OOTI students went Parallel: PARLE'94

ir. Jean Moonen, ir. Rob Gelderblom, ir. Bram Stappers, ir. Olaf Pigmans

A group of fifteen OOTI students decided to expand their knowledge of parallelism and took off to sultry Athens, where the PARLE'94 conference was held. A record of their findings is presented below.

Introduction

Every student at OOTI is offered the opportunity to attend a conference focusing on one of the areas covered by the post-masters programme. This year four interesting events came into view.

- ECOOP'94 on Object-Oriented Programming in Bologna, Italy
- Third International School and Symposium Formal Techniques in Real Time and Fault Tolerant Systems in Lubeck, Germany
- CAISE'94 on Advanced Information Systems Engineering in Utrecht, Netherlands
- PARLE'94 on Parallel Architectures and Languages in Athens, Greece

This article contains a report on the latter conference, which has been selected by fifteen students. As already has been stated, parallelism is not a new subject to an OOTI student. The curriculum of the programme contains a course in parallel programs and transputer networks covering subjects in the field of multicomputers, communicating processes, load balancing and routing, as well as some case studies. Furthermore, students are assumed to have a basic knowledge of the design of parallel programs as a result of their preceding education. If not, an additional course will be given during the OOTI programme to the students in question.

Given this information, it seems to be a legitimate step to attend a conference on the parallel processing subdomain of software technology. PARLE is the main scientific event in this area held in Europe. Since its origination in 1987 as an initiative coming from the ESPRIT I programme, it has become a major event which has assumed high international reputation. Furthermore, the organized activities promise the event to be attractive for both novices and experts.

The 1994 edition of PARLE was organized by the Computer Technology Institute (CTI) at Patras, Greece under auspices of the Council of European Informatics Societies (CEPIS). During the five-day event, starting at July 4th, about 120 participants could select from a wide range of activities. The first day a tutorial programme unveiling the state-of-the-art of parallel processing could be attended. The following three days were packed with plenary, poster, vendor, and regular sessions. The fifth and last day some satellite events were scheduled.

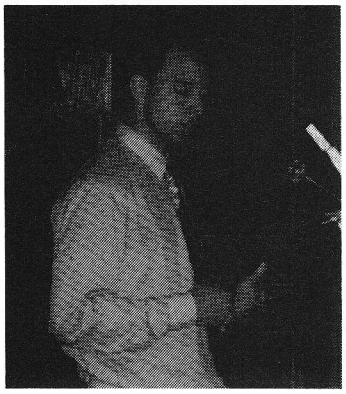
It would be impossible to discuss these few pages all topics encountered. Instead, we have selected those subjects that struck us most and give an extended description of them, starting with the submission of an OOTI student. The fact that an OOTI student has been invited by the organizing committee to present work done for his Master's degree during one of the regular sessions may be considered an indicator for the quality of the OOTI programme of Eindhoven University of Technology.

ILIAS, a new sequential programming language for parallel matrix computations

A hot topic in parallel computing nowadays is the development of a parallelizing compiler. With a parallelizing compiler, existing sequential applications can be ported to a parallel architecture by re-compilation. The design of a parallelizing compiler for a distributed memory parallel architecture is, however, a difficult and challenging task requiring sophisticated compiler techniques. It is not clear yet if there will be an efficiently parallelizing compiler for a general class of programs in the foreseeable future.

For the session on *Scientific Computing*, L.D.J.C. Loyens (Koninklijke/Shell-Laboratorium, Amsterdam), and J.R. Moonen (OOTI student) presented

a paper on the design and implementation of ILIAS, a sequential programming language for parallel matrix computations. Their approach is successful, mainly for two reasons. First, they restricted themselves to matrix computations, which are relatively easy to parallelize. Second, their language is not a conventional (low-level) language like C or FORTRAN, but contains high-level operators, which makes life easier for the programmer as well as for the compiler. Instead of having to extract parallelism from loops, using advanced data-dependency techniques, a programmer can often avoid writing loops, by using high-level matrix operators. The ILIAS system simply implements these high-level operators by parallel algorithms on distributed data objects. The price to be paid is the rewriting of existing C- or FORTRAN-applications in ILIAS.



