VU Research Portal

The distributed ASCI supercomputer project

Bal, H.E.; Bhoedjang, R.A.F.; Hofman, R.; Jacobs, C.J.H.; Kielmann, T.; Maassen, J.; van Nieuwpoort, R.V.; Romein, J.W.; Renambot, L.P.M.; Ruhl, T.; Veldema, R.S.; Verstoep, M.; Baggio, A.G.; Ballintijn, G.C.; Kuz, I.; Pierce, G.; van Steen, M.R.; Tanenbaum, A.S.; Doornbos, G.; Germans, D.M.

published in
ACM SIGOPS Operating Systems Review
2000

DOI (link to publisher)
10.1145/506106.506115

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Bal, H. E., Bhoedjang, R. A. F., Hofman, R., Jacobs, C. J. H., Kielmann, T., Maassen, J., van Nieuwpoort, R. V., Romein, J. W., Renambot, L. P. M., Ruhl, T., Veldema, R. S., Verstoep, M., Baggio, A. G., Ballintijn, G. C., Kuz, I., Pierce, G., van Steen, M. R., Tanenbaum, A. S., Doornbos, G., ... van der Steen, A. (2000). The distributed ASCI supercomputer project. *ACM SIGOPS Operating Systems Review*, *34*, 76-96. https://doi.org/10.1145/506106.506115

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 24. Sep. 2021

The Distributed ASCI Supercomputer Project

Henri Bal, Raoul Bhoedjang, Rutger Hofman, Ceriel Jacobs, Thilo Kielmann, Jason Maassen,
Rob van Nieuwpoort, John Romein, Luc Renambot, Tim Rühl, Ronald Veldema, Kees Verstoep,
Aline Baggio, Gerco Ballintijn, Ihor Kuz, Guillaume Pierre, Maarten van Steen, Andy Tanenbaum,
Gerben Doornbos, Desmond Germans, Hans Spoelder, Evert-Jan Baerends, Stan van Gisbergen

Faculty of Sciences, Vrije Universiteit

Hamideh Afsermanesh, Dick van Albada, Adam Belloum, David Dubbeldam, Zeger Hendrikse,
Bob Hertzberger, Alfons Hoekstra, Kamil Iskra, Drona Kandhai, Dennis Koelma,
Frank van der Linden, Benno Overeinder, Peter Sloot, Piero Spinnato

Department of Computer Science, University of Amsterdam

Dick Epema, Arjan van Gemund, Pieter Jonker, Andrei Radulescu, Cees van Reeuwijk, Henk Sips

Delft University of Technology

Peter Knijnenburg, Michael Lew, Floris Sluiter, Lex Wolters

Leiden Institute of Advanced Computer Science, Leiden University

Hans Blom, Cees de Laat, Aad van der Steen

Faculty of Physics and Astronomy, Utrecht University

Abstract

The Distributed ASCI Supercomputer (DAS) is a homogeneous wide-area distributed system consisting of four cluster computers at different locations. DAS has been used for research on communication software, parallel languages and programming systems, schedulers, parallel applications, and distributed applications. The paper gives a preview of the most interesting research results obtained so far in the DAS project.[1]

[1] More information about the DAS project can be found on http://www.cs.vu.nl/das/

1 Introduction

The Distributed ASCI Supercomputer (DAS) is an experimental testbed for research on wide-area distributed and parallel applications. The system was built for the *Advanced School for Computing and Imaging* (ASCI)[2], a Dutch research school in which several universities participate. The goal of DAS is to provide a common computational infrastructure for researchers within ASCI, who work on various aspects of parallel and distributed systems, including communication substrates, programming environments, and applications. Like a metacomputer [41] or computational grid [17], DAS is a physically distributed system that appears to its users as a single, coherent system. Unlike metacomputers, we designed DAS as a *homogeneous* system.

The DAS system consists of four cluster computers, located at four different universities in ASCI, linked together through wide area networks (see Figure 1). All four clusters use the same processors and local network and run the same operating system. Each university has fast access to its own local cluster. In addition, a single application can use the entire wide-area system, for example for remote collaboration or distributed supercomputing. DAS can be seen as a prototype computational grid, but its homogeneous structure makes it easier to avoid the engineering problems of heterogeneous systems. (Heterogeneous systems are the object of study in several other projects, most noticeably Legion [18] and Globus [16]). DAS can also be seen as a cluster computer, except that it is physically distributed.

This paper gives a preview of some research results obtained in the DAS project since its start in June 1997. We first describe the DAS architecture in more detail (Section 2) and then we discuss how DAS is used for research on systems software (Section 3) and applications (Section 4). Finally, in Section 5 we present our conclusions.

2 The DAS system

DAS was designed as a physically distributed homogeneous cluster computer. We decided to use cluster technology because of the excellent price/performance ratio of commodity (off-the-shelf) hardware. We wanted the system to be distributed, to give the participating universities fast access to some local resources. One of the most important design decisions was to keep the DAS system homogeneous. The reasons for this choice were to allow easy exchange of software and to stimulate cooperation between ASCI researchers. Both the hardware and the software of DAS are homogeneous: each node has the same processor and runs the same operating system. Also, the local area network within all clusters is the same. The only variations in the system are the amount of local memory and network interface memory (SRAM) in each node and the

[2] The ASCI research school is unrelated to, and came into existence before, the Accelerated Strategic Computing Initiative.

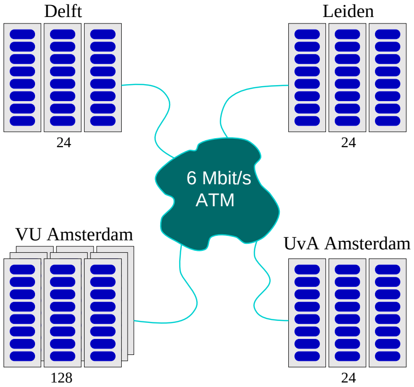

Figure 1: The wide-area DAS system.

number of nodes in each cluster. Three clusters have 24 nodes; the cluster at the VU initially contained 64 nodes, but was expanded to 128 nodes in May 1998.

