulting motion can be complex and involved, seem erratic and at times display violent bursts, all of which is understood by the word "turbulent".

The ratio between two time-scales is a dimensionless number. In incompressible hydrodynamics, the relevant ratio is the Reynolds number:

$$Re = \frac{t_{\text{inertial}}}{t_{\text{viscous}}} = \frac{LV}{\nu} \tag{5.1}$$

where L is the large spatial scale and V the large-scale velocity, and ν is the kinematic viscosity. Turbulence in hydrodynamics is the asymptotic state when the Reynolds number tends to infinity.

Kolmogorov postulated in 1941 [Kolmogorov, 1941] that there is a hierarchy of scales in turbulence, where energy is transferred steadily from large scales down to small scales. One then finds that the second-order moments, like $< |v_r|^2 >$, scale as $r^{2/3}$, which implies that the energy density at wave-number k decays as $k^{-5/3}$. This famous result is often referred to as the Kolmogorov "$k^{-5/3}$ law".

The experimental results do not contradict the Kolmogorov theory. In fully developed three-dimensional turbulence there is a wide range of excited scales, and direct numerical simulation will be out of reach in the foreseeable future. There are however many theoretical arguments which suggest that there should be corrections to the Kolmogorov laws, just as there are corrections to the mean-field predictions of critical phenomena in three dimensions. These corrections are called "intermittency effects" in turbulence, since they come about from rare but violent and energetic events. One may think of the collision of two large vortices, which tear each other apart and generate intense small-scale motion. Such events give more lasting impressions than the random steady behavior in the Kolmogorov scenario.

In [Aurell *et al.*, 1992] we constructed a hierarchical model of turbulence in two dimensions. We chose two dimensions, since there exist in this case results of many large-scale direct numerical simulations, at relatively high Reynolds number. It is therefore

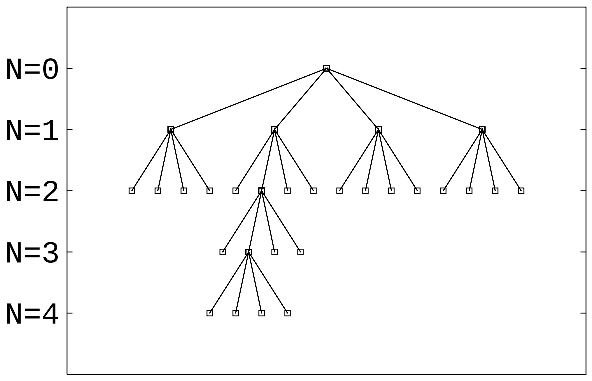

Figure 5.4. A hierarchical tree that displays the structure of interactions between vortices. A vortex on any scale is connected to four vortices at the next smaller scale, and reciprocally to one vortex on the next larger scale. A given vortex is driven by the influence of two others, that may be both below, both above or one above and one below in the tree, but all three always lie on the same branch. The full system which was simulated included 11 levels and 1398101 vortices.

well suited to comparison and evaluation of our model. We note however here, and refer to [Aurell *et al.*, 1992] for the details, that hydrodynamics in two spatial dimensions has many peculiar features, and changes in a major way the Kolmogorov predictions.

The hierarchical model consists of vortices of different sizes that move in the fluid. The vortices are connected in a regular tree, and the relative motions are restricted in such a way that the distances between "parent" vortices and "child vortices" are fixed for all times. We call it a *model* because we have so far not been able to find an exact representation of the hydrodynamical equations in such a basis. If the vortices were *not* allowed to move, one could build a complete, orthogonal hierarchical basis: such bases are today known as *wavelet bases*. We do not believe that one can gain much over traditional methods with such a representation: the very basic feature of fluid-flow is advection by large-scale motions. The deformation of the small scales caused by the strain is a much slower process, although of course very important for long times. We summarize here the main scientific results:

- The spectral law is independent of initial conditions and details of the driving forces.
- The intermittency correction is about 0.3, i.e. about 10% on the two-dimensional spectrum.
- The energy on small scales is concentrated in an asymptotically vanishing fraction of the vortices.
- Quantitative predictions for 2D-turbulence made in [Benzi *et al.*, 1990] also hold for our model.

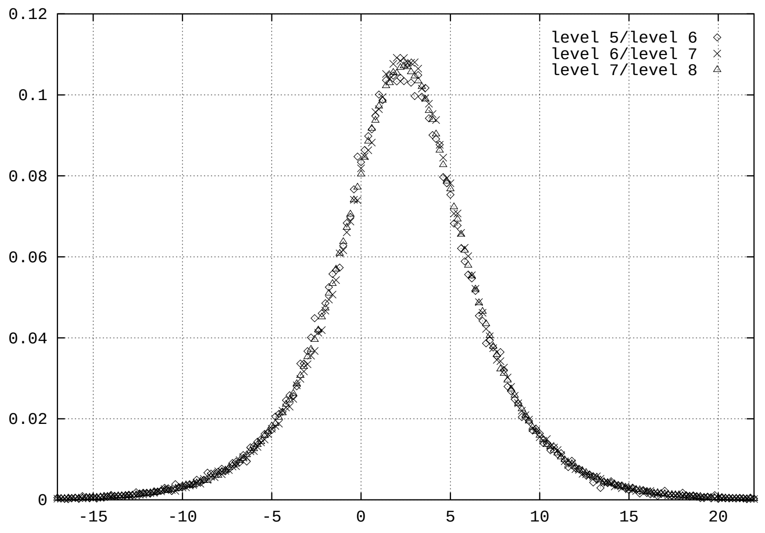

Figure 5.5. Histogram of logarithm of ratios between the enstrophy of parent and child vortices. Different markers correspond to different levels in inertial interval. The histogram shows that the cascade process from scale to scale is the same in the inertial range (within numerical accuracy), and displays a wide range of available local scale exponents.

The program was written mainly by Vladislav Shaidurov, supported by Erik Aurell and Fredrik Hedman. The scientific treatment was done mainly by Erik Aurell and Peter Frick. The present C* code is hence the result of three months' work and as such it is highly respectable: the central part of the algorithm runs at 100 MFlop/s, notwithstanding a quite involved data structure, and we have been able to obtain interesting and new results. The code can be improved considerably by embedding the hierarchical tree of vortices differently into the Connection Machine. We leave this for future work, as well as many interesting questions raised by the results in [Aurell *et al.*, 1992].

This Swedish-Russian cooperation was made possible by guest researcher grants from the Wenner-Gren Center Foundation, for which we are most grateful.

5.6 Smooth Particle Hydrodynamics

Magnus Selhammar
Uppsala Astronomical Observatory, Uppsala University

Smooth Particle Hydrodynamics (SPH) is a Lagrangian method in which one does not need a grid to calculate the spatial derivatives. The method is often ideally suited for many astrophysical applications, because the particle description of the mass distribution also allows complex physics in three dimensions to be included in a natural way, such as conduction or magnetic fields. The most important astrophysical phenomenon, self gravitation, has been included in this code.

In SPH a particle has a finite spatial size. Particles move freely and interact hydrodynamically with each other if they lie within the domain of interaction. As the particles move in space, the nearest particles with which they interact may become more distant and other may come closer. Thus, the particles are allowed to change their interacting neighbors. However, SPH is not without disadvantages. The smallest spatial scale resolved is a few interparticle distances, which imposes restrictions on e.g. how shocks can be treated. In the past, models have often been restricted to a few thousand particles, but with the CM much more data-intensive models can be used. On the 8K CM200, models with 256K particles are possible.

With the CM much more data-intensive models can be used

The program will be used in numerical studies of the environment around young stellar objects. Work on inclusion of radiation transport in the method is in progress. Radiation transport is typically a non-local phenomenon and more complicated than self gravitation. The implementation will use complicated communication patterns, thus requiring more of the CM hardware and software.

5.7 Dynamics of Disordered Magnetic Systems

Jan-Olov Andersson
Department of Technology, Uppsala University

Disordered magnetic systems, such as spin glasses, random-field magnets and diluted antiferromagnets, are important model systems for an understanding of the physics of disorder and randomness in general. They are simpler to treat analytically than other related systems, such as structurally disordered materials. Nevertheless, the randomness makes an analytical path to the solution of the problem intractable. On the other hand, the simplicity in formulation makes the problem ideally suited for Monte Carlo (MC) simulations.

By providing access to theoretical concepts that are not experimentally measurable, and because of the possibility to closely mimic the experimental procedure in a simulation, MC simulation can help bridge the gap between theory and experiment. The

The problem has made it necessary to use parallel computers to reach conclusive results

complexity of the problem has made it necessary to use special-purpose machines and parallel computers to reach conclusive results. While the majority of all simulations have been aimed at studying the static and equilibrium properties of spin glasses, we have focused on the crossover from equilibrium to non equilibrium dynamics, an issue that has been studied extensively in experiments for the last ten years, but has been almost neglected in simulations. In my earlier studies using conventional workstations, I have been able to reproduce the qualitative features found in experiments. Still, to look at more quantitative aspects and to study the spin system at higher temperatures we needed more computer power.

