Towards High-Performance Symbolic Computing in MuPAD: Multi-Polynomial Quadratic Sieve using Macro Parallelism and Dynamic Modules

Metzner, Radimersky, Sorgatz, Wehmeier
{tom,maradim,andi,stefanw}@uni-paderborn.de

Department of Mathematics and Computer Science
University of Paderborn
Germany

June 15, 1998

Abstract

This article discusses the approach of developing MuPAD as an open and parallel *problem solving environment* (PSE) for mathematical and technical problems, including so-called *real world applications*. It describes the new implementation of *macro parallelism*, based on *dynamic modules*, which now covers three fields of parallel programming: message passing, network variables and work groups. Parallel algorithms including benchmarks and examples of using MuPAD as a PSE are demonstrated.[1]

Introduction

Interactive general purpose computer algebra systems (CAS) are very good to define mathematical descriptions of technical problems, i.e. algorithms, systems of equations etc. They can transform them, do several kinds of symbolic computations and also display solutions graphically. Alas, they are mostly not efficient enough to solve so-called *real world applications*, meaning large physical systems and applications from industry, in a reasonable amount of time.

Our approach to solve this discrepancy is an open interactive CAS, able to integrate special purpose algorithms (C/C++,...) using them like native elements of a high-level CAS programming language in a parallel working environment for distributed computation in heterogeneous networks. The talk discusses this approach introducing the MuPAD key technologies *dynamic modules*, *macro parallelism* and *domains* [1][5].

[1] This work was supported by the DFG Sonderforschungsbereich 376: *Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen.*

The New Macro Parallelism

Macro parallelism follows the *master-worker* paradigm and covers three fields of parallel programming: *queues* (message passing), *global variables* and *work groups*. One can specify which hosts with how many MuPAD *clusters* on them are part of it and can (re-)configure and (re-)start the heterogeneous network at run-time. Tasks and data are distributed via *queues*. Global variables make it easy to share information between all clusters. Writing to them is an atomic operation, meaning they can be used for synchronization (*locks/semaphores*). Several *work groups* may coexist, each consists of one *group master* and an arbitrary number of *workers*. Once a job is delivered to a master, job scheduling is done automatical. One can check the job status and fetch the result when available. Group jobs may generate new jobs recursively. While waiting on results, they may process further jobs instead of busy waiting or suspending themselves. This way, it is very easy to run *branch-and-bound* or highly recursive algorithms like Karatsuba.

Writing parallel algorithms, we first concentrated on typical problem classes like a parallel sieve algorithm for factoring integers (*MPQS*), a modular algorithm for computing gcds over algebraic number fields, Karatsuba's method for multiplying polynomials and a Gröbner Basis algorithm. See `http://www.mupad.de/PAPER/PARALLEL/` for details [2].

Dynamic Modules

This concept allows users to integrate C/C++ routines as well as whole software packages into MuPAD in an easy, flexible and very efficient way. Dynamic modules are special kinds of a machine code libraries. When they are loaded at run-time, they reside within the MuPAD kernel process environment and directly access internal MuPAD methods and data objects on a C/C++ level. They can be unloaded and are automatically re-*loaded on demand* if necessary. Unloading modules means to unlink their machine code from the kernel environment and to remove it from the main memory. See `http://www.mupad.de/PAPER/MODULES/` for details [3][4].

Towards High-Performance Symbolic Computing

The talk describes the new macro parallelism and its implementation as a dynamic module and discusses the approach of developing MuPAD as an open and parallel problem solving environment (PSE) utilizing the key technologies:

Macro Parallelism: for distributed computations using resources of a heterogeneous network including special computer hardware like massive parallel systems.

Dynamic Modules: interfacing high-performance C/C++ algorithms and special purpose tools for hotspots.

Domains: integrating high-performance C/C++ data structures from dynamic modules: e.g. a bit-vector representation for polynomials $F_2[X]$ only needs a basic set access functions to be used by the whole MuPAD library (polymorphism) and will speed up the user's applications.

Additional components of the MuPAD PSE are the *graphics system*, the *source level debugger* and *profiler* for MuPAD algorithms.

With this, MuPAD performs an open and parallel universal working environment for mathematical applications, able to interact with special purpose tools in a very efficient and flexible way.

An implementation of the multi-polynomial quadratic sieve

We conclude with one example for using *MuPAD* as a PSE.

In the *MuPAD* versions available to the public so far, tools for integer factorization did not go beyond the elliptic curve method. Although mostly written in the *MuPAD* language, our new hybrid implementation of the so–called single large prime variation of the `mpqs`-algorithm (see [6]) provides a significant speedup.

The polynomials to be sieved are distributed among the clusters of a network. The two most time– and memory–critical steps are done using dynamic modules written in `C`: one module provides a data type for sieve arrays, the other contains a linear system solver over $\mathbb{F}_2$.

First tests show that integers up to about 45 decimal digits can be factored within a few minutes.

References

[1] Drescher, K. Axioms, Categories and Domains. Tech. Rep., Uni-GH Paderborn, Sep 1996.

[2] Metzner, T., Radimersky, M., Sorgatz, A., and Wehmeier, S. Parallelism in MuPAD 1.4. Technical Report, SFB 376, Universität-Gesamthochschule Paderborn, Jun 1998.

[3] Sorgatz, A. Dynamic Modules - C++ Kernel Extensions. *mathPAD 7* (Sep 1997), 16–20.

[4] Sorgatz, A. *Dynamic Modules – User's Manual and Programming Guide for MuPAD 1.4.* University of Paderborn, May 1998. Preprint, ~240 pages, accompanying CD-ROM including MuPAD, module demos and source code, refer to `http://www.mupad.de/BIB/sorgatz98.shtml`.

[5] Benno Fuchssteiner et al. *MuPAD User's Manual - MuPAD Version 1.2.2,* first ed. John Wiley and sons, Chichester, New York, March 1996. ISBN 0-471-96716-5.

[6] Silverman, *The Multiple Polynomial Quadratic Sieve,* Math.of Comp. 48, 1987, p.329–339.