We selected the 200 MHz Pentium Pro as processor for the DAS nodes, at the time of purchase the fastest Intel CPU available. The choice for the local network was based on research on an earlier cluster computer, comparing the performance of parallel applications on Fast Ethernet, ATM, and Myrinet [29]. Myrinet was selected as local network, because it was by far the fastest of the networks considered. Myrinet is a switch-based network using wormhole routing. Each machine contains an interface card with the LANai 4.1 programmable network interface processor. The interface cards are connected through switches. The Myrinet switches are connected in a ring topology for the 24-node clusters and in a 4 by 8 torus for the 128-node cluster. Myrinet is used as fast user-level interconnect. The nodes in each cluster are also connected by a partially switched Fast Ethernet, which is used for operating system traffic.

Since DAS is homogeneous, it runs a single operating system. We initially chose BSD/OS (from BSDI) as OS, because it is a stable system with commercial support. The increasing popularity of Linux both worldwide and within the ASCI school made us switch to Linux (RedHat 5.2) in early 1999.

The clusters were assembled by Parsytec. Figure 2 shows the 128-node cluster of the Vrije Universiteit. Each node is a compact module rather than a desktop PC or minitower, but it contains a standard PC motherboard. The clusters were delivered in June 1997.

The clusters are connected by wide-area networks in two different ways:

- using the National Research Network infrastructure (best effort network) and the LANs of the universities
- using an ATM based Virtual Private Network (Quality of Service network)

Figure 2: The 128-node DAS cluster at the Vrije Universiteit.

This setup allowed us to compare a dedicated fixed Quality of Service network with a best effort network. The best effort network consists generally of 100 Mbit/s Ethernet connections to a local infrastructure; each university typically had a 34 Mbit/s connection to the Dutch backbone, later increased to 155 Mbit/s. The ATM connections are all 6 Mbit/s constant bitrate permanent virtual circuits. The round trip times on the ATM connections have hardly any variation and typically are around 4 ms. On the best effort network the traffic always has to pass about 4 routers, which cause a millisecond delay each. Measurements show that the round trip times vary by about an order of magnitude due to the other internet traffic. Attainable throughput on the ATM network is also constant. On the best effort network, the potential throughput is much higher, but during daytime congestion typically gives throughputs of about 1-2 Mbit/s. This problem improved later during the project.

For most research projects, we thus use the dedicated 6 Mbit ATM links. The clusters are connected by these links using a fully-connected graph topology, so there is a link between every pair of clusters.

3 Research on systems software

DAS is used for various research projects on systems software, including low-level communication protocols, languages, and schedulers.

3.1 Low-level communication software

High-speed networks like Myrinet can obtain communication speeds close to those of supercomputers, but realizing this potential is a challenging problem. There are many intricate design issues for low-level network interface (NI) protocols [8]. We have designed and implemented a network interface protocol for Myrinet, called LFC [9, 10]. LFC is both efficient and provides the right functionality for higher-level programming systems. The LFC software runs partly on the host and partly on the embedded LANai processor of the Myrinet network interface card. An interesting feature of LFC is its spanning tree broadcast protocol, which is implemented on the NIs. By forwarding broadcast messages on the NI rather than on the host, fewer interactions are needed between the NI and the host, thus speeding up broadcasts substantially.

We have also developed a higher-level communication library, called Panda [5], which supports asynchronous point-to-point communication, remote procedure calls, totally-ordered broadcast, and multithreading. On Myrinet, Panda is implemented on top of LFC. The LFC and Panda libraries have been used for a variety of programming systems, including MPI, PVM, Java, Orca, Jackal, and CRL.

3.2 Languages and programming systems

Various languages and libraries have been studied using DAS. Part of this work focuses on local clusters, but several programming environments also have been implemented on the entire wide-area DAS system.

Manta is an implementation of Java designed for high-performance computing. Manta uses a static (native) compiler rather than an interpreter or JIT (just-in-time compiler), to allow more aggressive optimizations. Manta's implementation of Remote Method Invocation (RMI) is far more efficient than that in other Java systems. On Myrinet, Manta obtains a null-latency for RMIs of 37 μsec, while the JDK 1.1 obtains a latency of more than 1200 μsec [31]. This dramatic performance improvement was obtained by generating specialized serialization routines during compile-time, by reimplementing the RMI protocol itself, and by using LFC and Panda instead of TCP/IP. Manta uses its own RMI protocol, but also has the functionality to interoperate with other Java Virtual Machines. To handle polymorphic RMIs [53], Manta is able to accept a Java class file (bytecode) from a JVM, compile it dynamically to a binary format, and link the result into the running application program.

We have also developed a fine-grained Distributed Shared Memory system for Java, called Jackal [52]. Jackal allows multithreaded (shared-memory) Java programs to be run on distributed-memory systems, such as a DAS cluster. It implements a software cache-coherence protocol that manages regions. A region is a contiguous block of memory that contains either a single object or a fixed-size partition of an array. Jackal caches regions and invalidates the cached copies at synchronization points (the start and end of a `synchronized` statement in Java). Jackal uses local and global mark-and-sweep garbage collection algorithms that are able to deal with replicated objects and partitioned arrays. Jackal has been implemented on DAS on top of LFC.

Spar/Java is a data and task parallel programming language for semi-automatic parallel programming, in particular for the programming of array-based applications [47]. Apart from a few minor modifications, the language is a superset of Java. This provides Spar/Java with a modern, solid language as basis, and makes it accessible to a large group of users. Spar/Java extends Java with constructs for parallel programming, extensive support for array manipulation, and a number of other powerful language features. It has a flexible annotation language for specifying data and task mappings at any level of detail [46]. Alternatively, compile-time or run-time schedulers can do (part of) the scheduling. Spar/Java runs on the DAS using MPI and Panda.