So far, my work has been concentrated on validating and optimizing the program for the CM by trying to reproduce experimental results from temperature-cycling experiments.

I have also been working on a study of the nature of the phase space close to a local energy minimum. This work is done in cooperation with Paolo Sibani at Odense University, Denmark.

In the future I intend to look at the time-dependent AC susceptibility, the growth of correlations in a spin glass and the behavior close to the phase transition temperature.

5.8 Electrolytic Flow in 2D

Lars-Göran Sundström, Fredrik Wallgren
Department of Hydromechanics, KTH

Many electrochemical systems involve electrolyte flow between two parallel electrodes. The problem has been treated previously by others, but to our knowledge no one has taken the fully numerical approach, as in this project, to get the complete picture of the concentration gradients and electric potential field between the electrodes. The program was developed as a first step in treating the total problem where there is no forced convection, and the motion of the electrolyte is instead set up through the concentration differences.

This program was first developed for a serial computer, and then converted to a parallel code for the CM200. The two codes were compared with each other in terms of execution time, and we came to the conclusion that for our first implementation, the CM was slower than the serial Alliant FX80. The reason for this is that the algorithms we used were developed for serial computers, and we have not yet put any effort into making them efficient for parallel computers.

It should be noted, though, that in several cases this requires a totally different algorithm which forces the programmer to choose between parallel and serial computing at an early stage. The algorithm used to solve the non-linear equation system, for instance, was Gauss-Seidel´s point relaxation method. While this is reasonably efficient on a serial computer, it is not so well suited for parallelization.

The program was run with different Peclet numbers (affecting the boundary-layer thickness), and the result of $Pe = 100$ are shown in the color-graphs. The result agrees with previous results from simplified models based on assumptions of thin boundary-layers [Parrish and Newman, 1970], but gives the solution in the total domain instead of only adjacent to the electrodes. Full solutions were also found for very thick (interacting) boundary layers. (See Figure 2.7 on 18.)

5.9 Spatial Effects in Evolutionary Models

Kristian Lindgren
Institute of Physical Resource Theory, CTH
Mats Nordahl
Santa Fe Institute, New Mexico, USA

A simple model of a population of interacting individuals on a square lattice is presented. At each lattice point there is one individual interacting with its four neighbors, according to a variation of the Prisoner's Dilemma game. An individual has a genetic code that describes the strategy that it uses in the game. The most successful strategy in a local neighborhood (a lattice point with its four neighbors) will occupy the central lattice point in the next time-step (generation), and in this way better strategies may spread on the lattice. In the transition from one generation to the next (one time-step), mutations may alter the genetic code, leading to new strategies. In this way the whole system may evolve, and better strategies may appear.

The lattice is updated in parallel, which means that it can be described as a cellular automaton, although a probabilistic one due to the mutations. Therefore a CM is excellent for simulations of this model. A typical lattice size is 4096 points, which allows for one individual per processor.

In a previous paper [Lindgren, 1992b], a similar model was studied, in which the spatial dimension was not included. The present study aims at exploring spatial effects, e.g. , regions of different strategies and spatio-temporal chaos, that may evolve in artificial ecological systems. It is well known that the spatial degree of freedom may stabilize a system that otherwise would have been unstable. A preliminary report on this research is available in [Lindgren and Nordahl, 1993].

5.10 Simulations of Supersymmetry in Cahn-Hilliard-Cook Theory

William Klein
Boston University, Boston, USA
Lawrence Thomas
Theoretical Physics, KTH

Supersymmetry is an idea that has been around for a few years and caused a great deal of excitement in its early years. It is basically a theory that says that there is a an invariance in a system under the transformation of commuting to non-commuting variables. In condensed-matter physics, for example, supersymmetry representations have contributed to our understanding of the random-field model, the linear model and branched polymers. In high-energy physics it has yielded relations between bosons and fermions such that transformations are possible between the pairs.

W. Klein and G. G. Batrouni introduced a supersymmetric representation of the Cahn-Hilliard-Cook linear theory of spinodal decomposition and continuous ordering. In contrast to previous applications of supersymmetry in condensed matter physics, they related the Grassmann variables to physical objects.

In an attempt to take a step towards concretely proving these theories along with their mathematical formulations, we have done computer simulations that take advantage of percolation theory, with respect to a simple square short-range interaction Ising model system. First, a description of what we are seeking.

With respect to a graph of temperature versus density in a system made up of two constituents, there is a curve, called the spinodal, that represents the separation of the stable region from metastable and unstable regions. Above the curve, the two components are homogeneously and stably miscible in all proportions. Below the curve are the metastable and unstable regions. The metastable region is similar to the stable region in most respects, i.e. the mixture remains homogeneous. They differ in that the Gibbs free energy (at constant p and T) is higher for the homogeneous mixture than for a system formed by two co-existing phases. Further below, in the unstable region, the material separates into different phases.

In the past, computer simulations on various other machines have been done by several groups in order to display the phenomenon of spinodal decomposition. These experiments assume, a system at a very high temperature with equal amounts of two types of "atoms" distributed at random. A difference in the chemical potential or magnetic field is used in the direction from the top to the bottom of the system. The system is then quenched at $t = 0$ to a very low temperature, say $T = 0.5T_c$ (T_c defined as the point separating the stable and unstable points on the spinodal curve).

We find that the CM is well suited for this type of simulations. Taking advantage of its parallel processing, we have set up a very large Ising model random-spin system in order to simulate the particle-hole supersymmetry we are studying. We have been able to take full advantage of the parallel processing because we are able to work on every site within the system simultaneously. Consequently, simulation-times have been cut drastically. [Klein and Thomas, 1993]

6 Biocomputing

Scientific computing done by researchers in medicine and biology is beginning to form a new branch of science, often termed *biocomputing.* This new branch – driven by computing – is transforming many parts of medicine and biology from an informational science into a computational and analytical science. [Lander *et al.*, 1991] Almost since the start of PDC we have had people working on protein sequence matching. Currently, a protein sequence matching server is running on the CM. Researchers can connect to this server with a client program running on their local workstation and match a protein sequence against the database on the CM (Section 6.1). The other project comes from an area in biocomputing where large data sets are one of the obstacles, namely analysis of PET scan images (Section 6.2 on page 58).

6.1 Sequence Alignment on the CM

Gunnar von Heijne
CBT, Karolinska Institutet
Fredrik Hedman, Erik Wallin, Christian Wettergren
PDC, KTH

Currently, the amount of sequence-data is growing by a factor of two every three years

Proteins and nucleic acids, the building blocks of the genetic material, can be described as sequences of characters, with a twenty-letter alphabet for proteins and a four-letter alphabet for nucleic acids. Currently, the amount of collected sequence-data is growing by a factor of two every three years, and this rate will increase as the Human Genome Project (HUGO) 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

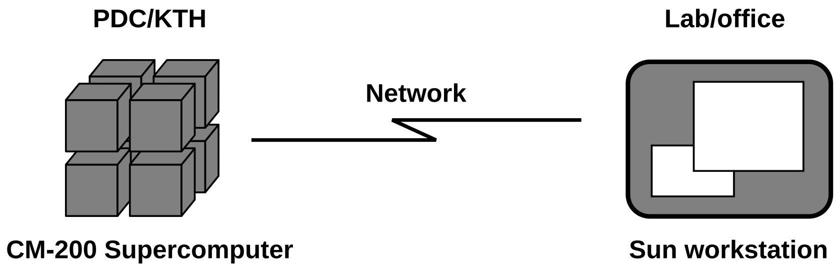

Figure 6.1. Bringing supercomputer power to the desktop. A client program running on the user's workstation takes care of all the communication with the server program which does the actual computation on the CM200. This application can easily be used by people who know very little about supercomputers. What they see is an application that could not even be run on their workstation before, and now it is within arm's reach.

hardware for parallel computing have led to vast increases in computational speed for certain classes of problems. Protein sequence data bank scanning is well suited to parallelization, and more sensitive comparison algorithms may be executed within acceptable times.

During 1991 we developed a program on the CM for searching these data banks. [Wallin *et al.*, 1993] It is based on the Needleman–Wunch algorithm, which is considered to be the most sensitive available. Unfortunately it is also the most computationally demanding, and it is therefore not very widely used for data bank searching. The algorithm is well suited for parallelization, and the fact that the databases are growing makes the parallel implementation more efficient every year. (see also Section 3.1)

Supercomputer power brought to the desktop

Not every researcher has a supercomputer in the lab. Therefore, to enable users to access the sequence databases through our programs, we have developed a client-server interface. This effectively brings supercomputer power to the desktop. The interface program was written during 1992, and we have recently begun testing and adding users. The search program is written in C*. During development we had tremendous help from Prism, the new debugging environment.

