Parallel Molecular Sequence Analysis on Workstations in the Internet

Volker Strumpen*
strumpen@ifi.unizh.ch
Department of Computer Science, University of Zurich
Winterthurerstr. 190, CH-8057 Zurich

June 24, 1993

Abstract

We present a highly scalable approach to *distributed parallel computing* on workstations in the Internet which provides significant speedup to molecular biology sequence analysis. Recent developments show that smaller numbers of workstations connected via a local area network can be used efficiently for parallel computing. We show that a massively parallel approach using *several hundred workstations*, dispersed over *all continents*, can successfully be applied for solving problems with low requirements on communication bandwidth. We calculated the optimal local alignment scores between a single genetic sequence and all sequences of a genetic sequence database using the `ssearch` code which is well known among molecular biologists. In a heterogeneous network with more than *800 workstations* this job terminated after several minutes, in contrast to several days it would have taken on a single machine.

1 Introduction

Recent developments in distributed parallel computing are aiming at the utilization of inexpensive computing resources for parallel computations as available in heterogeneous networks of workstations. Systems like *Linda* [1] or *PVM* [16] are widely used, especially for problems of scientific computing. Typically, up to ten machines, in more computing and

* Supported by *Siemens AG (ZFE), Germany* and *Schweizer Bundesamt für Konjunkturfragen*, Grant No. 2254.1

less communication intensive applications some tens of machines are employed within local area networks. Experiments in heterogeneous environments, spanning several LANs have also been reported [7]. We developed a prototypical software platform for distributed massively parallel computing to support the use of a virtually unlimited amount of machines connected via the Internet. Such a system is suited for parallel applications, where the computational expense within a LAN clearly outweighs the interprocessor communication across LAN borders, because the latency of the Internet connections is considerable.

Compared to conventional supercomputers, powerful workstations offer less floating point performance but almost comparable integer performance. Recent performance evaluations of molecular biology sequence analysis which do not require floating point but integer computations showed that the `blastpm` code for sequence analysis runs just 20% slower on a Sun4 (SPARCstation 330) compared to an unoptimized single processor run on a Cray Y-MP 8/864 [13]. The Y-MP processor delivered only approximately 40 MIPS which coincides with the peak performance of a Sun SPARCstation2.

In this paper, we show the possibility and describe how we employed a distributed workstation network of more than 800 machines, geographically scattered over all continents, to accelerate a computational problem, which fits the capabilities of this system's architecture. The computing challenge of molecular biology sequence analysis, one of the key techniques of the Human Genome projects [12][13], is accelerated from runtimes of a couple of days on a single workstation or Cray processor to some minutes with our platform.

Today's DNA and protein databases are growing extremely fast and seem to exceed any computational power for similarity searches already in near future. A distributed system may be a solution to this problem, where huge distributed databases are geographically dispersed and accessible via electronic networks. Since the rapidly growing size of these databases prohibits frequent transfers via networks, the computing power has to be provided at the geographical site of the database. The structure of our system is therefore a network of globally distributed local area networks, each with access to a local database.

2 Molecular Sequence Analysis

The biological challenge rises from the fact that DNA and protein molecules are the primary genetic materials which encode information necessary to understand life. The DNA consists of sequences of bases, and protein molecules are build from amino acids. The amino acids are encoded in sequences of three bases in the DNA. The function of a protein is determined by the structure in which the amino acids fold into a three dimensional macromolecule.

In order to understand this complex folding structure, the folding process would have to be simulated. However, Sander *et al.* [12] estimate that such simulations need 12 hours of CPU time on a computer with one million teraflops. Therefore, molecular biologists use less complex sequence analysis methods to understand the functionality of a new genetic sequence of a DNA or protein polymer by finding a DNA or protein with similar sequence structure and known functionality. Presently, this approach is the only well recognized to understand the genetic code.