Orca is an object-based distributed shared memory system. Its runtime system dynamically replicates and migrates shared objects, using heuristic information from the compiler. Orca has been implemented on top of Panda and LFC. An extensive performance analysis of Orca was performed on DAS, including a comparison with the TreadMarks page-based DSM and the CRL region-based DSM [5]. Also, a data-parallel extension to Orca has been designed and implemented, resulting in a language with mixed task and data parallelism. This extended language and its performance on DAS are described in [20].

In the ESPRIT project PREPARE, an HPF compiler has been developed with an advanced and efficient parallelization engine for regular array assignments [14, 39]. The PREPARE HPF compiler has been ported to the DAS and uses the CoSy compilation framework in combination with the MPI message passing library.

In another ESPRIT project, called Dynamite[3], we have developed an environment for the dynamic migration of tasks in a PVM program [21, 22, 44]. A version of this code is now available for SUN OS 5.5.1, SUN OS 5.6 and Linux/i386 2.0 and 2.2 (libc5 and glibc 2.0). DAS is being used for developing and testing the Linux version of this code. The aim of this work is to develop an environment for Linux, supporting dynamical task migration for PVM and MPI, and to make this environment available to the research community. Dynamite is minimally intrusive in the sense that it does not require modifications in the user's program and is implemented entirely in user space and thus does not require modifications to the kernel. The Dynamite system includes: a modified version of the Linux ELF dynamic loader, which does checkpointing and restarting of tasks; a modified version of PVM, supporting the transparent migration of tasks; monitoring programs for the system load and the PVM system; a dynamic task scheduler; and optionally, a GUI can be added that guides the user through the necessary steps to set up the environment and to start up a program.

Several programming systems have also been implemented on multiple clusters of the wide-area DAS system. Both Orca and Manta have been implemented on wide-area DAS and have been used to study the

[3] See http://www.hoise.com/dynamite

performance of wide-area parallel applications [35, 45]. In addition, we have developed a new MPI (Message Passing Interface) library for wide-area systems, called MagPIe [27]. MagPIe optimizes MPI's collective communication operations and takes the hierarchical structure of wide-area systems into account. With MagPIe, most collective operations require only a single wide-area latency. For example, an MPI broadcast message is performed by sending it in parallel over the different wide-area links and then forwarding it within each cluster. Existing MPI implementations that are not aware of the wide-area structure often forward a message over multiple wide-area links (thus taking multiple wide-area latencies) or even send the same information multiple times over the same wide-area link. On DAS, MagPIe outperforms MPICH by a factor of 2-8.

3.3 Schedulers

Another topic we are investigating with the DAS is the scheduling of parallel programs across multiple DAS clusters. Our current DAS scheduler (`prun`) only operates on single clusters, so for multi-cluster scheduling we need a mechanism for co-allocation of processors in different clusters at the same time. We are currently investigating the use of Globus with its support for co-allocation [13] for this purpose. So far, we have implemented a simple interface between Globus and `prun`, and we have been able to submit and run two-cluster jobs through Globus. An important feature of `prun` that facilitates co-allocation is its support for reservations. We are planning to enhance the interface between the local schedulers and Globus, and if necessary the co-allocation component of Globus, so that more optimal global scheduling decisions can be made. Our first results on the performance of co-allocation in DAS-like systems can be found in [11].

4 Research on applications

We have used the DAS system for applications that run on a single cluster (Section 4.1) and for wide-area applications (Section 4.2). Also, we have studied Web-based applications (Section 4.3) and worldwide distributed applications (Section 4.4).

4.1 Parallel applications

DAS has been used for a large number of parallel applications, including discrete event simulation [33, 40], Lattice Gas - and Lattice Boltzmann Simulations [15, 24, 25], parallel imaging [28], image searching [12], datamining, N-body simulation [42], game tree search [38], simulation of branch-prediction mechanisms, ray tracing, molecular dynamics, and quantum chemistry [19]. We discuss some of these applications in more detail below.

The PILE project is to design a parallel programming model and environment for time-constraint image processing applications [28]. The programming model is based on the analysis of typical solutions employed by users from the image processing community. The PILE system is built around a software library containing a set of abstract data types and associated operations executing in a data parallel fashion. As implementation vehicles on the DAS, MPI, CRL, and Spar/Java are being investigated.

Another application area is image databases. Visual concept recognition [12] algorithms typically require the solution of computationally intensive sub-problems such as correlation and the optimal linear principal component transforms. We have designed efficient algorithms for distributing the recognition problem across high bandwidth, distributed computing networks. This has led not only to new algorithms for parallelizing prevalent principal component transforms, but also to novel techniques for segmenting images and video for real time applications.

Another project investigates hardware usage for branch predictors. Branch predictors are used in most processors to keep the instruction pipeline filled. As a first step, we want to investigate the effects of many different (flavors of) algorithms. For this purpose, a database was built which currently holds about 8000 SQL-searchable records. Each record contains a detailed description of the state of a branch predictor after the run of a trace. These traces were created on the DAS machine using a MIPS-simulator, which simulates six different Spec95 benchmarks. The database was also built on the DAS machine, which took about 20 hours using 24 nodes. The individual nodes were used as stand-alone computers. Each node ran a copy of a branch-predictor simulator and worked on its own data-set. The main advantage of using the DAS machine for this project is that it provides a transparent multi-computer platform, which has proven to build the required database in a fraction of the time needed by a single computer. With the resulting database the investigation of the next steps in this project has been started.

We also use DAS for experimental research on the Time Warp Discrete Event Simulation method. The availability of fast, low-latency communication is an important asset here. We have made extensive use of the DAS to run Time Warp simulations for studying the influence of the application dynamics on the execution behavior of the Time Warp simulation kernel. The application under study is an Ising spin system. The results clearly show the influence of the Ising spin dynamics on the Time Warp execution behavior in terms of rollback length and frequency, and turnaround times. The results indicate the need for adaptive optimism control mechanisms. Successful experiments showed the versatility of optimism control. First results are obtained for describing and measuring self organization in parallel asynchronous Cellular Automata with

Time Warp optimistic scheduling. It was shown that different scaling laws exist for rollback lengths with varying Time Warp windows. These results were experimentally validated for stochastic spin dynamics systems. The work is described in more detail in [40].