Ir. Jean Moonen during his presentation at PARLE'94 (photo: Peter Kemp)

The ILIAS system consists of a sequential compiler (running on a SUN workstation) and a parallel interpreter (running on a torus network of transputers). A correct source program is first translated by the compiler into an intermediate code. This intermediate code is subsequently sent to the transputer network and executed by the parallel interpreter. The interpreter is built on top of a parallel library of basic linear algebra computations, and operates on matrices and vectors that are distributed over the processor network. Parallelism is achieved by letting each processor operate on its own matrix parts as much as possible (data parallelism).

The feasibility and scalability of the ILIAS system is demonstrated by timing results for two example ILIAS programs, LU decomposition and Strassen's matrix multiplication on transputer networks of up to 400 processors. These timing results also indicate that the execution time of an ILIAS program is about 50% slower than an equivalent explicit parallel program. The programming effort, however, is dramatically reduced because parallel matrix computations are specified in ILIAS in a high-level and sequential way.

Modeling photochemical pollution using parallel and distributed computing platforms

For the session on *Applications*, D. Abramson (Griffith University, Australia), M. Cope (Victorian Environment Protection Authority, Australia), and R. McKenzie (Royal Melbourne Institute of Technology, Australia) had submitted a paper on the simulation of air pollution using parallel computing. Scientists are increasingly turning to numerical simulation in order to investigate and model complex environmental systems. Numerical simulation has enormous advantages over laboratory or field experimentation because it has the potential to allow a much larger parameter space to be considered. Numerical simulation is being used for simulating the formation of photochemical air pollution (smog) in industrialized cities. However, computational hardware demands can be great. Abramson et.al. have investigated the computational resources that are required in order to achieve a realistic number of results in a timely manner. Furthermore, they describe the parallelization and distribution of programs that have been used as part of an air pollution study being conducted in Melbourne, Australia.

In order to be able to perform a full simulation of air pollution one has to model the photochemical pollution. Oxidants, such as ozone, are generated as a result of the chemical interaction between various precursors such as oxides of nitrogen and other reactive organics in the presence of ultra-violet radiation. Because the wind and the atmospheric stability are highly related to the smog concentration, the complex air flows have to be modeled to predict the photochemical smog production. On the other hand, the photochemical airshed has to be modeled, which involves the transport and production of chemicals that occurs in the atmosphere.

The models mentioned above are used in two different computational experiments. One of these generates the ozone distribution for one particular emissions data base and weather scenario, and

is used for event description and model verification. The other generates multiple scenarios, and is used to evaluate various control strategies. The first kind of experiment takes 16 hours on a typical workstation. Here a parallel solution might be appreciated. The second kind of experiment takes 8 hours per scenario. One is usually interested in, say, 50 scenarios, so this takes 400 hours. So, here also supercomputing technology is required to produce timely results.

The problems with the earlier mentioned response-times can be solved by using two different techniques, viz. parallelization and distribution. First the airshed model and the wind field model were parallelized so they performed well on parallel hardware. Subsequently, the airshed model was distributed over a heterogeneous array of computers in various locations, some with shared file systems. Knowledge of the hardware architecture was used to optimize the data distribution. Finally, a tool was built to control the distributed computation.

Having successfully conducted an air pollution study in Melbourne, the authors however remain sceptic about the results being used by the local government.

Bus-based parallel computers: a viable way for massive parallelism

In the session *Miscellaneous*, A. Ferreira (Ecole Normale Superieure de Lyon, France), A. Goldman vel Lejbman, and S.W. Song (University of Sao Paulo, Brazil) state that a bus-based massive parallel system is a promising alternative for point-to-point interconnection networks. In most distributed memory MIMD multiprocessors, processors are connected by a point-to-point interconnection network. Since inter-processor communication frequently constitutes serious bottlenecks, several architectures were proposed that enhance point-to-point topologies with the help of multiple bus systems so as to improve the communication efficiency. Ferreira et.al. study the global communication on parallel architectures where the communication means are constituted solely by buses. Attention is focused on the bus-based versions of well-used point-to-point linear and grid interconnection networks. Using theoretical concepts in order to model inter-processor communication in such networks, an extremely efficient algorithm for gossiping (all-to-all or total exchange communication) can be developed.