Modern laboratory techniques provide fast and cheap means of analyzing the sequence of bases in DNA and the sequence of amino acids in protein molecules. There exist databases of extracted DNA and amino acid sequences containing about 100,000 sequences, which can be searched for similarity. In our experiments we used the NCBI-GenBank (version 75.0) [8] which contains a total of 106,684 genetic sequences comprising 126,212,259 bases. Due to the growing size of the databases even the sequence analysis algorithms pose serious computational problems. For the search of a 5,000 sequence human chromosome versus a database of 3,000,000 sequences, Bork [2] estimates that a powerful workstation would take 5,000 days even with the `tfasta` approximation algorithm.

The similarity search can be conducted using various techniques which are still subject to active research. The computational aspects can be considered by regarding the sequences of bases of DNA or amino acids of a protein as strings over an alphabet. Wagner and Fischer [17] developed an algorithm for string correction in 1973 which has been adapted to the needs of biologists by Smith and Waterman [14] as well as Needleman and Wunsch [9]. These algorithms compute similarity scores for measuring the similarity between two sequences with respect to an optimal local alignment metric [6][10]. Due to the computational complexity of this approach, faster but heuristic algorithms, implemented for example in the `blast`, `fasta`, or `tfasta` programs can be used. The algorithm we used for obtaining the presented results on parallel alignment score computations is the rigorous optimal local alignment algorithm of Smith and Waterman. This algorithm is implemented in the `ssearch` program which is included in the popular `fasta` program package [10]. We determined those 50 genetic sequences of the NCBI-GenBank which best match a given query sequence by computing and sorting the scores between the query sequence and all sequences of the gene bank. We chose two query sequences consisting of 105 and 773 bases.

3 Parallel Internet Computing

Based on our experience with *The Parform* [3][4], we extended our work on parallel computing from local area networks to world-wide area networks. A system which performs

Internet-wide parallel computations has to fulfil the following main requirements: (1) portability, which is indispensible to employ an appropriate number of machines, (2) non-intrusion with respect to other users in the network by utilizing only idling CPU shares and not congesting the network, and (3) efficiency, which can only be the secondary goal of our efforts with respect to the interactive users of the workstations employed.

The existing infrastructure of the Internet and the increasing availability of powerful workstations running a UNIX operating system are the basis for this work. The de facto standard for open system interconnection, the TCP/IP Internet Protocol Suite and the Berkeley Socket Interface [5][15] provide the tools to implement a parallel computing platform spanning LANs in the Internet. Another feature of the Internet is the commonly installed Client-Server structure of individual LANs. Although there exist many administrative differences, features such as network file systems or access rights are in principle uniformly available. In contrast to the variety of computer architectures of earlier days, modern workstations also simplify the implementation of low level communication, because they use identical data representations.

The guiding principle for the design of our system architecture and its implementation has been simplicity. The system therefore provides only fundamental features: (1) remote process startup and termination, (2) asynchronous interprocess communication, (3) extensive logging and error handling, (4) fault tolerance and (5) load balancing. To achieve portability, all mechanisms are implemented in C and on top of the Berkeley Socket Interface. The first two points are based on UNIX system programming [5][15]. Error handling on system level and application level, and logging proved to be invaluable aids during the development of the software. Fault tolerance and load balancing have been adapted to the needs of the application and are described in Section 4. Fault tolerance is an indispensible tool when combining many machines which are scattered all over the world. First, the higher the number of machines used the higher the chance that one or more will crash or simply get switched off during a computation. Second, if these machines are installed in different geographical time zones, machines appear to be switched on and off at any time for the central user. Load balancing is an essential feature to achieve efficiency because of the heterogeneous performance profile of the participating machines and the permanently changing load situation caused by the owners of the workstations.

