SETIT 2005
3rd International Conference: **S**ciences of **E**lectronic,
Technologies of **I**nformation and **T**elecommunications
March 27-31, 2005 – TUNISIA

Survey of Symbolic Computations on the Grid

Dana Petcu[*], Dorin Ţepeneu[‡], Marcin Paprzycki[†],

Tetsuya Mizutani[‡] and Tetsuo Ida[‡]

[*] *Institute e-Austria in Timişoara and Department of Computer Science,*
Western University of Timisoara,
B-dul Vasile Pârvan 4, Timişoara 300223, Romania
petcu@info.uvt.ro

[†]*Oklahoma State University, Tulsa, Oklahoma, USA, and SWPS, Warszawa, Poland*
marcin@cs.okstate.edu

[‡]*Department of Computer Science, University of Tsukuba,*
Tennoudai 1-1-1, Tsukuba 305-8573, Japan
dorinte@score.cs.tsukuba.ac.jp
mizutani@cs.tsukuba.ac.jp
ida@cs.tsukuba.ac.jp

Abstract:
Symbolic and algebraic computations are one of the fastest growing areas of scientific computing. In this paper we present an overview of the state-of-the-art in symbolic and algebraic computations on parallel and distributed computers and on grids. We give some background information, including typical application areas, and then give a list of past and on-going projects involving symbolic computations. We also attempt at concisely summarizing our findings. This article is based on the chapter on symbolic computations on grid that will appear in 2005 the book "Engineering the Grid: status and perspective", Editors Beniamino di Martino, Jack Dongarra, Adolfy Hoisie, Laurence Yang, and Hans Zima, Nova Science Publishers, Inc..

Key words: Symbolic Computation, Grid, Parallel and Distributed Computing, Computer Algebra System

1 Introduction

There exist two basic approaches to computational solution of mathematical problems: numerical and symbolic. The numerical approach had an advantage of being capable of solving a substantially larger set of problems. However, the symbolic approach is systematically gaining more recognition as a viable tool for solving large-scale engineering problems. Symbolic solution of mathematical problems involves manipulations of symbolic objects, like logical or algebraic formulas, rules or programs. Unlike the numerical approach, one of the main goals of the symbolic approach is exactness. Typically, the final answer obtained through symbolic manipulations is either a rational number or a formula that represents an answer.

Developments in symbolic computing have been lagging relatively to numerical computing, mainly due to the lack of available computational resources: most importantly computer memory, but also processor power. Continuous growth in the capabilities of computer hardware led to an increasing interest in symbolic computation and resulted, among others things, in development of sophisticated Computer Algebra Systems (CASs). These systems allow users to directly study computational problems on the basis of their mathematical formulations and to focus on the problems themselves instead of spending time transforming the problems into forms that are

numerically solvable. As an effect, symbolic computations are being applied in a number of diverse disciplines such as, among others, pure and applied mathematics, physics, engineering and economics [24]. Symbolic computation is also becoming a basis for advanced applications in many areas of computer science, such as computer aided design or software development, VLSI design, geometric modeling and reasoning, robot programming etc. Finally, symbolic methods have also become popular in life sciences, in particular in studying human genome.

While the major purpose of the CAS is to manipulate formulas symbolically, many systems have substantially extended their capabilities. Nowadays, robust CASs offer other functionalities like numerical calculation, graphics, and simulations allowing a more comprehensive approach to problem solving. Furthermore, modern CASs are capable of solving very large problems. While, typically, CAS systems are utilized in an interactive mode, to solve large problems they can be also used in a "batch" mode and programmed using languages that are very close to common mathematical notation. Lists of existing CASs can be found in [8, 72, 73] and their facilities have been compared in [4, 76, 82].

As CASs become capable of solving large problems, they follow the course of development that has already been taken by numerical software: from sequential computers to parallel machines to distributed computing and finally to the grid. It is particularly the grid that has high potential as a discovery accelerator. Currently, its widespread adoption is still impeded by a number of problems, one of which is difficulty of developing and implementing grid-enabled programs. The aim of this paper is to present a comprehensive survey of the state-of-the-art of symbolic and algebraic computations involving parallel and distributed environments as well as the path leading toward grid-enabled CASs.

2 Parallel and distributed symbolic computations

Many users utilize Computer Algebra Systems as tools performing "small scale" mathematical calculations that would be tedious and error-prone when performed by hand. For them, CASs running on a single processor computer are quite satisfactory. However, some users employ CASs to solve large and very large problems. Here, the solution to the problem may involve, for instance, large scale symbolic computations requiring a significant amount of computational resources, or a combination of symbolic and numerical computations. These users often encounter the limitations of single-processor systems: processor speed and available memory. It is this class of users that could truly benefit from availability of parallel and distributed versions of CASs.