In the case of massively parallel architectures (i.e., systems with well over thousand processors) the interconnection between processors becomes a problem. For example, take a system of 8100 processors. If one were to use hypercubes this would amount to about 50,000 wires. A torus configuration would still give rise to some 16,000 wires. Using buses, full connectivity would be achieved with 180 buses in a 90 by 90 grid.

Bus-based architectures can use the power of bus technologies, providing a way to interconnect much more processors in a simple and efficient manner. Furthermore, the multiple bus architecture has the ability to expand. Without the underlying point-to-point network, such architectures are feasible and suitable for VLSI implementation, being able to interconnect a lot more processors, while keeping the diameter of the network small.

Bus interconnection networks (BINs) are best modelled by so-called hypergraphs. For BINs it is derived that the most efficient communication patterns are obtained when every processor performs a broadcast. This communication scheme is also known as *gossiping*.

The audience remained rather cynical about the physical feasibility of such bus-based systems and about whether these kind of large bus configurations could perform in acceptable time when a lot of communications needed to be performed. It was felt that state-of-the-art hardware was not yet suited to accommodate such large numbers of processors and to communicate over them at a sufficiently fast rate, ignoring for the time being the likely requirement of synchronization and how this should be achieved.

Improving the execution of the dependent AND-parallel Prolog DDAS

In the session *Language Implementation* two presentations were given about Prolog issues. The first was given by Kish Shen (University of Bristol, UK) and dealt with AND-parallelism. AND-parallelism allows several conjunctive goals to be computed in parallel. For correct execution this goals must be independent. For AND-parallel Prolog an execution scheme exists: DDAS.

The execution of DDAS can be improved by classifying the goals into groups. The goals in a group have no variable dependencies with goals in other groups and a member goal of a group must have direct or indirect variable dependencies with all other members of the group.

Now both the forward and backward executions can be improved.

- More precise discarding of work (in backward execution): only work done on AND-goals in the same group needs to be discarded.
- More precise determination of producer status (a producer of an unbound variable is the one that binds it, i.e. the leftmost clause in the goal that still contains this variable): the producer status is passed to the next goal in the same group.
- A more intelligent backtracking scheme: backtracking can take place inside each group without reference to other groups.

OR-parallel Prolog on distributed memory systems

The second presentation about Prolog was given by Peter Kacsuk (KFKI-MSZKI Research Institute, Hungary) and focused on OR-parallelism. For the OR-parallel implementation of Prolog on a distributed memory system two computational models can be considered.

1. The stack-copying model
2. The recomputation model

The idea is that each processor will handle a part of the execution tree. In the stack-copying model shared branches are copied to every processor, which costs a lot of memory. In the recomputation model every processor recomputes the shared branches, which costs a lot of time.

A more optimal model is suggested: when a processor is idle, it requests a path. The processor with the highest workload sends the path to one of his free branches (the top-most branch) to the requesting processor. The requesting processor compares the path to its own branch stack, and recomputes only the parts that are different.

Conclusion

In this article an overview has been given of the most interesting topics presented during the PARLE'94 conference. For the OOTI students, visiting this event was an enriching experience, not only regarding the contents of the presentations and lectures, but also w.r.t. the atmosphere in which such an event takes place.

However, although the overall quality of the conference was fairly reasonable, some of the presentations were barely understandable because of language problems and questionable presentational techniques. Furthermore, some speakers failed to point out the relevance and background of their research. This resulted in lectures that were inaccessible to non-experts on those particular subjects.

Except for those inconveniences, the topics presented were interesting to the OOTI audience. Their foreknowledge enabled them to understand most of the topics quite well and allowed them to discuss with other participants of PARLE'94.

As a final conclusion, we could state that participating in an international conference is a valuable experience in several ways. Therefore, it may be considered an important addition to the OOTI programme. □

Reference

PARLE'94 Parallel Architectures and Languages Europe, Lecture Notes in Computer Science, Springer-Verlag, 1994, Vol. 817.

Ir. Bram Stappers, ir. Rob Gelderblom, ir. Olaf Pigmans, and ir. Jean Moonen (clockwise, starting upper left) are students of the post-masters programme Software Technology. Three of them are members of XOOTIC. *(Photo: Eindhoven University of Technology, Stafgroep Reproduktie en Fotografie)*