Technical problems such as distributing the actual object code versions among the different heterogeneous LANs have been managed by a collection of *shell scripts*. To avoid any disturbance of local users, our processes use the C library call “nice” to keep the impact on the owners' processes as low as possible. To prevent accidentally lost processes from wasting CPU power, we implemented an alarm mechanism to terminate those processes after a predefined period of time. Security precautions of various local system administrations forced us to implement two different startup mechanisms for remote processes via the UNIX daemons "rexecd" and "rshd", respectively. Resource constraints, such as the limited number of file descriptors per process, contributed to the decision to implement the process topography in a tree structure as explained below. Despite of these resource limits, the number of leaf nodes of the tree can be increased arbitrarily by adding non-leaf nodes at the expense of higher communication costs because of the increased depth of the tree.

Besides the technical problems, organization and logistics of such an approach are quite complex, especially obtaining accounts and coordinating some 30 LANs with several hundred machines and users. Many organizations are not fond of giving the permission to foreign users to utilize their CPU power and providing disk space. Therefore, we are especially grateful to all people who generously supported us and did not delimit our disk space at a smaller capacity than necessary for the indispensible storage of gene data at the site where they have to be accessed during the match.

4 Parallelization Issues

The goal for our application is to accelerate the comparison of a molecular sequence of unkown functionality against all sequences stored in a gene database. The query sequence is sent from a central interactive working site to all sites holding parts of the distributed gene database. After the local area networks have finished the sequence analysis, the results are collected, sorted and delivered to the central administration process of the user.

The process structure of our system is based on a tree topography. The root node accommodates the *central administrative process*. The leaf nodes perform the sequence analysis. We distinguish a *global layer* and a *local layer*. The global layer comprises the root node and all descendants down to, but excluding the father nodes of the leaves. The local layer consists of the leaf nodes and their father nodes. Thus, all participating LANs are part of the local layer, while the global layer is responsible for connecting the LANs with the central administrative process.

The father nodes of the leaves accommodate an *administrative process*. These are either directly connected to the central administrative process, or via *router processes* which are responsible for bidirectional message routing. The router processes, as a part of the global layer, on the one hand are used to construct an arbitrarily sized and shaped tree topography by means of configuration files. On the other hand, they might be necessary to connect subnets which are not directly accessible from the Internet.

The gene database is distributed among all LANs. Thus, parallelization on the global layer is achieved by running the sequence analysis in each LAN on a different part of the database. On the local layer, that is within each LAN, the search is parallelized by assigning parts of the local database to the leaf nodes which run a conventional sequential search algorithm.

Parallelization within a LAN is organized by a *workpool* which is provided by the administrative process. The workpool schedules the sequence analysis by assigning genes of the local database to the *executive processes* which are accommodated by the leaf nodes. In practice, the father node of the leaves is mapped on a server machine of the LAN and the leaf nodes itself on the client machines, the workstations. The workpool provides two essential features for large globally distributed heterogeneous environments: *Fault tolerance* and *dynamic load balancing*. Fault tolerance is provided by keeping a task in the workpool if its executive process aborted because of a malfunctioning processor, for example. The workpool reschedules such tasks to another available processor. By placing the administrative processes on server machines which in general are indispensible to provide access to the mass storage devices and therefore the local gene database, no other fault tolerance mechanisms have been necessary. Dynamic load balancing is provided for efficiency by scheduling parts of the local database dynamically to the executive processes. When an executive process finishes its task, it reports its state to the workpool, which in turn assigns a new task to this process. The granularity of the tasks determines the efficiency of the computation. We obtained reasonably high efficiencies within a LAN with the number of tasks being proportional to the number of sequences in the local database, their lengths and the number of participating workstations as determined at program startup.

5 Experimental Results

For our experiments we implemented a simplified but optimized rigorous dynamic programming algorithm [6] by our own and also hooked up the `fasta` and `ssearch` codes [10] to our system. Here, we present the results of our experiments with the `ssearch` code to calculate the similarity scores of two genetic sequences against the NCBI-GenBank [8]. The `ssearch` program has been used with the default scoring matrix and gap penalties. We performed a number of single-query measurements comparing a *bacteriophage* sequence and a *human germline (AIDS)* sequence consisting of 105 and 773 bases, respectively, with the complete database.