It is well known, that the two main reasons driving the development of parallel computers are: (a) ability to reduce the wall-clock time i.e. the user waiting time for the solution (problems that are processor bound), and (b) ability to solve problems that cannot fit into memory of a "workstation" (problems that are memory bound). An argument has been formulated, that for the CASs it is the latter restriction that is the driving force for parallelization [47]. The argument relies on the following observations. Computation with algebraic terms involves interaction between the amount of available memory and the amount of memory required by the algorithm at any stage of the computation; the input size of a problem may be small, but its memory use in intermediate stages of the computation may grow considerably. This phenomenon is known as an *intermediate expression swell* and has been observed since the earliest CASs. However, parallelization of CASs is not easy to achieve. Symbolic computations tend to have unpredictable data dependencies, irregular data access patterns and varying dependent parameters – all these make it difficult to predict patterns of memory (and processor) usage. Moreover, in the case of distributed memory computers each processor has a local address space, and therefore it is quite possible for one processor to exceed its available memory, while there is still memory available "globally".

In this context, let us briefly look at algorithms that play a significant role in large-scale symbolic and algebraic computations.

2.1 Parallel algorithms for symbolic computations

Multi-precision integer arithmetic is one of the most important fields in symbolic and algebraic computations. It appears, among others, in factorizations [6] or Gröbner basis computations [35]. For parallel arithmetic in finite fields there exists an implementation on a massive parallel processor [74]. Modular integer multiplication [16] and exponentiation [52] have been also implemented on parallel architectures. Systolic algorithms for integer arithmetic were discussed in [35]. Furthermore, systems for multivariate integer arithmetic on distributed memory machines [71] and on the Internet [6] have been developed. Parallel implementation of

the Karatsuba algorithm for multi-precision integer multiplication was reported in [36].

The second class of algorithms that utilize significant amount of computational resources are implementations of polynomial arithmetic. Two categories of algorithms that can clearly benefit from multiple resources in parallel processing are (a) algorithms that depend on identification of similar terms such as the polynomial addition, and (b) knowledge based algorithms such as the symbolic differentiation [47]. The greatest common divisor is also an important operation on both integers and polynomials. Parallel integer GCD algorithms based on the Euclidean algorithm have been developed in [37]. Other parallel algorithms applicable to this problem, such as those proposed in [51], use Fast Fourier Transforms (FFTs). In [78] two aspects of parallelism in symbolic computing were discussed: implementation of parallel programs used in factorization of polynomials and automatic derivation and generation of parallel codes for finite element analysis. The first aspect illustrates the use of parallel programming to speed up symbolic manipulation, while the second one shows how symbolic systems can help create parallel software for scientific computation.

The third class of algorithms that can benefit from extra memory and processing power availability in parallel systems are the Cramer's rule and the Gaussian elimination with back substitution used to solve sparse systems of linear equations [47]. Parallel computers with distributed memory can provide the memory required for large determinant calculations by Cramer's rule which is impractical on single-processor computers. In [47] several methods for calculation of the determinant were compared: the Gaussian elimination method, the Bareiss recurrence formula (for large dense systems), and the classical minor expansion (for large sparse systems) and their benefits assessed.

Finally, the Gröbner basis algorithm has to be mentioned. Here, the size of the computation and the irregular data structures required make parallelization an attractive option for improving the algorithm performance. Several parallel implementations of the algorithm have been developed. That proposed in [3] consists of parameterized work distribution on shared memory architecture. In [9] application level threads are used on a distributed memory system. Implementations of the Gröbner basis algorithm on distributed memory systems have been reported in [47, 68, 70].

While memory-bound algorithms are clearly the most important driving force for the development of parallel CASs, there is one more reason that becomes more important with every new release of each of the major CAS. As indicated above, CASs increase their utility not only through adding new symbolic capabilities, but also through adding new functionalities, such as: simulation environments, visualization, numerical modules etc. In this way, modern CASs become less of a computational engine and more of a problem solving environment: the CAS is seen as an interface to a number of computational kernels that are used depending on the user needs. This change poses a need for addressing the parallelization of processor bound tasks. As examples of such tasks we can list: rendering of images for an animation illustrating the evolution of a system, multivariable simulation illustrating the solution space of an optimization problem or numerical solution of partial differential equations that are intractable through symbolic computations. In these cases, all of the typical problems involved in parallelization of numerical computations appear also in the context of parallel or distributed CAS. Overall, all the above listed examples illustrate the growing need for support of parallel and distributed computing within CASs.