A search which would take several hours using a sequential implementation takes only a couple of minutes on the CM

The computations are done by a server program running on the CM. The client program, which runs on the user's workstation, takes care of communication with the supercomputer, prepares data to be sent to the server and presents results once the computation is finished. Typically, a search which would take several hours using a sequential implementation takes only a couple of minutes on the CM. At the moment this interactive service is available during daytime. It was made possible by several new facilities. Firstly, it was absolutely necessary for the CM to run in time-sharing mode. This is because the server program usually only runs for a few seconds to service the user, and then sits idle for several minutes. Furthermore, the purchase of a DataVault and the upgrade to larger memory was also very important.

Currently we are adding more features to the server and also refining the user interface. The next step is to add DNA sequences to the database.

6.2 Analysis of 3D Brain Data

Per Roland
Karolinska Institutet, Stockholm
Björn Levin
SANS, KTH

New brain-imaging methods, such as measurements of the regional cerebral metabolism or the regional cerebral blood flow in combination with Positron Emission Tomography (PET), sample data from more than 50,000 parts of the brain.

Because of the 3D structure and the large amount of data, data analysis by a traditional sequential computer would be cumbersome

We have performed a series of simulations to study the effects of noise on different detection schemes to be used in the analysis of PET images. By generating 100 or 1000 re-runs of the same experiment using distribution data from a group of persons in the control state, i.e. a state in which the brain is not engaged in a particular task, it was possible to generate empirical distributions of false positive activation events in the brain.

The simulations were done in several steps. First the CM was used to investigate the spatial 3D autocorrelation function in a sample of regional cerebral blood-flow pictures. This function was then used to generate a 3D filter that imposes the found autocorrelation in subsequent simulations. Finally, to get the desired accuracy in the estimations, tens of thousands of randomly generated images having the correct statistical properties were analyzed using the mentioned methods, at the same time as false positive events were recorded. In each such generated image around 200,000 values mimicking measurements of biochemical and physiological variables in the brain are generated simultaneously, making the task very well suited for analysis by parallel computing.

7 Applications in Chemistry

Computer simulations of chemical systems is a complement to theoretical and experimental chemistry, made possible by the rapid growth of computer capabilities. These simulations can be regarded as computer experiments at a molecular level, and they require large amounts of computing power measured both as raw CPU power and as volume of stored and transferred data. By simulating a sufficiently large system one can compare experimental results with simulation results, thereby simplifying the fundamentally important interaction between experiment and theory.

In many systems studied, it is crucial to be able to perform as large simulations as possible to ensure that the results are physically relevant. These large simulations are often naturally massively parallel and require both very large memories and fast calculations. Typical examples are classical and quantum-mechanical many-particle calculations of structural features and dynamic behavior. The present projects running on the CM are examples from molecular dynamics (Section 7.1), reaction dynamics (Section 7.2) and quantum chemistry (Section 7.3).

Large simulations are often naturally massively parallel.

7.1 Molecular Dynamics for Liquids

Fredrik Hedman
PDC, KTH
Aatto Laaksonen
Department of Physical Chemistry, Stockholm University

The advent of massively parallel computers holds great promise in that they provide the power to substantially improve the accuracy of molecular simulations and hence their predictive potential. As raw compute power, main memory and I/O capacity increase, the size of the modeled systems can be enlarged. These larger systems become more realistic by limiting possible numerical artifacts and by preventing a too early truncation of the long-range interactions. Both of these problems are caused by applying periodic boundary conditions in a too small simulation cell.

Massively parallel computers change the underlying assumptions about how an effective simulation method should be designed.

Massively parallel computers change the underlying assumptions about how an effective simulation method should be designed [Modi, 1988]. To use these computers effectively, a great deal of effort must be put into developing new methods and adapting old ones. This project is concerned with developing data-parallel methods which will enable us to study large systems of liquids which are modeled by both short-range interactions.

We use a method of molecular dynamics (MD) to study simple Lennard–Jones particles in a liquid phase. A suitable model is constructed by taking a box and filling it with particles to a desired density. The number of particles is chosen as a compromise between available computer resources and the reliability of the model. The more particles we have in the box, the better "macroscopic" description of the system. After the particles have been initialized to the desired temperature we solve the numerical equations of motion in order to find the new positions and velocities of all particles in the box. The numerical equations of motion are based on a discretization of Newton's second law. The procedure is repeated over and over again, thereby advancing the system in time.

The computationally heavy part in each time-step is the calculation of the force F_i on each particle i:

$$F_i = \sum_{i \neq j} F(r_{ij}) \tag{7.1}$$

where r_{ij} is the distance between particles i and j. In a Lennard–Jones fluid the forces are short-range, and this means that most of the terms in Eq. (7.1) will be close to zero. In fact, with $r_c =$ the cut-off radius and $r_{ij} \geq r_c$ one sets $F(r_{ij}) = 0$. Several methods have been devised to avoid calculating these nonproductive terms, such as Verlet neighbor lists and linked cells (LC) [Allen and Tildesley, 1987].

We have chosen to start from a parallel version of the LC algorithm which is often called the coarse-grained cell method.

In this method the simulation box is divided into a number of equally sized cells. Usually these cells are chosen to be cubic. For short-range forces, a suitable choice of the cell size (with side length L) limits the interaction neighborhood of a molecule to the first layer of surrounding cells. Even though this method avoids a substantial part of the nonproductive force calculations one can easily see that there will still be interactions calculated which turn out to be negligible. In fact, a particle will find all of its interactions in a volume $V_c = 4/3\pi r_c^3$, but the method will search for neighbors in the volume $V_L = 27L^3 \geq 27r_c^3$ and $V_c/V_L \approx 0.16$.

To avoid a larger fraction of the nonproductive calculations we have developed a geometric sorting procedure based on particle distances to subcell boundaries. Due to particle migration, the contents of the subcells need to be updated. This is done with a method based only on nearest-neighbor communications. Special "null-particles" are introduced, which act as buffers during periodic updates and allow for a globally uniform algorithm during the force calculations.

We have implemented our code in CM Fortran on the CM. Communication cost is around seven percent of the total CPU time. The overall speed for one million particles is approximately 5.9 μs per MD time-step and particle, and 5.5 μs for five million particles. Our results show that it is possible to develop very efficient programs for large-scale computer simulations of liquids using the data-parallel programming model. Implementation of a program to include long-range interactions is in progress. [Hedman and Laaksonen, 1992a, Hedman and Laaksonen, 1993b, Hedman and Laaksonen, 1992b, Hedman and Laaksonen, 1993a]

Our results show that it is possible to develop very efficient programs for large-scale computer simulations of liquids using the data-parallel programming model.

7.2 Reaction Dynamics

Anders Broo
Department of Physical Chemistry, CTH

Non-reactive three-atom collisions are studied by Monte Carlo simulations and RRKM-theory for the formation and decomposition of the reaction complex. The underlying experimental information is obtained from molecular beam experiments. A Fortran 77 program [Rynefors, 1982] has been ported to the CM at PDC. The transformation from Fortran 77 to CM Fortran was quite time-consuming, and the performance is still not very impressive. The difficulties of the transformation were mainly due to the intrinsic sequential nature of the original Fortran 77 program, and much still remains to be done to increase the speed.

7.3 Semi-empirical Quantum Chemistry

Anders Broo
Department of Physical Chemistry, CTH

A semi-empirical quantum chemistry program [Broo and Larsson, 1992] written in Fortran 77 has been ported to the CM. The calculation spend most of the time on diagonalization of a relatively large matrix and the formation of matrix elements in the configuration-interaction matrix.

The diagonalization is performed with subroutines from the CMSSL library. The formation of the elements of the configuration-interaction matrix requires a double loop over the dimension of the basis set (100–1000) for each matrix element. The structure of the program is thus well suited for running very efficiently on a massively parallel computer. Benchmark calculations are in progress.

8 Geophysics

Parallel computers have in recent years found many applications in the field of computational geophysics. These applications include 2D and 3D solutions of the elastic anisotropic wave equation by finite-differences, reverse time migration, modeling of fracture behavior of rocks, lattice-gas modeling of wave propagation and solution of the seismic travel-time inversion problem, using neural networks. In particular, access to GBytes of fast data storage, as well as calculation speeds on the order of GFlop/s, allows the inversion of real seismic waveforms, given the data volume of a seismic survey.

Waveform inversion of seismic data is a highly compute-intensive process. The high degree of parallelism inherent in the problem allows the modeling algorithm to benefit from enhanced computational speed due to simultaneous operations on different data. (Section 8.1)

Numerous rocks and materials exhibit anisotropic behavior. The first step in understanding the effects of anisotropy in the seismic data we collect is to model seismic wave propagation in anisotropic media. (Section 8.2)

8.1 Seismic Waveform Inversion

Jonas Lindgren
Solid Earth Physics, Uppsala University

The seismic reflection method is presently one of the most powerful geophysical remote-sensing methods, yielding high-resolution information concerning the elastic properties of the Earth on a crustal or lithospheric scale. The theoretical foundation of the method is the study of wave propagation in an elastic medium. In a seismic survey, the response of the Earth to an artificial source, such as an explosion, is used to retrieve information about the medium through which the elastic energy has propagated. Recordings made at, or near, the surface by vibration-sensitive geophones on land, or pressure-sensitive hydrophones at sea, provide information that is used to describe how the energy has been transmitted and reflected from discontinuities in the elastic properties of the rock.

