22
SEMPA: Software Engineering Methods for Parallel Scientific Applications

P. Luksch, U. Maier, S. Rathmayer, M. Weidmann
Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR-TUM)
Institut für Informatik, Technische Universität München
D-80290 München
e-mail: {luksch, maier, maiers, weidmann}@informatik-tu.muenchen.de
WWW: http://wwwbode.informatik.tu-muenchen.de/
Tel.: (089)2105-8164; Fax: (089)2105-8232

Abstract
SEMPA is an interdisciplinary project that brings together researchers from computer science, mechanical engineering and numerical mathematics. Its central objective is to develop new software engineering (SWE) methods for (distributed memory) parallel scientific computing. SEMPA is being funded by the BMBF*.

1 MOTIVATION

In many applications, fluid simulations are required because experiments are either impossible (such as in the case of climate modeling) or are too expensive. Today, the main factor that limits the use of simulation is run-time. Only parallel processing together with efficient numerical algorithms can achieve the performance that is necessary to enable more wide-spread use of simulation. Providing the necessary computational power will make simulations feasible in many areas where they would require unrealistic run-times today. In mechanical engineering, productivity can be considerably increased if flow simulations, which today have to be run as batch jobs overnight, could be run interactively from a CAD program.

Parallel processing has developed successfully in the research area over the last years. Now, as there are standardized message passing interfaces, such as PVM [GBD+94], MPI [MPI94] etc., portable software can be developed for a wide range of hardware platforms – from (heterogeneous) networks of workstations (NOWs) to high-end massively parallel systems (MPPs). Since even small companies usually have a number of workstations connected by a local area network (LAN), developing parallel software on a commercial basis is becoming an attractive option.

However, experience has shown that software development for parallel systems still is much less productive than writing sequential programs. One reason for this is that there are no adequate tools for designing and analyzing

*Federal Department of Education, Research and Technology parallel software. In addition, there are no software engineering (SWE) methods that address the problems related to parallelism such as synchronization issues, deadlocks and non-determinism. Finally, there is only very little support for the software engineer who is faced with the problem of understanding and existing program in order to parallelize it for execution on a distributed memory multiprocessor.

2 PARTNERS

LRR-TUM (Lehrstuhl für Rechnertechnik und Rechnerorganisation, Institut für Informatik, Technische Universität München). LRR-TUM is in charge of project management. Our research focuses on

- multiprocessor architectures
- tools for designing and analyzing parallel programs
- parallel and distributed applications
- distributed shared memory systems.

Advanced Scientific Computing GmbH (ASC) , Holzkirchen. ASC is developing and marketing the CFD simulation package **TASCflow** which solves the Navier-Stokes equations in 3d space. **TASCflow** is used in many companies and universities for simulating flows in a wide range of applications [TUG95].

GENIAS Software GmbH, Neutraubling near Regensburg. The company markets a number of software packages for NOWs and MPPs. They have developed the batch queuing system **CODINE** on NOWs, which will be the basis for the resource to be developed in SEMPA.

Institut für Computeranwendungen (ICA III), Universität Stuttgart. ICA's research is focused on robust multigrid methods for a wide range of problems including computational fluid dynamics, flow in porous media and computational mechanics. They have developed the software tool-box UG, which simplifies the adaptive solution of partial differential equations on unstructured meshes in two and three dimensions.

3 OBJECTIVES

Software Engineering Methods. In parallel scientific computing, software engineers usually are faced with an existing program or with existing modules, typically written in FORTRAN 77, which they are expected to parallelize for execution on a distributed memory multiprocessors (NOW or MPP). Therefore the focus of SWE is on the following topics:

- Analysis of complex software systems.
- Approaches to Parallelization that are specific to certain classes of scientific applications
- Standards for documentation and program development
- Portability: cover a wide range of hardware platforms ranging from (low-end) NOWs to high performance MPPs.
- Modularity and Re-usability.
- Concurrent Software Engineering: coordinate the work of programmers from different disciplines and institutions

Parallelization of TASCflow. The software package solves the Navier-Stokes equation in 3d space on unstructured grids using a finite volume discretization and an algebraic multi-grid solver. The program is written in FORTRAN 77 and has about 113,000 lines of code.