Another project studies N-body simulations. The numerical integration of gravitational N-body problems in its most basic formulation requires $O(N^2)$ operations per time step and $O(N)$ time steps to study the evolution of the system over a dynamically interesting period of time. Realistic system sizes range from a few thousand (open clusters) through 10^6 (globular clusters) to 10^{12} (large galaxies). There are several options to speed up such calculations: use a parallel computer system; use fast, special purpose, massively parallel hardware, such as the GRAPE-4 system of Makino [32]; avoid recomputing slowly varying forces too frequently (i.e. use individual time-steps); or carefully combine groups of particles in computing the forces on somewhat distant particles (this leads to the well-known hierarchical methods). Each of these techniques has distinct advantages and drawbacks. In our research we strive to find optimal mixes of the above approaches for various classes of problems. We have attached two GRAPE-4 boards, which were kindly made available by Jun Makino, to two separate DAS nodes at the University of Amsterdam. The system is used both by the Astronomy Department for actual N-body simulations, and by the Section Computational Science to model the behavior of such hybrid computer systems and to guide the development of more advanced approaches to N-body simulations, combining some or all of the above techniques.

The implementation of a hierarchical algorithm on a parallel computer and the control of the resulting numerical errors are important components of our research. The methodologies to be developed have a wider applicability than astrophysical N-body problems alone. Experiments have been performed on two astronomical N-body codes that have been instrumented to obtain performance data for individual software components and the GRAPE. One of the codes was adapted to run as a parallel code on the DAS with GRAPE. A simulation model for the resulting hybrid architecture has been developed that reproduces the actual performance of the system quite accurately, so we will use this model for performance analysis and prediction for similar hybrid systems.

We also study particle models with highly constrained dynamics. These Lattice Gas (LGA) - and Lattice Boltzmann models (LBM) originated as mesoscopic particle models that can mimic hydrodynamics. We use these models to study fluctuations in fluids and to study flow and transport in disordered media, such as random fiber networks and random close packings of spheres. In all cases the computational demands are huge. Typical LGA runs require 50 hours compute time on 4 to 8 DAS nodes and are compute bounded. Typical LBM runs simulate flow on very large grids (256^3 to 512^3) and are usually bounded by the available memory in the parallel machine. Large production runs are typically executed on 8 to 16 DAS nodes.

Our parallel Lattice Boltzmann code was originally developed on a Cray T3E [24, 25] under MPI. Because of the inherent data locality of the LBM iteration, parallelization was straightforward. However, to guarantee good load balancing we use Orthogonal Recursive Bisection to obtain a good partitioning of the computational box. The code was ported to several systems (DAS, Parsytec CC, IBM SP2). Currently, we investigate flow and transport in random close packings of spheres, using the DAS.

We developed a generic parallel simulation environment for thermal 2-dimensional LGA [15]. The decomposition of the rectangular computational grid is obtained by a strip wise partitioning. An important part of the simulation is continuous Fourier transformations of density fluctuations after each LGA iteration. As a parallel FFT we use the very efficient public domain package FFTW (http://www.fftw.org/) which executes without any adaptations on the DAS.

Another project involves parallel ray tracing. The ray tracer we use is based on Radiance and uses PVM (on top of Panda) for inter-processor communication. The port to DAS was achieved with the aim to compare performance results on different platforms, including the DAS, a Parsytec CC, and various clusters of workstations. The algorithm consists of a demand-driven part for those tasks which require either a large amount of data or data which is difficult to predict in advance. This leads to a basic, but unbalanced workload. In order to achieve proper efficiencies, demand driven tasks are created where possible. These include tasks which are relatively compute intensive and require a small amount of data. Demand driven tasks are then used to balance the workload. An overview of the algorithm, including results, are given in [36].

In the Multigame project, we have developed an environment for distributed game-tree search. A programmer describes the legal moves of a game in the Multigame language and the compiler generates a move generator for that game. The move generator is linked with a runtime system that contains parallel search engines and heuristics, resulting in a parallel program. Using the Multigame system, we developed a new search algorithm for single-agent search, called Transposition Driven Search, which obtains nearly perfect speedups up to 128 DAS nodes [38].

Several interactive applications are being studied that use parallelism to obtain real-time responses, for example for steering simulations. Examples are simulation of the chaotic behavior of lasers or of the motion of magnetic vortices in disordered superconductors, real-time analysis of experimental data (e.g., determining the current patterns in flat conductors from magneto-optical imaging [54]), and post-processing experimental results. Examples of the latter category are reconstruction of 3D images of teeth from local CT-data and reconstruction of the sea-bottom structure from acoustical data. As many problems use linear-algebraic operations or Fast Fourier Transforms, excellent speed-ups can be obtained.

One of the largest applications ported to DAS is the Amsterdam Density Functional (ADF) program [19] of the Theoretical Chemistry section of the Vrije Universiteit. ADF is a quantumchemical program. It uses density functional theory to calculate the electronic structure of molecules, which can be used for studying various chemical problems. ADF has been implemented on DAS on top of the MPI/Panda/LFC layers. The obtained speed-up strongly depends on the chosen accuracy parameters, the molecule, and the calculated property. The best speed-up measured so far was 70 on 90 CPUs for a calculation on 30 water molecules. Current work focuses on eliminating the remaining sequential bottlenecks and improving the load balancing scheme.