Seismic inversion is the process of using these recordings to yield quantitative maps of the properties of the subsurface. The inversion process makes use of the wave equation to predict the characteristics of the subsurface that, given the excitation in the form of the seismic source, yield the observed wavefield.

Figure 8.1. Seismic modeling/inversion. Whereas the modeling, or the *forward*, problem consists of the calculation of seismograms (right) resulting from a seismic experiment in a known geology, the *inverse* problem is the retrieval of the geological model (left) from the seismograms.

This figure is only available in the paper version

Inversion can be thought of as the inverse process of modeling, see Figure 8.1. Whereas the modeling, or the *forward*, problem consists of the extrapolation in time or depth of the wavefield resulting from a seismic source function and some initial and boundary conditions through the use of a wave equation, the *inverse* problem is the retrieval of the properties of the medium, given the observations of the wavefield at some discrete locations, generally at the surface.

The formulation of the seismic inverse problem adopted here relies on the successive updating of an initial Earth model until observed and modeled seismograms match. Fundamental to the method is the ability to accurately model the propagation of seismic waves in a general elastic medium. The backbone of the algorithm consists of an elastic high-order finite-difference scheme.

The modeling/inversion program package was originally developed for a Convex C-1 at Institut de Physique du Globe in Paris. These codes have been further developed and adapted to the CM, using the CM-sites at Centre de Calcul in Paris and PDC in Stockholm. (See Figure 2.9 on page 20.)

The data-parallel concept provides a programming environment which simplifies the task of expressing operations that apply to many data simultaneously. To the first approximation, the programmer does not need to be concerned with the actual hardware implementation of the code; however, a knowledge of the machine architecture is crucial in the development of efficient codes of a more complex nature.

Knowledge of the machine architecture is crucial in the development of efficient codes

When attempting to perform seismic modeling, a number of operations have to be carefully coded, such as the input of the source, boundary conditions and the recording of the field at the receiver locations. On a serial computer the cost of these operations is small, or even negligible, compared to the cost of modeling. On the CM the cost of these operations seriously degrades the overall performance unless they are carefully coded. As an example, if the sampling of the wavefield were coded in the same way as for serial execution, general communication would have to be performed within the time-loop. Such an operation could easily double the overall execution time of the program.

An efficient solution of the sampling problem requires many lines of code. However, although several other examples exist where elaborate programming is necessary to avoid unacceptable execution times, the recoding into CM Fortran is generally a step toward simplicity rather than complexity.

Recoding into CM Fortran is generally a step toward simplicity rather than complexity

In order to be able to perform waveform inversion of real seismic waveform data, a seismic inversion package (15,000 lines of code) has been converted into CM Fortran. The experience shows that software optimized for a serial computer is not likely to perform well on a parallel machine like the CM without extensive reorganization. Due to the facilities of a parallel language, the resulting code can be made clearer and faster than the serial version.

The introduction of massively parallel computers in geoscience not only enables things to be done faster than before, but it allows problems to be attacked which, due to various computational considerations, were previously computationally impossible. Although parallel computing in geoscience is still at an initial stage, the access to computers like the CM is an important step towards the goal of the complete seismic waveform inversion of real 3D data sets. [Lindgren, 1992a]

8.2 Anisotropic Elastic Wave Propagation in 2D

Christopher Juhlin
Solid Earth Physics, Uppsala University

Numerous rocks and other materials are anisotropic, that is their physical properties are dependent upon the direction on which measurements on the material are made. Consequently, waves propagating through the rocks have a velocity which is a function of the azimuth and the dip angle of propagation. Discontinuities in the physical properties of the rocks reflect elastic waves, and it is the motion of these reflected waves recorded on the surface or in bore-holes that provides us with information about the subsurface.

Although it has been realized for some time that many rocks are anisotropic and that even a sequence of thin isotropic layers makes up a rock which is effectively anisotropic, anisotropy is not generally taken into account in seismic-data processing and modeling. However, this situation is changing. This is partly due to the advent of large fast computers, which allows anisotropy to be dealt with comprehensively, but also to the fact that if it is not taken into account in some, areas the resulting seismic images do not represent the geological subsurface adequately.

The situation is changing due to the advent of large fast computers

The first step in understanding the effects of anisotropy in the seismic data we collect is to model seismic wave propagation in anisotropic media. A suitable technique for modeling is the finite-difference method. In the past, finite-difference methods have been widely used to model seismic wave propagation in both acoustic and elastic isotropic media. Most of this modeling has been for the 2D problem, but there have also been 3D applications. In the majority of these cases a single second-order hyperbolic equation is solved for the acoustic case or two second-order hyperbolic equations for the 2D elastic case with pressure or displacement as the variable, respectively. However, in relatively recent applications to 2D elastic isotropic media [Levander, 1988] the wave equation, as a second order hyperbolic equation, is not solved directly. Instead, the problem is formulated in the form of a first-order hyperbolic system where the variables are stresses and velocities, rather than displacements.

The displacement formulation has a number of drawbacks. [Levander, 1988] The most serious objection to the second-order displacement approach is the complicated computer code which is required to set up the mixed spatial derivatives. This objection becomes progressively more serious as more complicated anisotropy is to be modeled.

In my work, thus far, I have assumed hexagonal anisotropy, also called transverse isotropy, to be present in the rocks. This is the simplest form of anisotropy, with one axis of rotational symmetry. Although simple, it is commonly the dominant one present in real rocks. Recently, in [Igel *et al.*, 1991] results from implementing a finite-difference scheme, fourth-order in time and eighth-order in space, to the 3D anisotropic problem has been presented. The scheme I am working with is similar, but applied only to 2D problems. With the present capacity of the CM200 at PDC it is possible to model relevant and realistic 2D problems, but not 3D ones. Therefore, I have concentrated on the 2D case for the present time.

The numerical code has been implemented in Fortran 77 and CM Fortran code. On a SPARC 10 running at 10.7 MFlop/s, the CPU-time for a 404×404 grid and 2000 time-steps was about 90 minutes. On an 8K CM200, the same problem requires about 5 minutes. The parallel code has not yet been optimized. Significant improvement can be expected with optimization.

5 minutes instead of 90 minutes

Future 3D modeling will require considerable amounts of memory and there will be a tradeoff between run-time and the size of the problem that can be solved. Since my project also involves physical modeling, i.e. carrying out seismic experiments in the laboratory with a scale model built from epoxies and similar materials, I will eventually have to deal with the 3D problem to be able to compare the results with numerical modeling. For example, a $256 \times 256 \times 256$ cube will require on the order of 2 GByte of memory. This is probably the minimum size that can be used to model realistic seismic problems. Extrapolating the current run-time for the present code to the 3D problem implies that a single model will take about 12 hours. Optimization will reduce this time, but it obvious that a more powerful and larger memory computer will be necessary for the full 3D problem.

A more powerful and larger-memory computer will be necessary for the full 3D problem

9 Numerical Analysis

Massively parallel machines present a large challenge to numerical analysts, since many old and established serial algorithms and methods turn out to perform very poorly on these machines. However, those algorithms that are well adapted to these machines are worth optimizing. This has been done in the FFT project (Section 9.1). It is also interesting to investigate how well some algorithms can be implemented directly in Fortran 90 (Section 9.4). New opportunities in using new parallel algorithms have been explored in the areas of: Legendre transforms (Section 9.2), parallelizable preconditioning methods (Section 9.3) and optimization (Section 9.5).

9.1 Development of Efficient FFT Library Routines

Lars Malinowsky
NADA, KTH

Has led to highly optimized FFT polyalgorithms, now included in CMSSL

Spectral methods give very high accuracy for problems with smooth solutions, and they have for some time been the methods of choice for computing unsteady viscous flows with resolution of all scales, i.e. the direct simulation of turbulent flows. The methods rely on the Fast Fourier Transform (FFT) for which the Connection Machine Scientific Subroutine Library (CMSSL) has efficient routines using complex arithmetic. The first versions were inefficient in terms of memory for real data, and a real-to-complex code which avoided this waste was first developed. The continued FFT code development has led to highly optimized FFT polyalgorithms, now included in CMSSL. The codes have options for many different combinations of hardware and data-layout. [TMC, 1992]

9.2 Fast Legendre Transform

Erik Aurell
Department of Mathematics, Stockholm University

A class of nonlinear partial differential equations that have been widely studied are the hyperbolic conservation laws. They express the conservation of a certain quantity, say momentum, in the continuous motion of a macroscopic system, e.g. a gas. The resulting equations must balance the transport of the conserved quantity against a force, thus a gas resists compression because the pressure increases with density. The equations of gas dynamics are of this type. In some situations the equations of fluid dynamics and plasma physics are also of this type.

The dominant physical effect in hyperbolic conservation laws is the formation of shock waves, where the solutions are discontinuous. On the other hand, in a real physical macroscopic system, there is always dissipation, which is generally well modeled by the Laplace operator multiplied by an appropriate transport coefficient (for instance ν, the viscosity in compressible hydrodynamics). When the solutions of the hyperbolic conservation law develop strong gradients, the dissipative term, which up to then can be neglected, becomes relevant. A careful analysis then shows that the shock waves are not completely sharp, but have a width proportional to the transport coefficient.