Table 1 reports elapsed runtimes of two experiments, A and B, with our parallel implementation in comparison to extrapolated sequential runtimes on a Sun SPARCstation1 and a Sun SPARCstation2. In this table, the *sequence length* denotes the number of bases of the DNA sequence which has been matched against the genes of the gene bank. All runtimes are wall clock times. To determine the extrapolated sequential runtimes, we compared our two genes with an 845 gene portion of the database. From the measured runtimes we extrapolated the sequential runtimes using the fact that the complexity of the sequencing algorithm is linear with the number of genes in the database and the number of bases of the genetic sequences to be matched.

Exp.	Number of Workstations	Sequence Length	Parallel Runtime	Sequential (SPARC1)	Sequential (SPARC2)
A	803	105	5.0 min	21.5 hours	10.4 hours
B	803	773	18.4 min	6.6 days	3.2 days

Table 1. Performance results.

We want to emphasize that these measurements have been conducted under normal daily loads. The computation utilized CPU power which would be idling otherwise. Therefore, the price/performance ratio of such configurations can be regarded to be *close to zero*.

Quantitatively, the two configurations which yielded the results presented in Table 1 are characterized in Table 2. The entries of the bottom two lines of Table 2 have been determined with a small loop consisting of an integer test load with an average run time of some 10 seconds on a SPARCstation2. Elapsed (wall clock) time and CPU time of this test load were measured and scaled so that a dedicated SPARCstation2 would deliver 40 MIPS. The MIPS values above are the sum of these MIPS values, measured on all machines used just before the sequencing. These values give a rough idea of the *total performance* we would get from a dedicated configuration. They are considerably higher than those of today's fastest supercomputers.

	Experiment *A*	Experiment *B*
Machines used	803	803
Local area networks	31	31
Administrative processes	46	46
Router processes	7	7
Machine types	32	32
MIPS (SPARCstation2 $\equiv$ 40 MIPS)	26,800	26,790
Average idle percentage	93.0 %	92.7 %

Table 2. System configuration.

Table 3 shows the different machine models which have been incorporated in our configuration. There have been various different types of most machine models and even more different UNIX operating system versions.

Manufacturer	Machine Models
DEC	DECstation, DECmicrovax
HP	HP9000, HP Apollo 9000
IBM	RS6000
NEC	EWS-4800
SGI	Crimson, Indigo, Iris 4D, Personal Iris
Sony	NEWS
Sun	Sun3, Sun4, SPARC10, SPARCserver

Table 3. Employed Machine Models.

In Table 1 and Table 2 we presented those two measurements of our experiments with the maximum number of machines. Since a network of several hundred machines permanently changes its configuration, it is impossible to get a fixed number of machines with reproducable load situation for reference measurements. We had hands on about 1,000 machines but typically about 20% of them were shut down. Most of the LANs we used had processor idle percentages of at least 90% of the time when our parallel application did not run. Such values are very common as we also confirmed by logging CPU idle percentage data in our LAN for a period of several months. This average utilization remains almost constant from our experience, although the utilization of individual workstations varies permanently. However, some LANs with high speed workstations were heavily loaded with parallel cluster applications.

Exp.	Sequence Length	Minimum Runtime	Maximum Runtime	Average Runtime	Speedup vs. SPARC1	Speedup vs. SPARC2
A	105	32 sec	302 sec	1.8 min	258	125
B	773	172 sec	1103 sec	8.0 min	514	249

Table 4. Performance data.

Table 4 gives additional information about our measurements. The minimum and maximum runtimes denote the minimum and maximum of the elapsed runtimes in the individual LANs. The maximum value determines the total elapsed runtime given in Table 1. The range of these values is relatively wide because of the rough static partitioning of the gene database by hand and because of the permanently changing load situation of the individual workstations. The former one has been the dominant effect. The runtime distribution and hence the total performance can easily be improved by proper organization of the distributed database. However, in our experiments this was not possible since on some of the LANs the disc space available for these experiments was too small.