Load Balancing and Resource Management. A resource manager is being developed, which basically is a batch

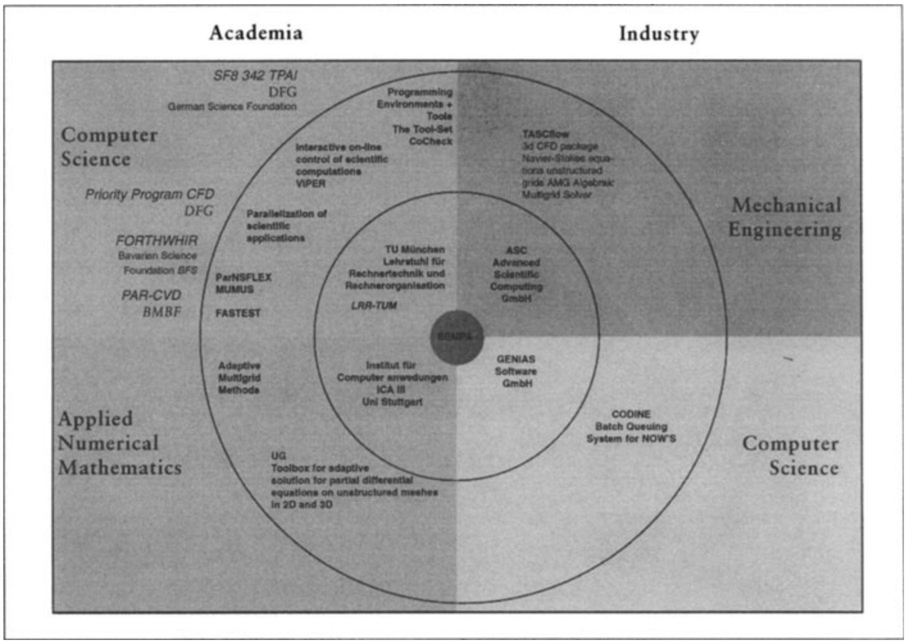
Academia
Industry
SF8 342 TPAI
DFG
German Science Foundation
Computer Science
Priority Program CFD
DFG
FORTHWHIR
Bavarian Science Foundation BFS
PAR-CVD
BMBF
Applied Numerical Mathematics
Programming Environments + Tools
The Tool-Set
CoCheck
Interactive on-line control of scientific computations
VIPER
Parallelization of scientific applications
ParNSFLEX
MUMUS
FASTEST
Adaptive Multigrid Methods
UG
Toolbox for adaptive solution for partial differential equations on unstructured meshes in 2D and 3D
TU München
Lehrstuhl für Rechnertechnik und Rechnerorganisation
LRR-TUM
Institut für Computer anwedungen
ICA III
Uni Stuttgart
SEMPA
TASCflow
3d CFD package
Navier-Stokes equations unstructured grids AMG Algebraic Multigrid Solver
ASC
Advanced Scientific Computing GmbH
GENIAS
Software GmbH
CODINE
Batch Queuing System for NOW'S
Mechanical Engineering
Computer Science

Figure 1 partners involved in the project

queuing system for parallel applications running on NOWs. The individual processes of the application are dynamically assigned to available processors (i.e. workstations). The resource manager will support load balancing by providing appropriate resource usage information and a mechanism to migrate processes from one workstation to another.

The result of SEMPA will be

- a collection of SWE methods that have been approved in practice,
- a prototypical implementation of the parallel version of **TASCflow**,
- a prototype of a resource and load manager for batch execution of parallel applications.

Upon completion of the research project, our industrial partners intend to develop further the prototypes of the parallel CFD package and the resource manager towards products that can be marketed commercially.