The design and implementation of a robust and scalable CAS relies on the same principles as those applied in other large systems (e.g. modularity, abstraction), but there are also some specific development problems (identified, for instance, in [47]) that play an important role, when development of a parallel CAS is concerned: special treatment of object representation, domain specification, intermediate expression swell, algorithmic dependence on irregular data which are difficult to be dynamically partitioned, complexity of some of ensuing algebraic computations limiting ability to estimate resource requirements.

2.2 Parallel and distributed CASs – state-of-the-art

It is a well known fact that developing completely new parallel or distributed systems, although efficient, in most cases is rather difficult. Moreover, usually only a few parallel algorithms within such a system are fully implemented and tested, making the resulting artifact too limited for practical uses.

An alternative approach is to add parallelism to the existing software. While based on different requirements and targeting competing parallel architectures, several systems for parallel computer algebra have been developed in this way by following developmental strategies identified in [47]:

- develop CASs for shared memory architecture;
- develop computer algebra hardware;
- add parallel primitives for communication and cooperation to existing CASs;
- build distributed memory systems based on standard communication middleware; build distributed systems for loosely coupled machines or across the Internet.

We next present a summary of existing CASs and other symbolic computations systems or tools that have been extended towards parallel and distributed computations. We first discuss three most popular CASs: Maple [39], Matlab [43] and Mathematica [42] and subsequently present a combined list of smaller-scale or niche projects.

2.2.1 Maple. There exist a large number of efforts to extend Maple to parallel and distributed environments and a comprehensive review can be found in [75]. Here, we present a few selected examples to illustrate the most important approaches taken by various research groups.

A message passing interface to port Maple to the Intel Paragon was presented in [5]: a master-slave approach to distributed scheduling was used to maintain a single node access for interactive use of Maple. A parallel version of Maple running on a network of workstations was reported in [80]: it is a message-passing system with primitives spawn and kill for creating and terminating processes, and procedures send, receive and reply for communication. ||Maple|| [67] is a portable system for parallel symbolic computations built as an interface between the parallel programming language Strand and the sequential CAS Maple.

Distributed Maple [75] is a portable system for writing parallel programs in Maple, which allows the creation of concurrent tasks and have them executed by multiple Maple kernels running on separate networked computers. A configuration program written in Java starts and connects external computational kernels on separate machines and schedules concurrent tasks for execution on them. A package written in Maple implements an interface to the scheduler and provides a parallel programming model. The design principles behind PVMaple [65] are very similar to those of the Distributed Maple. Here, several independent Maple kernels residing on separate computers connected by a network cooperate to solve a given problem. Maple is wrapped into external software that manages execution of tasks. A special binary is responsible for the message exchanges between Maple processes, coordinates the interaction between Maple kernels via PVM daemons, and schedules tasks among nodes. A Maple library implements a set of parallel programming commands available within Maple itself and supports connections with the command messenger.

2.2.2 Matlab. More than 20 different versions of parallel Matlab have been developed by different groups of researchers and an overview of them was presented in [10]. They can be compared according to their process-communication and user interfaces. A significant number of parallel versions of Matlab make use of message-passing for interprocessor communication and provide message-passing interface to the user. Commands like send and receive are based on standard MPI/PVM libraries and utilized in DP-ToolBox [20], MPITB/PVMTB [53], and MultiMatlab [54]. Simple communication functions have been used in Matlab Parallelization Toolkit [45], ParMatlab [62] and PMI [68], while file I/O synchronization functions via a shared file system has been implemented in MatlabMPI [44]. A few parallel Matlabs are designed for shared-memory systems and provide shared-memory programming interfaces – in MATmarks [46], for example, commands are provided for shared variable declaration and process synchronization. Other versions of Matlab are designed to release the user from parallel details by overloading several existing Matlab functions with their parallel versions – for example, the Matlab*p [11]. Another approach is to use Matlab compilers, like Conlab [14] or Otter [60]. These compilers can automatically translate a Matlab program into a parallel program written in C or another language. It is worth noting that the Symbolic Math Toolbox provided within Matlab uses the Maple kernel for symbolic computations.