The *speedup* values are calculated with respect to the elapsed parallel runtimes and the sequential runtimes given in Table 1. The *average runtime* denotes the mean runtime with respect to the runtimes of all participating LANs. Calculating speedups with these values yields 717 and 1188 corresponding to the 105 base and 773 base query sequences and the sequential runtimes on a SPARCstation1. The value of 1188 can also be interpreted as the number of processors of a virtual machine consisting of SPARCstation1 processing elements and delivering almost optimal performance.

We presented the MIPS performance of our dedicated configuration in Table 2. To get an idea of the efficiency of our experiments, we estimate the MIPS performance of our application by assuming the sequential run to perform 24 MIPS on a SPARCstation2, which is about 60% of its peak performance, and scaling this value with the achieved speedup given in Table 4. These values are given as *estimated MIPS* in Table 5. The *average MIPS* values are calculated with respect to the average runtimes given in Table 4 and indicate the possible performance of our configuration with an optimally balanced distributed database.

Exp.	Sequence Length	Estimated MIPS	Average MIPS
A	105	3,000	8,350
B	773	6,000	13,650

Table 5. MIPS Performance data.

Efficiency, which is traditionally the primary goal of parallelization, in our case has to obey the boundary condition to not disturb other users in the network. Nevertheless, efficiency remains our goal. It is known [4], that interprocessor communication causes performance degradation because of the low bandwidth of today's networks like the Ethernet. In our application, only communication within the LANs deserves closer examination, because communication across the LANs comprises just the initial distribution of the query sequence and the final collection of the best alignment scores. We investigate the behavior and efficiency of our implementation within the LAN of our institute by means of Figure 1.

Figure 1 shows *normalized speedups* of the two sequence analyses with query sequences consisting of 105 bases and 773 bases with respect to a unit processing element delivering a performance equivalent to a SPARCstation1, which is the slowest machine of the configuration. All participating machines have been benchmarked with the sequence analysis application which is subject of our investigations. The normalized performances are given in Table 6. The speedup measurements up to 18 machines have been conducted on SPARCstation1 only. Then, other machines have been added in the order as listed in Table 6. We used both processors of the Iris 4D/320, each with a normalized performance of 2.1.