We have also compared the application performance of the DAS clusters with that of a 4-processor SGI Origin200 system (at the University of Utrecht). This study mainly uses the EuroBen benchmark, which indicates performance for technical/scientific computations. The single-node observed peak performance of the Pentium Pro nodes of DAS is 3–5 times lower than that of the Origin200 nodes, mainly because the O200 nodes are super scalar (they have more independently schedulable floating-point operations per cycle). We compared the communication performance of DAS and the O200 by running a simple ping-pong test written in Fortran 77/MPI. The MPI system used for DAS is a port of MPICH on top of Panda and LFC. The bandwidth within a local DAS cluster is about half of that of the O200. Finally, we measured the performance of a dense matrix-vector multiplication The DAS version scales well, but for large problem sizes the program runs out of its L2 cache, resulting in a performance decrease. The maximum speed obtained on DAS (1228 Mflop/s) is a factor of 6 lower than on the O200 (7233 Mflop/s).

4.2 Wide-area applications

One of the goals of the DAS project was to do research on distributed supercomputing applications, which solve very large problems using multiple parallel systems at different geographic locations. Most experiments in this area done so far (e.g., SETI@home and RSA-155) use very coarse-grained applications. In our work, we also investigate whether more medium-grained applications can be run efficiently on a wide-area system. The problem here, of course, is the relatively poor performance of the wide-area links. On DAS, for example, most programming systems obtain a null-latency over the local Myrinet network of about 30-40 μsec, whereas the wide-area ATM latency is several milliseconds. The throughput obtained for Myrinet typically is 30-60 Mbyte/sec and for the DAS ATM network it is about 0.5 Mbyte/sec. So, there is about two orders of magnitude difference in performance between the local and wide-area links.

Many researchers have therefore come to believe that it is impossible to run medium-grained applications on wide-area systems. Our experience, however, contradicts this expectation. The reason is that it is possible to exploit the *hierarchical structure* of systems like DAS. Most communication links in DAS are fast Myrinet links, and only few links are slow ATM links. We have discovered that many parallel algorithms can be optimized by taking this hierarchical structure into account [6, 35, 45]. The key is to avoid the overhead on the wide-area links, or to mask wide-area communication. Many applications can be made to run much faster on the wide-area DAS using well-known optimizations like message combining, hierarchical load balancing, and latency hiding. The speedups of the optimized programs often is close to those on a single cluster with the same total number of CPUs.

In addition to this distributed supercomputing type of application, interest is also increasing in using DAS for other types of computational-grid applications. An important issue is to harness the large computational and storage capabilities that are provided by such systems. A next step is to develop new types of application environments on top of these grids. Virtual laboratories are one form of such new application environments. They will, in the near future, have to allow an experimental scientist (either being a physicist, a biologist or an engineer) to do experiments or develop designs. A Virtual laboratory environment will consist of equipment (like a mass spectrometer or a DNA micro array) that can be remotely controlled and that will provide data sets that can be stored in the information management part of the laboratory. Interaction with the data can, among others, take place in virtual reality equipment like a CAVE. We have used DAS and another SMP-based cluster computer (Arches) to develop the distributed information management for such systems as well as to study the possibilities for more generic user oriented middle ware for such laboratories [23].

In another project (which is part of the Dutch Robocup initiative), we have developed an interactive and collaborative visualization system for multi-agent systems, in particular robot soccer. Our system allows human users in CAVEs at different geographic locations to participate in a virtual soccer match [37, 43]. The user in the CAVE can navigate over the soccer field and kick a virtual ball. In this way, the user interacts with a running (parallel) simulation program, which runs either on an SP2 or a DAS cluster. In general, many other scientific applications can benefit from such a form of high-level steering from a Virtual Reality environment. The wide-area networks of DAS are an interesting infrastructure for implementing distributed collaborative applications in general, as the link-latency has hardly any variation.

4.3 Web-based applications

We have studied Web caching using DAS and Arches, focusing mainly on cache replacement and coherence strategies. This study showed that the techniques currently used for Web caching are nevertheless very simple and are mostly derived from earlier work in computer architecture systems. The experiments show that these techniques are still quite efficient compared to some new techniques that have been proposed specially for

Web caching [1, 2, 3]. Since in Web caching both strong and weak document coherency are considered, we have performed experiments in which we studied the quality of the hits (good hits are performed on up-to-date cached documents). We have shown the existence of two categories of replacement strategies, performing the hits on recently requested documents or (mainly) on long term cached documents. The usage of the cached documents is quite different in both classes and the cache size has different impact on each category. We have compared the efficiency of strong and weak document coherency. The results show that with weak document coherency, between 10% and 26% of the forwarded documents were out-of-date, while the useless generated traffic remains quite high: 40% to 70% of the messages exchanged to check the state of the cached documents are useless. To study strong coherency, we used a typical method that uses the invalidation protocol. The results show that the cost paid for this can be quite high. On average 70% of the invalidation messages are useless and arrive at a cache server after the target document has been removed from the cache.

In another project, called ARCHIPEL [7], we study information service brokerage systems. In order to support the wide variety of application requirements, fundamental approaches to electronic commerce, Web-based clearing houses, distributed databases, and information service brokerage systems need to be developed. The ARCHIPEL system developed at the University of Amsterdam aims at both addressing and analyzing these complex needs, and providing the environment and system for their adequate support. The results achieved at this early stage of the project contain the identification of the challenging requirements for the ARCHIPEL support infrastructure, such as Web-based, cooperative and interoperable, high performance, support for node autonomy, preservation of information visibility (and information modification) rights, and the platform heterogeneity. So far, a comprehensive description of the ARCHIPEL infrastructure is provided that unites different characteristics of existing parallel, distributed, and federated database management systems within one uniform architecture model. Currently, different methodologies and technologies are being evaluated to fulfill the requirements. From this point of view, among others the XML technology, ODBC standard, rapidly improving Java based programming and distributed management tools are being evaluated.