Since a transport coefficient, for dimensional reasons, is proportional to the mean velocity times the mean free path, we find that the width of a shock wave at ordinary temperatures is always much larger than the scales where the macroscopic laws of motion begin to hold. This means that the discontinuities in the solutions to the hyperbolic conservation laws do not contradict the assumptions made when one derives these laws from the microscopic equations of motion. This also means that the solutions of the hyperbolic conservation laws *with discontinuities* make mathematical sense: the limit when transport coefficient tends to zero selects, among many possible weak solutions, the one that is physically relevant [Lax, 1973].

It is a remarkable fact that a large class of hyperbolic conservation laws can be integrated as Legendre Transforms of the initial conditions [Lax, 1973, Burgers, 1974]. In words: it is not necessary to integrate numerically the partial differential equation up to some time T; one can transform directly from the initial conditions, and find the solution at time T, without computing the solutions at intermediate times. There exists a version of the Legendre Transform which requires a number of operations of the order $N \log N$, if the initial conditions are realized on N discrete points in one dimension [Burgers, 1974]. By analogy with the Fast

Fourier Transform, this transformation is called a "Fast Legendre Transform", and it is extremely useful if one wants to study the behavior on large scales after long times [She *et al.*, 1992, Aurell, 1992a].

With the aim of continuing the study on one of the simpler conservation laws, the one-dimensional Burgers equation [She *et al.*, 1992, Aurell, 1992a], a parallel version of the Fast Legendre Transform was implemented on the CM200 at PDC [Aurell, 1992b].

On the CM the data are naturally organized in two different ways: either as initial conditions (called Lagrangian coordinates in hydrodynamics), or as the positions at time T (called Eulerian coordinates in hydrodynamics). One cannot avoid transmitting data between the two sets of coordinates. In between, one performs maximizations over partitions of the initial (Lagrangian) coordinates. The delimiters of this partitioning are determined by the initial conditions, so they are only known at run-time. Fortunately there is a built-in function on the CM, the "segmented scan" operation, which is precisely suited for this purpose.

At present the algorithm needs at most $2 \log N$ general communications operations, and $8 \log N$ segmented scan operations. If each of these operations each take of the order of $\log N$ operations, the integration would theoretically take about $10(\log N)^2$ time-steps, as long as N is less than the number of physical processors. In reality, one is of course interested in as large arrays as possible, that is with high virtual-processor ratios.

The test runs are encouraging, but so far no systematic measurements have been made of the execution speed actually achieved. Probably this is not the most important point: the real advantage in implementing the Fast Legendre Transform on a parallel architecture is that one can use the large memory and lose relatively little in speed compared to a serial implementation.

The advantage in implementing the Fast Legendre Transform on a parallel architecture is that one can make good use of the large memory

The program was written in C, with function calls to the assembly-level language PARIS. In retrospect, the results would not have been very different if the code had been written in C*. There would probably be more to gain from rewriting the code in CM Fortran, since that language uses the slice-wise model, a different machine model which is not supported by C*.

9.3 Completely Parallelizable Preconditioning Methods

Ivar Gustafsson, Gunhild Lindskog
Department of Computer Sciences, CTH

We present a class of parallelizable preconditioned iterative methods for the solution of finite-element linear systems of equations. The iterative method may for instance be the conjugate-gradient method. We consider elliptic second-order orthotropic boundary-value problems. The matrix in the resulting linear system of equations is symmetric and positive definite.

Most of the ideas presented in the field of parallel iterative methods are based on block-wise parallelization suitable for parallel computers with a small number of processors. In our present work we consider completely parallelizable preconditioning methods, that is, methods in which the construction of the preconditioning matrix as well as the solution of the preconditioning system can be done in parallel over the total number of mesh-nodes. Hence, the methods are suitable for implementation on massively parallel computers such as the CM.

Completely parallelizable suitable for implementation on massively parallel computers such as the CM

Our main idea for the construction of the preconditioner is based on the calculation of approximative inverses of the Symmetric Successive Over-Relaxation (SSOR) factorization of the matrix of the linear system. The solution of the preconditioning system is then performed by matrix-vector multiplications, which makes it possible to carry out all operations involved in parallel over the total number of unknowns. The rate of convergence of the method is increased by making the approximations of the inverses more accurate. However, this also increases the computational complexity in each iteration.

We have chosen two test problems: an isotropic problem and a problem with strongly varying orthotropy. The calculation times for the iterations, when the residual error in 2-norm is reduced by a factor of 10^{-4}, show an improvement of about 25 percent compared to diagonal scaling. This is for a problem of size 512^2. The improvement is somewhat less for smaller problems.

9.4 Matrix Computations on the CM2

Göran Svensson, Lars Eldén
Department of Mathematics, LiTH

Fortran 90 is a standardized language available for many different platforms, and programs written in this language will be easy to port. However, it has not been clear if it is possible to program massively parallel machines like the CM on a sufficiently low level with Fortran 90, i.e. if assignment of data to processors and load balancing can be made efficiently. The purpose of this project is to investigate if certain matrix computations can be executed efficiently on the CM when programmed in CM Fortran.

Investigate if certain matrix computations can be executed efficiently on the CM when programmed in CM Fortran

In particular we have studied the implementation of an algorithm for QR decomposition of a matrix based on Householder transformations. The QR decomposition is used in numerous applications, e.g. when solving linear least squares problems. By applying a sequence of Householder transformations, the matrix is step by step reduced to upper triangular form, which is the r matrix in the QR decomposition. This reduction starts at the top left corner of the matrix and proceeds down along the diagonal. If the Householder transformations are multiplied together, then the q matrix in the decomposition is obtained.

Loosely speaking, an algorithm for a parallel computer is said to have a good load balance if most of the processors are busy doing useful work most of the time. For the computation of matrix decompositions, load balancing is usually achieved by performing the reduction in the standard way, and assigning matrix elements to processors in a clever way, or, alternatively, by performing the reduction in a clever way and assigning matrix elements to processors in the standard way.

The standard way of assigning matrix elements to processors on the CM assigns each matrix element to a virtual-processor. Virtual processors are then mapped to physical processor in such a way that a block of neighboring matrix elements is stored in the same physical processor.

It is obvious that load balancing cannot be done by using the standard virtual-processor assignment, since after some steps in the Householder procedure, more and more of the processors would become idle. We avoid this problem by storing the matrix as a blocked matrix in a rank-4 array, with two dimensions parallel and two serial. This corresponds to a block-cyclic assignment of matrix elements to processors, and using this procedure, we can design a block-matrix algorithm, where the computations are parallel within the blocks (to a large extent matrix-vector operations), and serial over the blocks. Further, this algorithm can be coded completely in CM Fortran, with data-parallel operations within the blocks. Since the algorithm is serial over the blocks, all processors will be active almost all the time, and the algorithm is load balanced. This kind of parallelization also works well on MIMD parallel computers of hypercube type, e.g. *i*PSC/2.

Our results indicate that at present matrix algorithms written in CM Fortran cannot compete in speed with low-level codes, such as the ones in CMSSL. However, using knowledge about the architecture and the compiler, it is possible to get much higher efficiency, than that which would have been obtained by straight-forward translation of a serial algorithm into CM Fortran.

9.5 The Traffic Assignment Problem

Athanasios Migdalas, Olof Damberg, Saied Ghannadan
Department of Optimization, LiTH

Most large cities face serious urban transportation problems because of the increasing number of vehicles. In order to analyze the present situation and to predict future transportation needs, mathematical programming and modelling are suitable tools.

For a given network, the traffic assignment problem involves a travel demand associated with each origin and destination pair of nodes, and a travel cost function for each link. The travel costs may depend on the load pattern, due to the congestion effect, while the different travel demands may depend upon the travel costs associated with the origin and destination pairs. The problem is to find the traffic pattern which possesses the property that, once established, no network user can decrease the personal travel cost by altering travel decisions.

Our intention is to implement and test different strategies for the solution of the traffic assignment problem on real-world data from large cities, e.g. Barcelona and Stockholm. These real-world problems easily become very large; many thousands of links and nodes together with many thousands of origin and destination nodes are generally present. It is a nonlinear problem which often has to be solved repeatedly or, in the case of route guidance systems, instantaneously. Currently, on a workstation, more than eight minutes of computation is needed to solve just one instance of the problem in a network with 1000 nodes, 2500 links and 8000 origin and destination pairs. Suitable optimization algorithms that can be implemented on massively parallel machines must be developed.

Suitable optimization algorithms that can be implemented on massively parallel machines must be developed

Currently we are working with the so-called partial linearization technique and the regularized Frank–Wolfe algorithm [Larsson and Migdalas, 1990, Migdalas, 1990]. The idea is to decompose the problem into similar subproblems, by origin and destination pairs, and to solve the subproblems in parallel by algorithms for single-commodity network flows with quadratic link costs. In this setting we define two problems to be similar if they occupy the same amount of memory and if the same algorithm, applied separately to each of them, solves the problems. This similarity property allows the subproblems to share the same virtual-processor set and to be solved in parallel by the same algorithm.