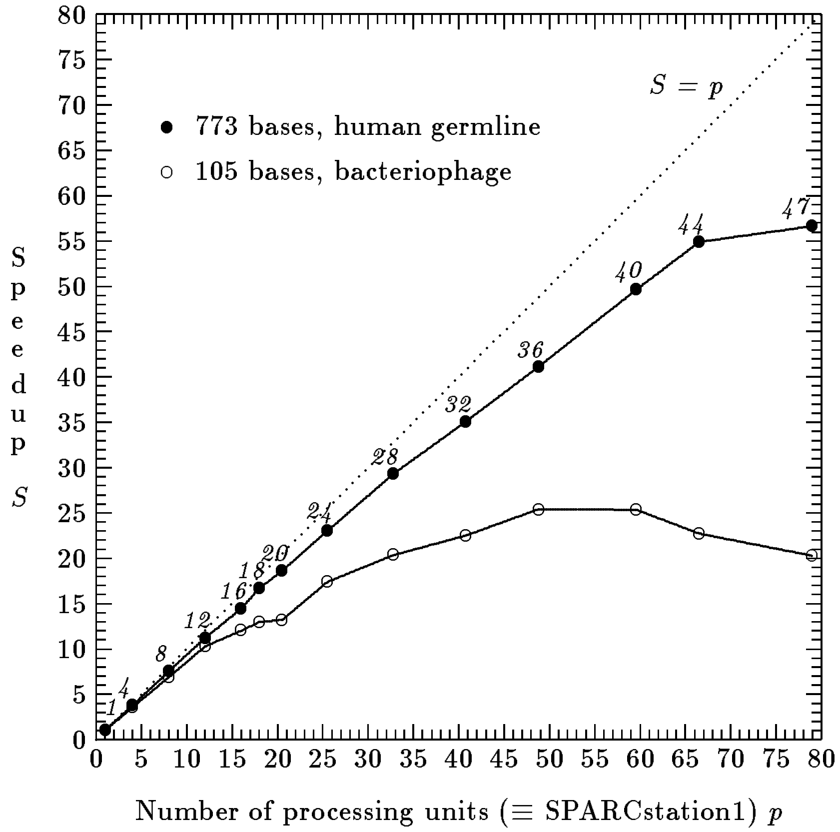

Figure 1. Normalized speedup in heterogeneous environment (cf. Table 6).

The numbers above the filled circles in Figure 1 denote the number of machines actually used. Both speedup curves in Figure 1 show a typical behavior. Up to 18 processors, we obtain slightly sublinear speedup as expected with respect to the increasing communication rate. Then, the additional and more powerful machines raise the overall performance. With higher numbers of machines the increase of speedup is reduced. The two curves are distinguished by the ratio of computation and communication, which increases proportional to the number of bases of the query sequence. Communication during the match of the

105 base query sequence even leads to Ethernet congestion and decreasing speedup with more than 36 machines. Matching the 773 base gene, this effect does not occur with the number of machines used, because the ratio of computation and communication is still high enough.

Machine	Number	Normalized Performance
Sun SPARCstation1	18	1.0
Sun SPARCstation1+	7	1.25
Sun SPARCstation2	11	2.0
Sun SPARCserver490	2	1.7
Sun SPARCserver690	2	3.7
SGI Iris 4D/70	1	1.1
SGI Iris 4D/20	1	1.7
SGI Iris 4D/320 (2 proc.)	2	2.1
SGI Personal Iris	1	2.1
SGI Iris Crimson	1	5.4
IBM RS6000	1	5.0

Table 6. Machine configuration and normalized performance (cf. Figure 1).

The communication within a LAN comprises the exchange of identifiers and partitioning information between the workpool and all executive processes, and the reading of the local gene database. The latter one is the reason for the high traffic and Ethernet congestion with higher numbers of processors. This implementation can still be improved because each executive process reads the whole database during the sequence analysis. The amount of data transferred via the Ethernet is therefore the size of the data base multiplied by the number of executive processes. If each executive process would only read that part of the database which it utilizes for matching, the amount of transferred data would be reduced to the size of the database, independent of the number of participating processes. This would improve speedup and efficiency especially in those LANs with high numbers of workstations. The benefit for sequence analyses with shorter query sequences will be more significant because the improvement of the ratio of communication and computation will be relatively higher.

6 Conclusion and Future Work

This work shows that a massively parallel approach using several hundred workstations, dispersed over all continents and connected via the Internet, can be successfully employed for solving problems with low communication requirements across LAN borders. Technically, we have confirmed the following points:

- Managing a parallel computation on about 800 workstations located in 31 local area networks and five continents.
- Coordinating a complex distributed and heterogeneous architecture, including different file system structures, and permission, authorization, and security mechanisms.
- Automatized distribution and installation of a portable program to 1,000 workstations of 7 different manufacturers and numerous different UNIX operating system versions.
- The proof that massively parallel computing in the Internet is feasible for appropriate applications.
- Running a molecular sequence analysis program in several minutes, whereas the sequential implementation would calculate several days.

Although many technical problems have in principle been solved by our work this approach has to be refined technically as well as organizationally. It would be very interesting to extend our prototype and design a general purpose platform for parallel computing in the Internet. Such a platform might serve as a basis for investigations on the feasibility of other compute intensive problems to be solved with large numbers of machines dispersed in the Internet.