4.4 Worldwide distributed applications

The goal of the Globe project [50, 51] is to design and build a prototype of a scalable infrastructure for future worldwide distributed applications. The infrastructure is designed to support up to a billion users, spread over the whole world. These users may own up to a trillion objects, some of them mobile. Many aspects of the system, such as replicating data over multiple locations and managing security should be automatic or at least easily configurable. Although Globe interworks with the World Wide Wide, it is intended to run native on the Internet, just as email, USENET news, and FTP do. DAS, with its 200 nodes at four locations, has been used as a testbed to test pieces of the Globe system. We hope that a first prototype of Globe will be available in early 2001.

The software technology underlying Globe is the distributed shared object. Each object has methods that its users can invoke to obtain services. An object may be replicated on multiple machines around the world and accessed locally from each one as if it were a local object. The key to scalability is that each object has its own policies with respect to replication, coherence, communication, security, etc., and these policies are encapsulated within the object. For example, an object providing financial services may require sequential consistency, whereas an object providing sports scores may have a much weaker consistency. It is this ability to tailor the various policies per object that makes Globe scalable because those objects with demanding requirements can have them without affecting objects that can live with weaker guarantees.

One of the aspects studied in detail is locating objects (files, mailboxes, Web pages, etc.) in such a large system. When an object wishes to be found, it registers with the location server, which tracks the location of all objects in a worldwide tree [48, 49]. The tree exploits locality, caching, and other techniques to make it scalable. Recent work has focused on handling mobile objects. Other work has looked at automating the decision about where to place replicas of objects to minimize bandwidth and delay [34]. The main conclusion here is that the ability to tailor the replication policy to each object's access patterns provides significant gains over a single replication policy for all objects and far better than having only a single copy of each object. We have also looked at security issues [30].

Three applications of Globe have been built. The first one, Globedoc [26], is used to produce a better Web on top of Globe. Globedoc allows one or more HTML pages plus some appropriate collection of icons, images, etc. to be packaged together into a single Globedoc and transmitted all at once, a vastly more efficient scheme than the current Web. Gateways to and from the Web have been constructed so Globedoc objects can be viewed with existing browsers. The second application is the Globe Distribution Network [4], an application to provide a scalable worldwide distribution scheme for complex free software packages. The third application is an instant-messaging service (called Loc8) built as a front end to the location service. This service allows users all over the world to contact each other, regardless whether they are mobile or not. Special attention has been paid to security. In contrast to the centralized approach of existing services, our Loc8 service is highly distributed and exploits locality as much as possible to attain scalability.

5 Conclusions

The Distributed ASCI Supercomputer (DAS) is a 200-node homogeneous wide-area cluster computer that is used for experimental research within the ASCI research school. Since the start of the project in June 1997, a large number of people have used the system for research on communication substrates, scheduling, programming languages and environments, and applications.

The cluster computers of DAS use Myrinet as fast user-level interconnect. Efficient communication software is the key issue to obtain high communication performance on modern networks like Myrinet. We designed an efficient low-level communication substrate for Myrinet, called LFC. LFC provides the right functionality to higher level layers (e.g., MPI, PVM, Java RMI, Orca), allowing them to obtain high communication performance. Most programming systems (even Java RMI) obtain null-latencies of 30-40 μsec and throughputs of 30-60 Mbyte/sec over Myrinet. We have therefore been able to successfully implement a wide variety of parallel applications on the DAS clusters, including many types of imaging applications and scientific simulations.

DAS also is an excellent vehicle for doing research on wide-area applications, because it is homogeneous and uses dedicated wide-area networks. The constant bandwidth and the low round trip times of the WANs make message passing between the clusters predictable. On heterogeneous computational grids [17], additional problems must be solved (e.g., due to differences in processors and networks). Our work on wide-area parallel applications on DAS shows that there is a much richer variety of applications than expected that can benefit from distributed supercomputing. The basic assumption we rely on is that the distributed system is structured hierarchically. This assumption fails for systems built from individual workstations at random locations on the globe, but it does hold for grids built from MPPs, clusters, or networks of workstations. We expect that future computational grids will indeed be structured in such a hierarchical way, exhibiting locality like DAS.

Acknowledgements

The DAS system is financed partially by the Netherlands Organization for Scientific Research (NWO) and by the board of the Vrije Universiteit. The system was built by Parsytec, Germany. The wide-area ATM networks are part of the Surfnet infrastructure. A large number of other people have contributed to the DAS project, including Egon Amade, Arno Bakker, Sanya Ben Hassen, Christopher Hänle, Philip Homburg, Jim van Keulen, Jussi Leiwo, Aske Plaat, Patrick Verkaik, Ivo van der Wijk (Vrije Universiteit), Gijs Nelemans, Gert Polletiek, (University of Amsterdam), Wil Denissen, Vincent Korstanje, Frits Kuijlman

(Delft University of Technology), and Erik Reinhard (University of Bristol). We thank Jun Makino for kindly providing us with two GRAPE-4 boards.

References

[1] A.Belloum and B. Hertzberger. Dealing with One-Time Documents in Web Caching. In *"EUROMICRO'98 Conference"*, Sweden, August 1998.

[2] A.Belloum and B. Hertzberger. Replacement Strategies in Web Caching. In *"ISIC/CIRA/ISAS'98 IEEE conference"*, Gaithersburg, Maryland, September 1998.

[3] A.Belloum, A.H.J Peddemors, and B. Hertzberger. JERA: A Scalable Web Server. In *"Proceedings of the PDPTA'98 conference"*, pages 167–174, Las Vegas, NV, 1998.

[4] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, I. van der Wijk, M. van Steen, and A.S. Tanenbaum. The Globe Distribution Network. In *Proc. 2000 USENIX Annual Conf. (FREENIX Track)*, pages 141–152, June 2000.

[5] H. Bal, R. Bhoedjang, R. Hofman, C. Jacobs, K. Langendoen, T. Rühl, and F. Kaashoek. Performance Evaluation of the Orca Shared Object System. *ACM Transactions on Computer Systems*, 16(1):1–40, February 1998.

[6] H.E. Bal, A. Plaat, M.G. Bakker, P. Dozy, and R.F.H. Hofman. Optimizing Parallel Applications for Wide-Area Clusters. In *International Parallel Processing Symposium*, pages 784–790, Orlando, FL, April 1998.