So far we have implemented algorithms for the quadratic subproblem. We have tried two different methods. A partial linearization method on the Lagrangian dual and a dual active-set conjugate-gradient method. The implementations are being tuned for the CM.

10 Computer Science

Many regarded the CM as having a rather strange computer architecture and predicted that the most of its users would come from predominantly computer science related research areas and not from other research fields. This prediction has turned out not to be true; the bulk of the users at the center come from areas that are not computer science related. In fact, rather few of the large number of projects at the center are strictly computer science related.

Three different areas: data-parallel programming languages, parallel logic programming, and architecture simulations

The three current computer science related projects represent three different areas: data-parallel programming languages, parallel logic programming, and architecture simulations. In the language project (Section 10.1) a new language model suitable for describing data-parallel computations is presented and this model can act as a guideline for implementing new data-parallel programming languages; the logic project (Section 10.2) presents possible extensions to the logic programming language Prolog, and also their implementation; and finally the architecture project (Section 10.3) presents some aspects of low-level process synchronization primitives of interconnection networks, which are studied for a future hardware realization.

10.1 Data-Parallel Functional Programming Languages

Per Hammarlund
SANS, NADA, KTH
Björn Lisper
IT, KTH

Data-parallel programming, a powerful abstract programming paradigm

Data-parallel programming is becoming an increasingly important tool for exploiting parallelism in data-intensive applications, especially on SIMD and vector computers. Many algorithms appearing in such applications can be succinctly expressed in data-parallel languages: this indicates that data-parallel programming can be a powerful abstract programming paradigm rather than just a high-level syntax for explicit programming of SIMD computers. The data-parallel languages in practical use today are, however, exponents of exactly the latter point of view: even though they incorporate some elements of abstraction, their semantics is in every case to some extent based on a SIMD execution model. Therefore it is hard to use these languages to express algorithms in the problem domain in an abstract, machine-independent way. This is likely to make programming in these languages more error-prone and programs less portable than if they had been designed with more clean-cut abstract semantics.

We have analyzed and evaluated existing data-parallel programming languages and parallel execution techniques [Hammarlund and Lisper, 1992]. Existing data-parallel programming languages are mainly extensions of existing sequential languages, e.g. C* and *LISP are extensions of ANSI C and Common Lisp respectively [TMC, 1991a, TMC, 1991b]. This makes them very specific to the target machine hardware architecture and therefore difficult to port. In order to capture the more machine-independent aspects of data-parallel programming we have made mathematical definitions of some data-parallel primitives. These can be used to guide the design of data-parallel languages with a higher level of abstraction [Hammarlund and Lisper, 1993]. The key idea is to view data-parallel entities as tabulated functions, where the tables are stored in a distributed fashion. Operations on data-parallel entities are then simply operations on functions, just as operations in pure functional languages. Thus, from a programming perspective there is no difference between the function that generates a data-parallel entity and the data parallel entity as such. This implies that transformation techniques for functions can be applied to data-parallel entities. An interesting observation is that traditional data structures, like lists and arrays, are also covered by our definitions. This illustrates the level of abstraction achieved.

An especially interesting possibility is to integrate data-parallel and lazy higher order functional languages

An especially interesting possibility is to integrate data-parallel and lazy higher order functional languages. "Lazy data-fields", i.e. data-parallel entities whose entries are computed on demand, are the result. With such fields, many data parallel algorithms can be succinctly expressed. This is especially true when the domain of computation is irregular and varies strongly with the input. We thus believe that such languages are eminently suitable as specification languages for data-parallel algorithms.

10.2 Massively Parallel Logic Computation

Jonas Barklund, Henrik Arro, Johan Bevemyr
Computing Science Department, Uppsala University

Unlike most other projects carried out on PDCs CM, our work is research within parallel computation *itself*, rather than on *using* parallel computation for doing research in an application domain.

Advocates of logic programming languages have always claimed that these languages are good for computation on parallel computers, because their semantics is not based on a sequence of changes to a store. Still, imperative languages, such as Fortran 77, have so far been more successful for parallel computation than logic programming languages, even taking into account the results of the Japanese project for Fifth Generation Computer Systems.

The reason for this is that the Single Program Multiple Data (SPMD) model of computation has been successfully applied in these languages, by exploiting parallelism when running definite iterations (so-called *for* or *do* loops). Logic programming languages have mainly attempted to implement more general methods for achieving parallelism; so far these methods have unfortunately had quite significant overheads.

Computer programming languages based on logic usually have recursion as their only means for repetition. Theoretically this is sufficient, and in practice it often works fine. However, for essentially the same reasons that parallelizing so-called *while* loops is difficult, recursion is not easy to run in parallel.

We have therefore turned our attention to another construct for repetition in logic, namely quantification. In fact, we have restricted ourselves to bounded quantifications, an example of which is the expression

$$\forall i : 0 \leq i < n \rightarrow a[i] = b[i] + c[i].$$

That a quantification is bounded means that we know *a priori* that we only need to evaluate the body, $a[i] = b[i] + c[i]$ for a finite number of values of the bound variable i, here the values 0 to $n-1$. Seen as a truth-valued statement in a programming language, this expression says that the first n elements of the vector a are the sums of the corresponding elements of the vectors b and c, provided that they are all vectors whose indices go from 0 and upwards. This clearly corresponds to a *do* loop in Fortran 77.

We have designed an extension of the logic programming language Prolog that contains bounded quantifications. For example, a goal can be a universal bounded quantification $all(\rho)$ where ρ is a *range formula* that restricts some locally scoped variable ι to a finite number of values, and β is a goal, presumably containing ι. A goal can also be an arithmetic bounded quantification $\kappa(\rho, \beta, \tau)$ where κ is *sum* or *product* (or any other supported quantifier), ρ is as above, β is an arithmetic expression and τ is a variable, which is unified with the sum or product of the values of β. For example, $sum(I\ index\_of\, A, A[I]*A[I],\ R)$ unifies R with the sum of the square of every element of the array A.

An implementation of a data-parallel version of Prolog

Our work on PDCs CM has been to add this construct to an implementation of the logic programming language Prolog that has previously been developed at our department [Bevemyr, 1992]. This gives us a data-parallel version of Prolog where ordinary Prolog expressions are run sequentially, but bounded quantifications are run on the parallel processors of the Connection Machine.

The implementation is based on a compiler that translates the Prolog programs to sequences of instructions for an abstract machine [Warren, 1983]; the abstract machine is then realized by an emulator. This sequential emulator was written in C; therefore we could extend the machine with parallel instructions for bounded quantifications by adding a little data-parallel C* code to the existing C code. This was a relatively modest effort in terms of programmer time. We must conclude that the idea of having a data-parallel language as a conservative extension of a sequential programming language has worked very well.

More work is certainly needed before we can tell whether bounded quantification is a good extension of logic programming languages, but our experience so far is positive. Not only do our example programs run well on the Connection Machine, but bounded quantifications are perceived by many as more natural than recursion for expressing common forms of repetition. We hope that bounded quantifications will be useful in expanding the application areas of logic programming to include numeric computations [Barklund and Millroth, 1992, Barklund and Bevemyr, 1993, Arro *et al.*, 1993].

10.3 Barrier Synchronization for Multicomputers

Abdel-Halim Smai, Lars-Erik Thorelli
IT, KTH

Barrier synchronization is an important mechanism for coordinating parallel processes. In shared-memory multiprocessors common memory space is used to implement barrier synchronization. In a distributed memory parallel machine, however, synchronization is accomplished by passing messages between processors.

In most existing multicomputers, wormhole routing is adopted to support the underlying interprocessor communication. Apart from deadlock, a notable source of problem with this technique is congestion. Because a packet is not removed from the network when it is blocked, it can rapidly lead to more blocked packets and therefore to more congestion, especially with large message lengths and network sizes. This situation can be difficult to manage, and system performance can be significantly affected. occurs with non-uniform traffic patterns, for instance in the realization of software barriers. In particular, this occurs with non-uniform traffic patterns, for instance in the realization of software barriers. There are readily apparent drawbacks in such a case: not only is the delay for a barrier operation increased but it can also affect the communication delay of the other types of messages in the network.

A physical communication channel may be split into multiple virtual channels. A virtual channel is a logical channel which has its own buffer, data- and control-paths. A network can thus provide better node connectivity, and multiple paths and routes for packet transmission are made available. Our first goal is to test and compare different scheduling methods for virtual channels in order to optimize the delay for barrier synchronization operations and to limit the network congestion.

This work is part of the EDA (Extended Dataflow Architecture) project at the IT. In this study, we are considering some aspects of interconnection networks for a future hardware realization of an architecture supporting the EDA execution models.

We are using C* as our programming language. Its parallel features suit quite well the natural parallelism present in the model being simulated. Performance is, however, still a certain handicap. A few hours' execution time for a simulation run is common.