Combining our approach with modern database technology would be a next step towards a parallel, distributed system for molecular sequence analysis. To establish a system that utilizes the techniques developed in this paper, some technical as well as organizational and logistical efforts have to be undertaken. A straightforward design would include a *distributed database*, where replication could be employed to guarantee reliability and load balancing at the same time. Since the genetic sequences in such databases are *read-only* data, consistency mechanisms can easily be implemented. Currently, researchers are investigating new concepts for managing biological DNA and protein databases [11]. The main problem of achieving efficient load balancing with a particular distribution might be solved with a *Balance-by-Growth* concept, where those new sequences which are input to the distributed database would be stored at a site which has computationally been least utilized during previous computations. Assuming a sufficiently high rate of new sequences per day proper balancing could even be guaranteed if new machines join the system. Furthermore, coordinated shutdown hours of the individual LANs for administrative purposes would increase the system reliability. Most of these aspects are technically straightforward to implement but a challenge to be done.

Acknowledgements

Many people have contributed to make this work possible. First of all, Clemens H. Cap, who had the initial idea and participated in the organization, design and implementation of the platform. We thank all our friends for supporting our requests to obtain accounts on their LANs and all the system administrators who have been confident in our work. The system administrators Beat Rageth and Rico Solca of our Institute provided disk space to store the gene database files, Martin Härtig reported the unpredicted termination of his window system and Paul Verschure complained energetically about exhausted process tables and memory. Both helped us fixing bugs. Nick Almassy and Barbara Rieche improved the quality of this paper. Martin Dürst gave helpful comments and support. Edgar Lederer accompanied the whole project encouragingly and significantly contributed to the quality of this work. We are grateful to Kurt Bauknecht and Lutz Richter for providing us with a stimulating working environment. From all the people and institutions we list in the following, we want to emphazise our special thanks to Walter Huber for his continuous support in Munich:

- Austria
  - Günther Berchtolt, Karl-Franzens-University of Graz
  - Walter Müller, Bernhard Quatember, University of Innsbruck
  - Bruno Buchberger and Karoly Erdei, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz
  - Harald Fritscher, Technical University of Vienna
- Australia
  - Anthony Botham and Ross Wilkinson, Computer Science Department, Royal Melbourne Institute of Technology
- Canada
  - Stewart Lee, Peter Pereira and Songnian Zhou, University of Toronto
- France
  - Christel Seguin, ONERA-CERT, Toulouse
- Germany
  - Siegfried Knecht and Michael Weber, Central Institute for Applied Mathematics, KFA Jülich
  - Arndt Bode, Walter Huber, Harald Loevenich, Thomas Schnekenburger, Klaus Tilk and Jörg Veit, Department of Computer Science, Technical University of Munich
  - Karl Behler, Max Planck Institute for Plasma Physics, Munich-Garching
  - Thomas Setz, Computer Science Department, University of Saarland, Saarbrücken

- Greece
  - Christos Nikolaou and Stelios Sartzetaki, Institute of Computer Science, Foundation of Research and Technology Hellas (FORTH), Heraklio
- Japan
  - Tatsuki Ogino and Hideo Sugai, Solar-Terrestrial Environment Laboratory, Nagoya University
  - Yoshihisa Shinagawa, Department of Information Science, University of Tokyo
- Norway
  - Jan Berger-Hendriksen and Paul Tvete, Department of Informatics, University of Bergen
- USA
  - Bob Byrnes and Phil Oldiges (DEC), School of Electrical Engineering, Cornell University
  - Charles Karney, Plasma Physics Laboratory, Princeton University
  - William Drummond and William H. Miner, Jr., Fusion Research Center and Institute for Fusion Studies, University of Texas at Austin
- South Africa
  - Michael Alport, University of Durban
- Switzerland
  - Heike Horn and Markus Studer (TNT), Institute for Applied Mathematics, University of Bern
  - Nicolas Mayencourt and Christian Pellegrini, Departement of Computer Science, University of Geneva
  - Ilario Tironi, Laboratory of Physical Chemistry, Swiss Federal Institute of Technology, Zurich
  - Christoph Wicki, Integrated Systems Laboratory, Swiss Federal Institute of Technology, Zurich
  - Peter Arbenz and Christoph Sprenger, Interdisciplinary Project Center for Supercomputing, Swiss Federal Institute of Technology, Zurich
  - All colleagues of the Computing Center and the Department of Computer Science of the University of Zurich