[7] A. Belloum, H. Muller, and B. Hertzberger. Scalable Federations of Web Caches. *submitted to the special issue on Web performance of the Journal of Performance Evaluation*, 1999.

[8] R.A.F. Bhoedjang, T. Rühl, and H.E. Bal. Design Issues for User-Level Network Interface Protocols for Myrinet. *IEEE Computer*, 31(11):53–60, November 1998.

[9] R.A.F. Bhoedjang, T. Rühl, and H.E. Bal. Efficient Multicast on Myrinet Using Link-Level Flow Control. In *Proc. of the 1998 Int. Conf. on Parallel Processing*, pages 381–390, Minneapolis, MN, August 1998.

[10] R.A.F. Bhoedjang, K. Verstoep, T. Rühl, H.E. Bal, and R.F.H. Hofman. Evaluating Design Alternatives for Reliable Communication on High-Speed Networks. In *Proc. 9th Int. Conference on Architectural*

Support for Programming Languages and Operating Systems (ASPLOS-9), Cambridge, MA, November 2000.

[11] A.I.D. Bucur and D.H.J. Epema. The Influence of the Structure and Sizes of Jobs on the Performance of Co-Allocation. In *Sixth Workshop on Job Scheduling Strategies for Parallel Processing (in conjunction with IPDPS2000)*, Lecture Notes in Computer Science, Cancun, Mexico, May 2000. Springer-Verlag, Berlin.

[12] J. Buijs and M. Lew. Learning Visual Concepts. In *ACM Multimedia'99*, Orlando, FL, November 1999.

[13] K. Czajkowski, I. Foster, and C. Kesselman. Resource Co-Allocation in Computational Grids. In *Proc. of the 8-th IEEE Int'l Symp. on High Performance Distributed Computing*, pages 219–228, Redondo Beach, CA, USA, July 1999.

[14] W.J.A. Denissen, V.J. Korstanje, and H.J. Sips. Integration of the HPF Data-parallel Model in the CoSy Compiler Framework. In *7th International Conference on Compilers for Parallel Computers (CPC'98)*, pages 141–158, Linkoping, Sweden, June 1998.

[15] D. Dubbeldam, A.G. Hoekstra, and P.M.A. Sloot. Computational Aspects of Multi-Species Lattice-Gas Automata. In P.M.A. Sloot, M. Bubak, A.G. Hoekstra, and L.O. Hertzberger, editors, *High-Performance Computing and Networking (HPCN Europe '99), Amsterdam, The Netherlands*, number 1593 in Lecture Notes in Computer Science, pages 339–349, Berlin, April 1999. Springer-Verlag.

[16] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Int. Journal of Supercomputer Applications*, 11(2):115–128, Summer 1997.

[17] I. Foster and C. Kesselman, editors. *The GRID: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998.

[18] A.S. Grimshaw and Wm. A. Wulf. The Legion Vision of a Worldwide Virtual Computer. *Comm. ACM*, 40(1):39–45, January 1997.

[19] C. Fonseca Guerra, J. G. Snijders, G. te Velde, and E. J. Baerends. Towards an Order-N DFT method. *Theor. Chem. Acc.*, 99:391–403, 1998.

[20] S. Ben Hassen, H.E. Bal, and C. Jacobs. A Task and Data Parallel Programming Language based on Shared Objects. *ACM. Trans. on Programming Languages and Systems*, 20(6):1131–1170, November 1998.

[21] K.A. Iskra, Z.W. Hendrikse, G.D. van Albada, B.J. Overeinder, and P.M.A. Sloot. Experiments with Migration of PVM Tasks. In *Research and Development for the Information Society Conference Proceedings (ISThmus 2000)*, pages 295–304, 2000.

[22] K.A. Iskra, F. van der Linden, Z.W. Hendrikse, B.J. Overeinder, G.D. van Albada, and P.M.A. Sloot. The implementation of Dynamite — an environment for migrating PVM tasks. *ACM OS Review (submitted)*, 2000.

[23] E. Kaletas, A.H.J. Peddemors, and H. Afsarmanesh. ARCHIPEL Cooperative Islands of Information. Internal report, University of Amsterdam, Amsterdam, The Netherlands, June 1999.

[24] D. Kandhai. *Large Scale Lattice-Boltzmann Simlations: Computational Methods and Applications.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1999.

[25] D. Kandhai, A. Koponen, A.G. Hoekstra, M. Kataja, J. Timonen, and P.M.A. Sloot. Lattice Boltzmann Hydrodynamics on Parallel Systems. *Computer Physics Communications*, 111:14–26, 1998.

[26] A.M. Kermarrec, I. Kuz, M. van Steen, and A.S. Tanenbaum. A Framework for Consistent, Replicated Web Objects. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS)*, May 1998.

[27] T. Kielmann, R.F.H. Hofman, H.E. Bal, A. Plaat, and R.A.F. Bhoedjang. MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 131–140, Atlanta, GA, May 1999.

[28] D. Koelma, P.P. Jonker, and H.J. Sips. A software architecture for application driven high performance image processing. *Parallel and Distributed Methods for Image Processing, Proceedings of SPIE*, 3166:340–351, July 1997.

[29] K. Langendoen, R. Hofman, and H. Bal. Challenging Applications on Fast Networks. In *HPCA-4 High-Performance Computer Architecture*, pages 125–137, Las Vegas, NV, February 1998.

[30] J. Leiwo, C. Hänle, P. Homburg, C. Gamage, and A.S. Tanenbaum. A security design for a wide-area distributed system. In *Proc. Second Int'l Conf. Information Security and Cryptography (ICISC'99), In LNCS 1878*, December 1999.

[31] J. Maassen, R. van Nieuwpoort, R. Veldema, H.E. Bal, and A. Plaat. An Efficient Implementation of Java's Remote Method Invocation. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 173–182, Atlanta, GA, May 1999.