The parallel features of C suit quite well the natural parallelism present in the model being simulated*

Glossary

CFD, Computational Fluid Dynamics CFD 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.

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

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.

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 A term describing 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 Corporation and the Paragon from Intel.

SHPCNet, The Swedish High-Performance Computing Network The intention of the project is to connect the 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.

SIMD, Single Instruction Multiple Data Computer The term 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.

Bibliography

[Allen and Tildesley, 1987] Allen, M. and Tildesley, D. (1987). *Computer Simulation of Liquids.* Clarendon Press, Oxford. 60

[Argos *et al.*, 1991] Argos, P., Vingron, M., and Vogt, G. (1991). Protein sequence comparison: methods and significance. *Protein Engineering* **4**: 375–383. 56

[Arro *et al.*, 1993] Arro, H., Barklund, J., and Bevemyr, J., (1993). Parallel Bounded Quantifications—Preliminary Results. (To appear). 78

[Aurell, 1992a] Aurell, E. (1992a). Burgers' Equation with Scaling Initial Data. In: *PDC Center for Parallel Computers Progress Report 1990-1991.* Parallelldatorcentrum. 70, 70

[Aurell, 1992b] Aurell, E. (1992b). Program FLT.C. (Code commented and documented in LaTeX.). 70

[Aurell *et al.*, 1992] Aurell, E., Frick, P., and Shaidurov, V. (1992). Hierarchical Tree-model of 2D-Turbulence. Technical report, Center for Parallel Computers, Royal Institute of Technology, S–100 44 Stockholm, Sweden. 48, 49, 50

[Barklund and Bevemyr, 1993] Barklund, J. and Bevemyr, J. (1993). Prolog with Arrays and Bounded Quantifications. In: *Proceedings of the 4th Intl. Conf. on Logic Programming and Automated Reasoning*, edited by V. Andrei, LNCS, Berlin. Springer-Verlag. 78

[Barklund and Millroth, 1992] Barklund, J. and Millroth, H. (1992). Providing Concurrency and Iteration in Logic Programming through Bounded Quantifications. In: *Proc. Intl. Conf. on Fifth Generation Computer Systems 1992*, edited by H. Tanaka, Tokyo. Ohmsha, p. 817–824. 78

[Benzi *et al.*, 1990] Benzi, R., Paladin, G., and Vulpiani, A. (1990). Power spectra in two-dimensional turbulence. *Phys. Rev. A* **42**: 3654–3656. 49

[Berglund and Erlingsson, 1992] Berglund, M. and Erlingsson, S. (1992). Simulation of ground vibrations on a massively data-parallel computer. In: *Numerical Methods in Engineering '92*, edited by C. Hirsch, O. Zienkiewicz, and E. Oñate. Elsevier Science Publishers B.V., 1992. 47

[Bevemyr, 1992] Bevemyr, J. (1992). The Luther WAM Emulator. UPMAIL Technical Report 72, Computing Science Department, Uppsala University. 78

[Bomholt *et al.*, 1992] Bomholt, L., Choquet, R., and Leyland, P. (1992). Finite Element Solvers Using Data-Parallel Programming. In: *Numerical Methods in Engineering '92*, edited by C. Hirsch, O. Zienkiewicz, and E. Oñate. Elsevier Science Publishers B.V., 1992, p. 329–334. 27

[Bomholt and Leyland, 1992] Bomholt, L. and Leyland, P. (1992). Data-Parallelism for Finite Element Codes Using Unstructured Meshes. *EPFL Supercomputing Review* **4**: 13–17. 27

[Broo and Larsson, 1992] Broo, A. and Larsson, S. (1992). Ab initio and semi-empirical studies of electron transfer and spectra of binuclear complexes with organic bridges. *Chem. Phys.* **161**: 363. 62

[Burgers, 1974] Burgers, J. (1974). *The Nonlinear Diffusion Equation.* D. Reidl Publ Co. 69, 69

[Clayton and Engquist, 1977] Clayton, R. and Engquist, B. (1977). Absorbing boundary conditions for acoustic and elastic wave equations. *Bull. Seism. Soc. Am.* **67**: 1529–1540. 47

[Coulson *et al.*, 1987] 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. 56

[Durand and El Dabaghi, 1991] Durand, M. and El Dabaghi, F. (eds.). (1991). *Concurrent Evaluation of Boundary Conditions for the Euler equations.* High Performance Computing II, Elsevier Science Publishers B. V., 1991. 40

[Ekeberg *et al.*, 1993] Ekeberg, Ö., Hammarlund, P., Levin, B., and Lansner, A. (1993). SWIM—A Simulation Environment for Realistic Neural Network Modeling. In: *Neural Network Simulation Environments*, edited by J. Skrzypek. Boston, MA: Kluwer. 24

[Ekeberg *et al.*, 1990] Ekeberg, Ö., Stensmo, M., and Lansner, A. (1990). SWIM—A Simulator for Real Neural Networks. Tech. Rep. TRITA-NA-P9014, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 24

[Ekeberg *et al.*, 1991] Ekeberg, Ö., Wallén, P., Lansner, A., Tråvén, H., Brodin, L., and Grillner, S. (1991). A Computer based Model for Realistic Simulations of Neural Networks. I: The Single Neuron and Synaptic Interaction. *Biol. Cybernetics* **65**: 81–90. 25

[Engquist and Sjögreen, 1989] Engquist, B. and Sjögreen, B. (1989). Numerical Approximation of Hyperbolic Conservation Law with Stiff Terms. CAM 89-07, UCLA. 38

[Engquist and Sjögreen, 1991] ENGQUIST, B. AND SJÖGREEN, B. (1991). Robust Difference Approximations of Stiff Inviscid Detonation Waves. CAM 91-03, UCLA. 38

[Eriksson *et al.*, 1992] ERIKSSON, K., HANSBO, P., AND JOHNSON, C. (1992). Parallelization of numerical methods for the neutron transport equation. Technical report, Department of Mathematics, Chalmers Univeristy of Technology. 35

[Erlingsson and Berglund, 1992] ERLINGSSON, S. AND BERGLUND, M., (1992). 3-D Simulation of a live load in Nya Ullevi stadium. (To appear in proceedings from XIII International Conference on Soil Mechanics and Foundation Engineering, New Delhi, 1994). 47

[Fransén and Lansner, 1990] FRANSÉN, E. AND LANSNER, A. (1990). Modelling Hebbian Cell Assemblies Comprised of Cortical Neurons. In: *Proc. Open Network Conference on Neural Mechanisms of Learning and Memory*, London. p. 4–9. 25

[Fransén *et al.*, 1993] FRANSÉN, E., LANSNER, A., AND LILJENSTRÖM, H. (1993). A Model of Cortical Associative Memory based on Hebbian Cell Assemblies. In: *Computations and Neural Systems 92*, edited by F. M. Eeckman and J. M. Bower, Boston, MA. Kluwer. 25

[Hammarlund and Lansner, 1992] HAMMARLUND, P. AND LANSNER, A. (1992). Implementations of Very Large Recurrent ANNs on Massively Parallel SIMD Computers. In: *Proceedings of the 1992 International Conferance on Artificial Neural Networks*, edited by I. Aleksander and J. Taylor, Amsterdan. ICANN-92, North-Holland, p. 1287–1290. 23

[Hammarlund *et al.*, 1991] HAMMARLUND, P., LEVIN, B., AND LANSNER, A. (1991). BIOSIM—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. In: *Proceedings of ICANN-91*, edited by T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Amsterdam. North-Holland, p. 1477–1480. Espoo, Finland, June 24-28, 1991. 24

[Hammarlund *et al.*, 1992a] HAMMARLUND, P., LEVIN, B., AND LANSNER, A. (1992a). Biologically Realistic and Artificial Neural Network Simulators on the Connection Machine. In: *Science on the Connection Machine*, edited by T. Lippert, K. Schilling, and P. Ueberholz. World Scientific, 1992, p. 49–63. (Proceedings of the First European CM Users Meeting, June 16–17). 24

[Hammarlund *et al.*, 1992b] HAMMARLUND, P., LEVIN, B., AND LANSNER, A. (1992b). BIOSIM—A SIMD Parallel Simulator for Biologically Realistic Simulations of Neural Networks. (Submitted.). 24

[Hammarlund *et al.*, 1993] Hammarlund, P., Levin, B., and Lansner, A. (1993). Biologically Realistic and Artificial Neural Network Simulators on the Connection Machine. *International Journal of Modern Physics C* **4**: 49–63. 23

[Hammarlund and Lisper, 1992] Hammarlund, P. and Lisper, B. (1992). Data Parallel Programming—A Survey and a Proposal for a New Model. Technical report, TDS—Department of Telecommunication and Computer Systems, Royal Institute of Technology, S–100 44 Stockholm, Sweden. 23, 76

[Hammarlund and Lisper, 1993] Hammarlund, P. and Lisper, B. (1993). On the Relation between Functional and Data Parallel Programming Languages. Accepted for publication at the Sixth Conference on Functional Programming Languages and Computer Architecture. 23, 76