2.2.3 Mathematica. Parallel Computing Toolkit (PCT) [61] is an extension of Mathematica which allows communication between servers using rsh. A typical installation involves a master kernel, a license manager (to manage licenses and passwords), and one Mathematica kernel available on each computational node. The master kernel handles all input, output, and scheduling, and can be controlled interactively from front end or in batch mode. Parallel computation has four stages: extension load, launch of computational servers, execution of commands and stopping remote

servers. ParallelEvaluate is used for parallel evaluation of multiple expressions, ParallelMap for application of a function to several remote data objects, and ParallelTable for building tables. Variables and functions can be exported to remote servers using ExportEnviroment. The extension Parallel`VirtualShared` and the declaration SharedVariables simulate shared variables, but the access of each such variable is achieved through the network.

Based on the PCT, gridMathematica [30] was constructed as a parallel computing solution for dedicated clusters facilitating parallel computing within Mathematica, and requiring only TCP/IP connectivity. Since it is focused on management of clusters of heterogeneous machines, a better name for this environment would be clusterMathematica.

Distributed Mathematica [64], similar to Distributed Maple, is a system for writing parallel programs in Mathematica allowing to create concurrent tasks executed by Mathematica kernels running on networked computers.

2.2.4 Special libraries and systems. Threads for parallel symbolic computations on a shared-memory computers were used in parallelizing the SAC-2 CAS, resulting in the ParSac-2 parallel system [38], written in C. It has been used to substantially improve performance of applications such as Gröbner basis computations [3], Karatsuba multiplication and FFT-based methods. Paclib [33] is a parallel extension of Saclib, a library of C programs for computer algebra derived from the SAC-2 CAS. Here, parallelization was achieved through addition of lightweight processes, resulting in a general parallel CAS. Communication between processes is achieved through access to shared data. Since shared memory limits scalability, very large problems could not have been solved. Givaro [28] is a more recent C++ library that supports parallel programming for arithmetic and algebraic computations with basic algebraic objects, such as vectors, matrices and univariate polynomials.

A special L-language designed for L-machines was provided for parallel programming of computer algebra algorithms in [7]. The L-machine consists of a reconfigurable assembly of processors, memory, a bus switch, and a sensor bit used for access rights to the shared memory and for synchronization.

A dedicated machine called FLATS [29] was constructed for large scale CAS applications. It was equipped with special hardware for arbitrary precision arithmetic and parallel hashing in addition to the instruction set for executing Lisp primitives directly by the hardware. In [41] Lisp was extended with a tool for automatic identification of concurrency: the system accepts a Lisp program, analyses it for available concurrency and generates a program for parallel execution on a multi-processor comprising of networked workstations. This data flow analyzer can be also utilized to analyze a complete Lisp-based CAS such as the Reduce [69] and identify areas that can be parallelized. A different Lisp extension for a distributed network of workstations was given in [32] where explicit concurrency primitives were provided. Another system called Star/MPI [15] is available for Gnu Common Lisp and GAP [23].

The first version of Cabal [58] was written as a small system for polynomial algebra. It used the PVM communication library for parallelization. A later development switched to the MPI [48]. The memory model has been extended and several packages for multi-precision integers, matrix algebra and Gröbner basis computations were added. Applications of Cabal to large sparse systems of linear equations and to parallel Gröbner basis computations have been described in [47].

MuPad [55], the multi-processing algebra data tool, is a CAS developed for shared memory parallel machines designed to be an efficient tool for fast access to large databases and to allow functional programming.

Form [22], used in quantum field theory computations, is a program for symbolic manipulation of algebraic expressions that was tuned to handle very large expressions, involving millions of terms, arising in its application area. A parallel Form prototype ParForm, based on MPI, was presented in [21]. The Form user does not have to know anything about the mechanisms behind the parallel version to run existing programs in parallel (no need to modify the sequential versions of the programs) as parallelization is facilitated internally within the ParForm environment.

The FoxBox system [18] provides client-server interfaces to several CASs running in distributed computing environments; distribute, wait, kill functions are compliant with the MPI. Finally, the DSC system [19] is designed for symbolic computation on heterogeneous network of workstations and across the Internet. It has a master scheduler to distribute tasks based on availability of

resources determined through some threshold conditions. DSC has been used in algebraic computing with large integers and sparse linear systems [17].

2.3 Web-enabled systems

In number theory there exist a number of successful Internet projects [34] aiming, among others, at finding large prime numbers, factoring large numbers, computing digits of π, finding collisions on known encryption algorithms etc. For Gröbner basis computations, the online system OGB [49] has been recently deployed.

A CAS web-wrapper component that can be used by multiple systems was reported in [81]. Here, shared-memory parallelism was used to speed up Gröbner basis computations. Furthermore, a new algorithmic development in Gröbner basis computations, the fractal Gröbner walk, was made available to various CASs in the form of a single specialized implementation.