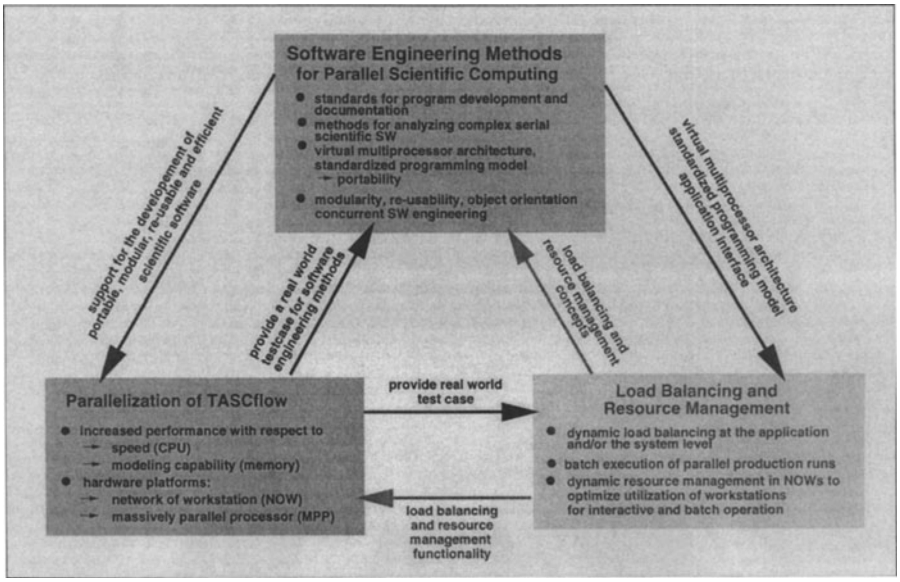
Software Engineering Methods for Parallel Scientific Computing
● standards for program development and documentation
● methods for analyzing complex serial scientific SW
● virtual multiprocessor architecture, standardized programming model → portability
● modularity, re-usability, object orientation concurrent SW engineering
support for the developement of portable, modular, re-usable and efficient scientific software
provide a real world testcase for software engineering methods
load balancing and resource management concepts
virtual multiprocessor architecture standardized programming model application interface
Parallelization of TASCflow
● increased performance with respect to
→ speed (CPU)
→ modeling capability (memory)
● hardware platforms:
→ network of workstation (NOW)
→ massively parallel processor (MPP)
provide real world test case
load balancing and resource management functionality
Load Balancing and Resource Management
● dynamic load balancing at the application and/or the system level
● batch execution of parallel production runs
● dynamic resource management in NOWs to optimize utilization of workstations for interactive and batch operation

Figure 2 project objectives and their interactions

4 PROGRESS REPORT

The project has started in April, 1995. Up-to-date information about progress as well as project reports and publications related to SEMPA are available via WWW (URL http://wwwbode.informatik.tu-muenchen.de/parallelrechner/applications/sempa/). In the subsequent section, we summarize the results achieved so far.

4.1 Analyzing the sequential program

The first step in parallelizing **TASCflow** has been to acquire the necessary understanding of the algorithms it uses and their implementation. ASC and LRR-TUM have been organizing a series of meetings, covering the following topics (in that sequence):

1. basics of CFD, i.e. the governing equations and their physical interpretation, discretization methods, and numerical methods for solving the system of linear equations that results from discretization.
2. a global overview of the code structure and the main data structures.
3. a more detailed review of **TASCflow**'s main modules, stepping through each module subroutine by subroutine.

Each meeting started with a presentation by ASC, which was followed by a discussion. At LRR-TUM, we documented our view of what we had learned in a meeting in form of an internal report, which then was reviewed by ASC. This procedure has proved to be an efficient way for know-how transfer between groups from different disciplines, since it has helped to identify and fix sources of misconception very early and quickly improved our understanding of each other's terminology and point of view.

As a final step, a framework has to be set up that defines a standard for documenting the design of the sequential program from the computer science point of view.

4.2 A Concept for Parallelizing TASCflow

Based on the insight gained from analyzing the program structure, a parallelization concept has been defined and documented [Luk95]. SEMPA follows a two-level concept of parallelism.