[Hansbo and Johnson, 1992] Hansbo, P. and Johnson, C. (1992). Space-time oriented streamline diffusion finite element methods for compressible flow on the CM-2. Technical report, Department of Mathematics, Chalmers Univeristy of Technology. 35

[Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behavior*. New York: John Wiley. 24

[Hedman and Laaksonen, 1992a] Hedman, F. and Laaksonen, A. (1992a). An Approach to Data Parallel Molecular Dynamics for Liquids. In: *Science on the Connection Machine*, edited by T. Lippert, K. Schilling, and P. Ueberholz. World Scientific, 1992, p. 41–48. (Proceedings of the First European CM Users Meeting, June 16–17). 61

[Hedman and Laaksonen, 1992b] Hedman, F. and Laaksonen, A. (1992b). An Efficient Massively Parallel Molecular Dynamics Program for Liquids with Million+ Particles. In: *Proceedings of the Joint Nordic Spring Meeting*, edited by P.-A. Lindgård, May 7-10 1992. 61

[Hedman and Laaksonen, 1993a] Hedman, F. and Laaksonen, A. (1993a). An Approach to Data Parallel Molecular Dynamics for Liquids. *International Journal of Modern Physics C* **4**: 41–48. 61

[Hedman and Laaksonen, 1993b] Hedman, F. and Laaksonen, A. (1993b). Data Parallel Large-Scale Molecular Dynamics for Liquids. *International Journal of Quantum Chemsitry* **46**: 27–38. 61

[Igel *et al.*, 1991] Igel, H., Mora, P., and Rodrigues, D. (1991). 3-D wave propagation using finite differences. In: *61st SEG Annual Meeting Expanded Abstracts*, 1991, p. 1577–1579. 67

[Jackson and Persson, 1992] Jackson, B. and Persson, M. (1992). A quantum mechanical study of recombinative desorption of atomic hydrogen on a metal surface. *J. Chem. Phys.* **96**: 2378. 43

[Klein and Thomas, 1993] Klein, W. and Thomas, L., (1993). SuperSymmetry in Condensed Matter Physics. (To be completed). 55

[Kolmogorov, 1941] Kolmogorov, A., (1941). Dokl. Akad. Nauk SSSR. 48

[Lander *et al.*, 1991] Lander, E. S., Langridge, R., and Saccocio, D. M. (1991). Mapping and Interpreting Biological Information. *Communications of the ACM* **34**: 32–39. 56

[Lansner and Ekeberg, 1989] Lansner, A. and Ekeberg, Ö. (1989). A One-layer Feedback, Artificial Neural Network with a Bayesian Learning Rule. *Int. J. Neural Systems* **1**: 77–87, Also extended abstract in Proceedings from the Nordic Symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland. 21, 22

[Lansner and Fransén, 1992] Lansner, A. and Fransén, E. (1992). Modelling Hebbian Cell Assemblies Comprised of Cortical Neurons. *Network* **3**: 105–119. 25

[Larsson and Migdalas, 1990] Larsson, T. and Migdalas, A. (1990). An Algorithm for Nonlinear Programs over Cartesian Product Sets. *Optimization* **4**: 535–542. 74

[Lax, 1973] Lax, P., (1973). Shock Waves and Entropy. Proceedings of "Contributions to Nonlinear Functional Analysis and Applications", Zarantonello E.A. (editor), Academic Press, New York. 69, 69

[Levander, 1988] Levander, A. (1988). Fourth order finite difference P-SV seismograms. *Geophysics* **53**: 1425–1436. 66, 67

[Levin, 1990] Levin, B. (1990). Implementation of the SANS algorithm on the CM–2. Tech. Rep. TRITA-NA-P9011, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 22

[Levin *et al.*, 1990] Levin, B., Hammarlund, P., and Lansner, A. (1990). BIOSIM—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. Tech. Rep. TRITA-NA-9021, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 24

[Levin and Lansner, 1992] Levin, B. and Lansner, A. (1992). Document Retrieval, Protein Sequence Matching and Sensor Selection Methods using a Neural Network. Tech. Rep. TRITA-NA-P9238, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 22

[Lindgren, 1992a] Lindgren, J. (1992a). *Waveform Inversion of Seismic Reflection Data Through Local Optimisation Methods.* PhD thesis, Uppsala University, Dept. of Geophysics, Sect. of Solid Earth Physics, Box 556, S-751 22 Uppsala, Sweden. 66

[Lindgren, 1992b] LINDGREN, K. (1992b). Evolutionary Behaviour in Simple Dynamics. In: *Artificial Life II*, edited by C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Redwood City, CA. Addison-Wesley, p. 295–312. 54

[Lindgren and Nordahl, 1993] LINDGREN, K. AND NORDAHL, M. (1993). Evolutionary dynamics of spatial games. Technical report, Institute of Physical Resource Theory, Chalmers University of Technology. (preprint). 54

[Migdalas, 1990] MIGDALAS, A. (1990). A Regularization of the Frank-Wolfe Method. Technical report, Division of Optimization, Linköping Institute of Technology. LiTH-MAT-R-90-10. 74

[Modi, 1988] MODI, J. J. (1988). *Parallel Algorithms and Matrix Computation.* Oxford University Press, New York. 60

[Olsson and Yström, 1993] OLSSON, F. AND YSTRÖM, J. (1993). Some properties of the Upper Convected Maxwell Model for Viscoelastic Fluid-Flow. (To appear in Journal of Non-Newtonian Fluid Mechanics). 34

[Olsson, 1991] OLSSON, P. (1991). Stable Approximation of Symmetric Hyperbolic and Parabolic Equations in Several Space Dimensions. Technical Report 138, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden. 39, 39, 40

[Olsson, 1992] OLSSON, P. (1992). *High-Order Difference Methods and Dataparallel Implementation.* PhD thesis, Dept. of Scientific Computing, Uppsala University. 39

[Palm *et al.*, 1993] PALM, T., THYLÉN, L., NILSSON, O., AND SVENSSON, C. (1993). Quantum Interference Devices and Field Effect Transistors: A Switch Energy Comparison. (To appear in Journal of Applied Physics). 42

[Parrish and Newman, 1970] PARRISH AND NEWMAN, (1970). Current distribution on plane, parallel electrodes in channel flow. (J. Electrochemical Soc.). 53

[Rynefors, 1982] RYNEFORS, K. (1982). UNIMOL: A program for Monte Carlo simulation of RRKM unimolecular decomposition in molecular beam experiments. *Comp. Phys. Comm.* **27**: 202. 61

[Sawley, 1993] SAWLEY, M. (1993). Control- and Data-Parallel Methodologies for Flow Calculations. In: *Supercomputing Europe '93*, edited by R. Tucker, Utrecht. Royal Dutch Fairs, p. 169–187. 31, 33

[Sawley and Bergman, 1991] SAWLEY, M. AND BERGMAN, C. (1991). Computational Fluid Dynamics on Massively-parallel SIMD Computers. *EPFL Supercomputing Review* **3**: 20–23. 31

[Sawley and Bergman, 1993] Sawley, M. and Bergman, C. (1993). A Comparative Study of the Use of the Data-Parallel Approach for Compressible Flow Calculations. Technical Report T-93-2, IMHEF, EPFL. (Submitted). 32

[Sawley *et al.*, 1992] Sawley, M., Perrel, F., Bergman, C., and Persson, I. (1992). Control and Data-Parallel methodologies for Flow Calculations using Supercomputers. *EPFL Supercomputing Review* **4**: 2–8. 31

[She *et al.*, 1992] She, Z., Aurell, E., and Frisch, U. (1992). The Inviscid Burgers Equation with Initial data of Brownian type. *Comm. Math. Phys.* **148**: 623–641. 70, 70

[TMC, 1991a] TMC, Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142–1264. (1991a). *Connection Machine: Programming in C**, 6.1 edition, 1991. 76

[TMC, 1991b] TMC, Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142–1264. (1991b). *Connection Machine: Programming in *Lisp*, 6.1 edition, 1991. 76

[TMC, 1992] TMC, Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142–1264. (1992). *CMSSL Release Notes for the CM-200*, 3.0 edition, June 1992. 68

[von Heijne, 1988] von Heijne, G. (1988). Getting sense out of sequence data. *Nature* **333**: 605–607. 56

[Wallin, 1991] Wallin, E. (1991). Optimized sequence matching on the CM2. Master's thesis, Department of Telecommunication and Computer Systems, Royal Institute of Technology. 22

[Wallin *et al.*, 1993] Wallin, E., Wettergren, C., Hedman, F., and von Heijne, G. (1993). Fast Needleman–Wunsch scanning of sequence databanks on a massively parallel computer. *CABIOS* **9**: 117–118. 57

[Warren, 1983] Warren, D. H. D. (1983). An Abstract Prolog Instruction Set. SRI Technical Note 309, SRI International, Menlo Park. 78

[Wenger and Östlund, 1993] Wenger, F. and Östlund, S., (1993). d-wave pairing in tetragonal superconductors. (To be published in Phys.Rev.B). 46

Index