Furthermore, Martin Bichsel, Steven Benner, Ferdinand F. Cap, Gaston Gonnet, Terence Hale, Rolf Pfeifer, Steffi Schmidt and Janos Varga supported our efforts to contact potential contributors of machines.

References

[1] Ahuja S., Carriero N., Gelernter D., "Linda and Friends," *IEEE Computer* **19**(8) 1986, 26–34.

[2] Bork P., "Entschlüsselung von Proteinfunktionen mit Hilfe des Computers: Erkennen und Interpretation entfernter Sequenzähnlichkeiten," in *Informatik in den Biowissenschaften*, Eds. Hofestädt R., Krückeberg F., Lengauer T., Springer-Verlag (1992) 67–78.

[3] Cap C.H., Strumpen V., "The Parform — A High Performance Platform for Parallel Computation in a Distributed Workstation Environment," Technical Report **92.07**, Department of Computer Science, University of Zurich, June 1992.

[4] Cap C.H., Strumpen V., "Efficient Parallel Computing in Distributed Workstation Environments," accepted for publication in *Parallel Computing*.

[5] Comer D.E., Stevens D.L., Internetworking With TCP/IP, Vol. II: Design, Implementation, and Internals, Prentice-Hall (1991).

[6] Lipton R.J., Marr T.G., Welsh J.D., "Computational Approaches to Discovering Semantics in Molecular Biology," *Proceedings of the IEEE* **77**(7) 1989, 1056–1060.

[7] Nakanishi H., Rego V., Sunderam V., "Superconcurrent Simulation of Polymer Chains on Heterogeneous Networks," *Supercomputing '92*, Minneapolis, 1992, 561–569.

[8] NCBI-GenBank, National Center for Biotechnology Information, National Library of Medicine, 38A, 8N805, 8600 Rockville Pike, Bethesda, MD 20894.

[9] Needleman S.B., Wunsch C.D., "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology* **48**(3) 1970, 443–453.

[10] Pearson W.R., "Searching Protein Sequence Libraries: Comparison of the Sensitivity and Selectivity of the Smith-Waterman and FASTA Algorithms," *Genomics* **11**(3) 1991, 635–650.

[11] Rieche B., Dittrich K.R., "An Extensible Semantic Data Model for Molecular Biology", Technical Report, Department of Computer Science, University of Zurich, June 1993.

[12] Sander C., Schneider R., Stouten P., "The Human Genome and High Performance Computing in Molecular Biology," in *Supercomputer '92, Anwendungen, Architekturen, Trends*, Ed. Meuer H.W., Springer-Verlag (1992) 32–48.

[13] Smith D.W., Jorgensen J., Greenberg J.P., Keller J., Rogers J., Garner H., Ten Eyck L., "Supercomputers, parallel processing, and genome projects," in *Biocomputing: Informatics and Genome Projects*. Ed. Smith D., Academic Press, in press.

[14] Smith T.F., Waterman M.S., "Identification of Common Molecular Subsequences," *Journal of Molecular Biology* **147**(1) 1981, 195–197.

[15] Stevens W.R., UNIX Network Programming, Prentice-Hall (1990).

[16] Sunderam V.S., "PVM: A framework for parallel distributed computing," *Concurrency: Practice and Experience* **2**(4) 1990, 315–339.

[17] Wagner R.A., Fischer M.J., "The String to String Correction Problem," *Journal of the ACM* **21**(2) 1974, 168–173.