MapleNet [40] is a software platform to enhance mathematics and related courses over the web. The client machine must be able to run Java applets. A publisher machine is responsible for creating and editing content of web pages and, when complete, uploading them to the server. To access web pages and the applets associated with them, users will connect to the server. It is also the server that manages concurrent Maple instances launched to serve client requests for mathematical computations and display services. Finally, the server facilitates additional services such as user authentication, logging information, and database access.

webMathematica [82] offers access to Mathematica applications through a web browser. Mathematica can be seen as the computational engine for the webMathematica sites. webMathematica uses standard Java technologies: Java Servlet and JavaServer Pages. It allows a site to deliver HTML pages that incorporate Mathematica commands. When a request is made for one of these pages, the Mathematica commands are evaluated and the computed result is inserted "back" into the page. Input can come from commands forms, applets, JavaScript, and web-enabled applications. It is also possible to send data files to a server for processing. Output can use different formats such as HTML, images, Mathematica notebooks, MathML, XML, PostScript, and PDF.

A framework for description and provision of web-based mathematical services was recently designed within the Monet project [50]. Its aim was to demonstrate the applicability of the semantic web to the world of mathematical software. The key to such a framework is the ability to discover services dynamically based on published descriptions which express both their mathematical and non-mathematical attributes. The discovery service and subsequent interaction were mediated by software agents capable of recognizing the criteria which should determine how particular problems are to be solved, and extracting them from the user's problem description. A wrapper for the symbolic solver was also designed to provide an environment that encapsulates CASs and expose their functionalities through symbolic services. A symbolic service runs as an independent web service and is reachable through its own unique URL. All symbolic services are registered to the symbolic server and managed by the wrapper tool. The following technologies were used for implementing the symbolic solving services: Java, Axis, Tomcat, SOAP, WSDL, JSP, MSDL. Maple was chosen for the computing engine in the initial implementation, and Axiom was used to demonstrate the ability to incorporate different computational engines without major changes.

3 Grid-enabled systems

There exist a number of grid-oriented projects that involve CASs. Even though some of these projects have just been initiated, we report their existence and goals for completeness of the overview of the field as well as to illustrate its liveliness.

Open source package NetSolve [1] was one of the earliest grid systems developed. Its initial motivation was focused on the usability, portability and availability of existing optimized software libraries for high-performance computing, particularly those for numerical linear algebra. NetSolve is a middleware between desktop systems equipped with simple APIs and the existing services supported by the grid architecture. One of the goals of NetSolve project is to create a system capable of integrating arbitrary computational resources. NetSolve APIs are available for Mathlab, Mathematica, and Octave [59]. Version 2.0, released in 2003, introduces GridSolve re-designed for interoperability with the grid.

GridSolve is a GridRPC [56] based client-server agent system that enables users to solve complex scientific problems remotely using distributed resources on the grid. When a user submits a problem

to the NetSolve agent, the agent searches the grid, chooses a set of suitable services and requests that the problem be solved. After the task is completed the NetSolve agent returns the solution to the user. Load balancing and retry for fault-tolerance are handled automatically by the system. Access to different grids is made possible through proxies. At the present time, proxies for Globus and Condor-G are available.

Ninf-G [57] is another GridRPC system implementing the Ninf system on top of the Globus Toolkit. In version 3, released in 2004, different Ninf client APIs were build, including one for Mathematica.

Grid-Elimino [83] is a recent Java-based computation system for grids based on Globus and Iamc [79]. It contains a master program that controls the slave servers, i.e. Elimino instances running as grid services. Elimino is a stand-alone symbolic computation system. The client should be written in Java by the user and must describe the tasks for each Elimino instance.

The Grid Enabled Numerical and Symbolic Services [26] project, Genss, was initiated in 2004 and follows the ideas formulated in the Monet [50] project. It intends to combine grid computing and mathematical web services using a common open agent-based framework. Thus far research was focused in two areas: (1) matchmaking techniques for advertisement and discovery of mathematical services, and (2) design and implementation of an ontology for symbolic problems.

The Geodise [27] system, implemented within the Matlab environment, is an engineering portal providing grid access to computational fluid dynamics and design optimization tools. Two different mechanisms are used to submit jobs to computing resources. The first one uses a web service interface to Condor [13]. The second one is implemented as a collection of Matlab functions. In the latter case, submission of jobs to Globus-enabled resources is achieved via Java CoG tools. The new Matlab functions allow users to run and control jobs on the grid, or to archive, query, and retrieve data. Special Matlab functions are used to notify the (mobile) user about the status of the job.