[32] J. Makino, M. Taiji, T. Ebisuzaki, and D. Sugmimoto. GRAPE-4: A Massively-parallel Special-purpose Computer for Collisional N-body Simulation. *Astrophysical Journal* , (480):432–446, 1997.

[33] B.J. Overeinder, A. Schoneveld, , and P.M.A. Sloot. Self-Organized Criticality in Optimistic Simulation of Correlated Systems. *Submitted to Journal of Parallel and Distributed Computing*, 2000.

[34] G. Pierre, I. Kuz, M. van Steen, and A.S. Tanenbaum. Differentiated Strategies for Replicating Web Documents. In *Proc. 5th International Web Caching and Content Delivery Workshop*, May 2000.

[35] A. Plaat, H. Bal, and R. Hofman. Sensitivity of Parallel Applications to Large Differences in Bandwidth and Latency in Two-Layer Interconnects. In *Fifth International Symposium on High-Performance Computer Architecture*, pages 244–253, Orlando, FL, January 1999. IEEE CS.

[36] E. Reinhard, A. Chalmers, and F.W. Jansen. Hybrid scheduling for parallel rendering using coherent ray tasks. In J. Ahrens, A. Chalmers, and Han-Wei Shen, editors, *1999 IEEE Parallel Visualization and Graphics Symposium*, pages 21–28, October 1999.

[37] L. Renambot, H.E. Bal, D. Germans, and H.J.W. Spoelder. CAVEStudy: an Infrastructure for Computational Steering in Virtual Reality Environments. In *Ninth IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, PA, August 2000.

[38] J.W. Romein, A. Plaat, H.E. Bal, and J. Schaeffer. Transposition Table Driven Work Scheduling in Distributed Search. In *16th National Conference on Artificial Intelligence (AAAI)*, pages 725–731, Orlando, Florida, July 1999.

[39] H.J. Sips, W. Denissen, and C. van Reeuwijk. Analysis of Local Enumeration and Storage Schemes in HPF. *Parallel Computing*, 24:355–382, 1998.

[40] P.M.A. Sloot and B.J. Overeinder. Time Warped Automata: Parallel Discrete Event Simulation of Asynchronous CA's. In *Proceedings of the Third International Conference on Parallel Processing and Applied Mathematics*, number 1593 in Lecture Notes in Computer Science, pages 43–62. Springer-Verlag, Berlin, September 1999.

[41] L. Smarr and C.E. Catlett. Metacomputing. *Communications of the ACM*, 35(6):44–52, June 1992.

[42] P.F. Spinnato, G.D. van Albada, and P.M.A. Sloot. Performance Analysis of Parallel N-Body Codes. In *High-Performance Computing and Networking (HPCN Europe 2000)*, number 1823 in Lecture Notes in Computer Science, pages 249–260. Springer-Verlag, Berlin, May 2000.

[43] H.J.W. Spoelder, L. Renambot, D. Germans, H.E. Bal, and F.C.A. Groen. Man Multi-Agent Interaction in VR: a Case Study with RoboCup. In *IEEE Virtual Reality 2000 (poster)*, March 2000. The full paper is online at http://www.cs.vu.nl/~renambot/vr/.

[44] G.D. van Albada, J. Clinckemaillie, A.H.L. Emmen, J. Gehring, O. Heinz, F. van der Linden, B.J. Overeinder, A. Reinefeld, and P.M.A Sloot. Dynamite - Blasting Obstacles to Parallel Cluster Computing. In *High-Performance Computing and Networking (HPCN Europe '99)*, number 1593 in Lecture Notes in Computer Science, pages 300–310. Springer-Verlag, Berlin, April 1999.

[45] R. van Nieuwpoort, J. Maassen, H.E. Bal, T. Kielmann, and R. Veldema. Wide-Area Parallel Computing in Java. In *ACM 1999 Java Grande Conference*, pages 8–14, San Francisco, California, June 1999.

[46] C. van Reeuwijk, W.J.A. Denissen, F. Kuijlman, and H.J. Sips. Annotating Spar/Java for the Placements of Tasks and Data on Heterogeneous Parallel Systems. In *Proceedings CPC 2000*, Aussois, January 2000.

[47] C. van Reeuwijk, A.J.C. van Gemund, and H.J. Sips. Spar: a Programming Language for Semi-automatic Compilation of Parallel Programs. *Concurrency Practice and Experience*, 9(11):1193–1205, November 1997.

[48] M. van Steen, F.J. Hauck, G. Ballintijn, and A.S. Tanenbaum. Algorithmic Design of the Globe Wide-Area Location Service. *The Computer Journal*, 41(5):297–310, 1998.

[49] M. van Steen, F.J. Hauck, P. Homburg, , and A.S. Tanenbaum. Locating Objects in Wide-Area Systems. *IEEE Communications Magazine*, pages 104–109, jan 1998.

[50] M. van Steen, P. Homburg, and A.S. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, pages 70–78, January-March 1999.

[51] M. van Steen, A.S. Tanenbaum, I. Kuz, and H.J. Sips. A Scalable Middleware Solution for Advanced Wide-Area Web Services. *Distributed Systems Engineering*, 6(1):34–42, March 1999.

[52] R. Veldema, R.A.F. Bhoedjang, R.F.H. Hofman, C.J.H. Jacobs, and H.E. Bal. Jackal: A Compiler-Supported, Fine-Grained, Distributed Shared Memory Implementation of Java. Technical report, Vrije Universiteit Amsterdam, July 2000.

[53] J. Waldo. Remote Procedure Calls and Java Remote Method Invocation. *IEEE Concurrency*, pages 5–7, July–September 1998.

[54] R.J. Wijngaarden, H.J.W. Spoelder, R. Surdeanu, and R. Griessen. Determination of Two-dimensional Current Patterns in Flat Superconductors from Magneto-optical Measurements: An Efficient Inversion Scheme. *Phys.Rev.B*, (54):6742–6749, 1996.