On the top level the SPMD model is used. The sequential algorithm[†] is replicated in multiple processes each of which operates on a partition of the problem description. An additional master process is used for program set up and for doing I/O. Using parallel I/O systems, which are available for a number of platforms, is being considered, too.

Partitioning is done node-based, i.e. the nodes of the (unstructured) grid are divided into disjoint sets. We use a public domain graph partitioning package (MeTiS [MET95]) for assigning nodes to partitions.

Below the SPMD level of parallelism, parallelization is considered at the level of processing nodes. Each replicated worker of the SPMD model can be further parallelized into a number of concurrent threads (light-weight processes having access to shared memory). This second level of parallelism can make use of multiple CPUs per processing node as they are available in new MPPs or workstations.

4.3 Interactive and automatic Parallelization Tools

The parallelization of an existing program, especially if it is complex and has been developed by many engineers over a long time, is a quite difficult and error-prone task.

Research projects as well as commercial efforts during the last years have been dealing with this problem. Most available tools are source-code analyzers for FORTRAN 77 programs which parallelize according to the SPMD model. One of those tools has already been subject of investigation within SEMPA. It is the quite sophisticated interactive an automatic parallelization tool FORGE [Res95]. There the most significant loops are identified by either using profiling information, or checking the code for the deepest loop nestings. Once the loops have been chosen, the arrays referenced inside of them are partitioned and distributed according to the partitions. The parallel processes then run the same program but only on a subset of the partitioned data structures following the so-called *owner computes* rule.

The advantage of these tools is that the user can get a better understanding of the program that he is about to parallelize. He also is taken off the burden to explicitly program message passing code. On the other hand he anyhow has to have a good understanding of how message passing really works because the tools are not yet at a point where they can produce efficient code. Neither can they really work on very complex packages as for example TASCflow.

4.4 New Languages

Moving from FORTRAN 77 to newer programming languages meets the requirements of modern computer architectures, programming paradigms, and software engineering aspects. Fortran 90 for example offers not only more

[†] augmented by additional code for communication and synchronization

complex data structures and data encapsulation but also provides language constructs (array operations) for concurrent execution. The latest of all FORTRAN evolutions, High Performance Fortran, additionally has constructs for explicit data distribution as well as constructs for expressing concurrency.

FORTRAN 77 compilers produce fast object code and numerous numerical programming libraries are available due to its long time of existence. Object oriented design is still uncommon in scientific computing because the compilers (e.g. for C++) do not yet generate optimized code that is comparable to the one generated by FORTRAN 77 compilers. However, the fundamental ideas of object oriented programming – objects, class-hierarchies and polymorphism – are of great advantage to modern software engineering and can help to overcome the gap between the code development and its concept.

In SEMPA, we have decided that rewriting TASCflow as a whole in an object-oriented language is infeasible due to manpower restrictions. Instead, we have selected a module of reasonable size for implementation in Fortran 90, C++ (and possibly other languages) to demonstrate the integratability of object-oriented techniques to a scientific application, and to evaluate the appropriateness of these languages for our purposes.

REFERENCES

[GBD+94] *Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam.* "PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing". MIT Press (1994). www: http://www.netlib.org/pvm3/book/pvm-book.html.

[Luk95] *Peter Luksch.* A Concept for Parallelizing TASCflow. SEMPA-Report SEMPA-TUM-95-05, Technische Universität München, Institut für Informatik (September 1995). www: http://wwwbode.informatik.tu-muenchen.de/archiv/Projektberichte/SEMPA/ws-aug-95.ps.gz. draft version.

[MET95] "METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System". George Karypis and Vipin Kumar, University of Minnesota (1995).

[MPI94] MPI: A Message Passing Interface Standard. Technical report University of Tennessee, Knoxville, Message Passing Interface Forum (May 1994).

[Res95] *Applied Parallel Research.* "The FORGE Product Set". Applied Parallel Research Inc., 550 Main Street, Placerville, CA 95667 (February 1995).

[TUG95] 3rd TASCflow User Conference – Presentations. Tech. Report ASCG/TR-95-04, Advanced Scientific Computing GmbH, Aying (May 1995).