Another on-going project [12] based on Java CoG builds two sets of software tools to enable access to Globus grid resources from Matlab. The first one is a set of wrappers necessary to invoke the CoG batch files directly from the Matlab command line. The second one is a set of Java CoG libraries providing the integration of user codes.

Matlab*g [10] builds a parallel Matlab on a platform-independent grid. It exploits a client-server architecture based on the distributed shared memory model. Each server receives a work package, performs computations, and sends results back to the client. Two types of operations are supported: distributed matrix computations and a parallel for-loop. First implementation of Matlab*g was based on Alice [2], grid-computing middleware using JavaSpaces. A more recent implementation is based on the Globus Toolkit.

A grid extension of Matlab*p based on the mpich-G2 and the Globus Toolkit is reported in [63]. It has three components: a server connection manager handling communications, a matrix manager handling the task distribution and a package manager registering the available services.

The recent Maple2g package [66] allows the connection between Maple and computational grids based on the Globus Toolkit. It is a grid-enabling wrapper consisting of two parts, a CAS-dependent and a grid-dependent one: a library of Maple functions allowing the Maple user to interact with the grid middleware, and a package of Java classes, acting as an interface between the library and the grid environment. Using it, the user can augment Maple's facilities with external modules, in particular being able to explore computational grid facilities, to connect to a specific grid service, to use that grid service, and to translate obtained results back into the Maple interface. Moreover, multiple Maple kernels can cooperate within a grid in a similar way like they would in a cluster.

MathGridLink [77] is a software component designed to act as a middleware between Mathematica and the grid. MathGridLink allows both the development/ deployment of Mathematica computational services on the grid and the usage of any existing grid service from within Mathematica. MathGridLink is built on top of the Globus Toolkit 3 and it consists of three main components: a general purpose grid service client, a component used for grid service generation from Mathematica, and a specialized Mathematica kernel manager providing remote access to grid services implemented in Mathematica.

The Grid-TLSE [31] project, started in 2003, aims to design an expert site for users who are searching for

sparse matrix solvers. The administrator interface, called Weaver is used to define, to deploy, and to exploit services over the grid. The web interface, called Websolve, allows a web browser to submit computational requests to a grid by using third-party software such as Matlab, Octave, Scilab, NetSolve and Diet.

Finally, Gemlca [25] is a recent solution to deploy a legacy code application as a grid service without modifying the code. The Gemlca front-end, described in WSDL, offers grid services to deploy, query, submit, check the status of, and get the results back from computational jobs. In order to access a legacy code program, the user executes the Gemlca grid service client that creates a legacy code instance with the help of a legacy code factory. Following this, the system submits the job to the compute server through Globus Toolkit.

As can be seen from the above presented summary of the state-of-the-art in parallel and distributed as well as grid-enabled CASs, this area is brimming with activities. In particular, very large body of research is devoted to facilitating symbolic computations on the grid.

4 Conclusions

We have tried to discuss recent developments in symbolic computations, in particular, computer algebra systems. We are particularly interested in initiatives leading to porting CASs to parallel and distributed computers as well and making them Web and grid enabled. We have presented a rather extensive list of past, present and future projects attempting to reach these goals. Our most important findings can be summarized as follows:

- a growing interest in symbolic computations and computer algebra systems can be observed;
- application of CASs to solution of large problems often demands application of parallel and/or distributed computers;
- porting CASs to parallel and distributed computers is not a trivial task;
- number of existing and newly started projects indicates fast growing interest in porting CASs to the Web and to the grids in particular;
- real world applications of CASs on grids are in very early phases and efficiency of proposed tools will have to be further investigated and improved;
- while most of project involving CASs and grids is in very early stages, existing tools are mature enough to allow for experimental work to be initiated and it is this type of work that is required to lay ground for the future developments.

This article is based on the chapter "Symbolic Computation on Grid" in the book "Engineering the Grid: status and perspective", Editors Beniamino di Martino, Jack Dongarra, Adolfy Hoisie, Laurence Yang, and Hans Zima, Nova Science Publishers, Inc., which is to appear in 2005.

References

[1] S. Agrawal, J. Dongarra, K. Seymour and S. Vadhiyar, NetSolve: *past, present, and future - A look at a grid enabled server*, in Making the Global Infrastructure a Reality, Wiley Publishing, Berman, F., Fox, G., Hey, A. eds., 2003, pp. 613-622, http://icl.cs.utk.edu/news_pub/submissions/netsolve-ppf.pdf.

[2] ALiCE, http://www.comp.nus.edu.sg/~teoym/alice.htm.

[3] B. Amrhein, O. Gloor and W. Küchlin, *A case study of multithreaded Gröbner basis completion*. In Procs. ISSAC96, ACM Press (1996), http://www-sr.informatik.unituebingen.de/projects/pareqs/issac96.ps .

[4] L. Bernardin, *A review of symbolic solvers* (1996), http://www.inf.ethz.ch/personal/bernardi/solve/review_A4.ps.

[5] L. Bernardin, *Maple on a massively parallel, distributed memory machine*, in Procs. PASCO'97, M. Hitz and E. Kaltofen, eds., ACM Press (1997), pp. 217–222.

[6] R. P. Brent, *Some parallel algorithms for integer factorisation,* in Procs. Euro-Par'99, P. Amestoy et al, eds., LNCS 1685, Springer (1999), http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pd/rpb193sp.pdf.

[7] B. Buchberger, *The L-machine: An attempt at parallel hardware for symbolic computation*, in Procs. AAECC-3, LNCS 229 (1986), Springer-Verlag, pp.333–347.

[8] Cain, http://research.mupad.de/CAIN/.

[9] S. Chakrabarti and K. Yelick, *Implementing an irregular application on distributed memory multiprocessor*, in Procs. 4th ACM SIGPLAN POPP'93, ACM Press (1993), pp. 169-178,

http://www.cs.berkley.edu/~yelick/soumen/grobner-ppopp93.ps.

[10] Y. Chen and S. Fong Tan, Matlab*g: *A grid-based parallel* Matlab, SMA, NUS, Singapore (2002), http://ntu-cg.ntu.edu.sg/Grid_competition/report/grid-9.pdf.

[11] R. Choy, Matlab*p *2.0*: *Interactive supercomputing made practical*. MS Thesis, EECS, MIT (2002), http://www.cs.ucsb.edu/~gilbert/cs290iSpr2003/ChoyThesis.ps.

[12] CoG Kit Matlab Page, http://www-unix.globus.org/cog/matlab/index.php.

[13] Condor, http://www.cs.wisc.edu/condor/.

[14] Conlab, http://www.cs.umu.se/research/conlab/.

[15] G. Cooperman, Star/MPI: *binding a parallel library to interactive symbolic algebra systems*, in Procs. ISSAC'95, ACM Press (1995), pp. 126–132.

[16] M. Diab, *Systolic architectures for multiplication over finite fields GF2m*, in Procs. 8th Internat. Conference Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, S. Sakata ed., LNCS 508, Springer (1990), pp. 329–340.

[17] A. Diaz, M. Hitz, E. Kaltofen, A. Lobo and T. Valente, *Process scheduling in DSC and the large sparse linear systems challenge*, Journal of Symbolic Computation, 19:1-3 (1995), pp. 269–282, DISCO'93 Springer LNCS 722, pp. 66–80, ftp://ftp.cs.rpi.edu/pub/kaltofen/DSC/DISCO93/jsc.ps, http://www.math.unm.edu/ACA/1995/Proceedings/Sessions.html.

[18] A. Diaz and E. Kaltofen , FoxBox: *A system for manipulating symbolic objects in black box representation*, in Procs. ISSAC'98, O. Gloor ed., ACM Press (1998), pp. 30–37, http://www4.ncsu.edu/~kaltofen/bibliography/98/DiKa98.ps.gz.

[19] A. Diaz, E. Kaltofen, K. Schmitz and T. Valente, DSC: *A system for distributed symbolic computation*, in Procs. ISSAC '1991, S. M. Watt ed., ACM Press (1991), pp. 323–332, http://www4.ncsu.edu/eos/users/k/kaltofen/bibliography/91/DKSV91.ps.gz.

[20] DP-Toolbox, http://www-at.e-technik.uni-rostock.de/dp/.

[21] D. Fliegner, A. Retey and J.A.M. Vermaseren, *Parallelizing the symbolic manipulation program FORM* (1999), http://arXiv.org/abs/hep-ph/9906426.

[22] Form, http://www.nikhef.nl/~form/.

[23] GAP, http://www-gap.dcs.st-and.ac.uk/~gap/.

[24] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra* (2003), 2nd edition Cambridge University Press.

[25] Gemlca, http://www.cpc.wmin.ac.uk/GEMLCA.

[26] Genss, http://genss.cs.bath.ac.uk/index.htm.

[27] Geodise, http://www.geodise.org/.

[28] Givaro, http://www-lmc.imag.fr/Logiciels/givaro/.

[29] E. Goto et al, *Design of a* Lisp *machine* FLATS, in Procs. ACM Symposium on Lisp and Functional Programming, Pittsburgh(1982), pp. 208–215.

[30] gridMathematica, http://www.wolfram.com/products/gridmathematica/.

[31] GridTLSE, http://www.enseeiht.fr/lima/tlse.

[32] R. H. Jr. Halstead, *Parallel symbolic computing: Languages, systems, and applications*, Procs. US/Japan Workshop, Cambridge, LNCS 748 (1992).

[33] H. Hong, A. Neubacher and W. Schreiner, *The design of the* SacLib/PacLib *Kernels*, Proceedings of DISCO'93, Gmunden, Austria, 1993, Springer, Berlin, Alfonso Miola ed., LNCS 722, http://www.risc.uni-linz.ac.at/people/schreine/papers/disco93.ps.gz.

[34] Internet-based Distributed Computing Projects, http://www.aspenleaf.com/distributed/ap-math.html.

[35] T. Jebelean, *Integer and rational arithmetic on MasPar*, in Design and Implementation of Symbolic Computation Systems, J. Calmet and C. Limongelli eds., LNCS 1128, Springer (1996), pp. 162–173.

[36] T. Jebelean, M. Dragan, D. Tepeneu and V. Negru, *Parallel algorithms for practical multiprecision arithmetic using the Karatsuba method*, Technical Report 00-42, RISC (2000), ftp://ftp.risc.uni-linz.ac.at/pub/techreports/2000/00-42.ps.gz.

[37] R. Kannan, G. Miller and L. Rudolph, *Sublinear parallel algorithm for computing the greatest common divisor of two integers*, SIAM Journal Comp., 16:1 (1987), pp.7–16.

[38] W. Küchlin, Parsac-2: *parallel computer algebra*, Computer Algebra in Science and Engineering, World Scientific, in J. Fleisher, J. Grabmeier, F. Hehl, W. Küchlin, eds. 1995, http://www-sr.informatik.uni-tuebingen.de/projects/pareqs/zif94.ps.

[39] Maple, http://www.maplesoft.com/.

[40] MapleNet, http://www.maplesoft.com/maplenet/.

[41] J. Marti and J. Fitch, The Bath concurrent LISP machine, in Procs. of the European Computer Algebra Conference on Computer Algebra, LNCS 162 (1983), pp. 78–90.

[42] Mathematica, http://www.wolfram.com/products/mathematica/.

[43] Matlab, http://www.mathworks.com/.

[44] MatlabMPI, http://arXiv.org/abs/astro-ph/0107406.

[45] Matlab Parallelization Toolkit, http://www.mathworks.com/matlabcentral/fileexchange / (utilities, distributed computing).

[46] MATmarks, http://polaris.cs.uiuc.edu/matmarks/.

[47] M. Matooane, *Parallel systems in symbolic and algebraic computation*, PhD Thesis (2001), University of Cambridge, Technical report No. 537, http://www.cl.cam.ac.uk/ TechReports/UCAM-CL-TR-537.pdf.

[48] M. Matooane and A. Norman, *A parallel symbolic computation environment: Structures and mechanics*, in Procs. Euro-Par '99, P. Amestoy et al, eds., LNCS 1685, Springer (1999), pp. 1492–1495.

[49] M. McGettrick, *OGB: Online Gröbner Bases* (2003), Technical report NUIG-IT-251103, National University of Ireland, Galway, http://grobner.it.nuigalway.ie.

[50] Monet, http://monet.nag.co.uk.

[51] P.L. Montgomery, *An FFT extension of the elliptic curve method of factorization*, PhD, Univ.California Los Angeles (1992), ftp://ftp.cwi.nl/pub/pmontgom/ucladissertation.ps.gz.

[52] M. Morii and Y. Takamatsu, *Exponentiation in finitefields using dual basis multiplier*, in Procs. 8th Internat. Conference Applied Algebra, Algebraic Algorithms and Error- Correcting Codes, S. Sakata ed., LNCS 508, Springer (1990), pp. 354–366.

[53] MPI/PVM Toolbox for Matlab (MPITB/PVMTB), http://atc.ugr.es/javier-bin/mpitb_eng and http://atc.ugr.es/javierbin/pvmtb_eng.

[54] MultiMatlab, http://www.cs.cornell.edu/Info/People/lnt/multimatlab.html.

[55] MuPAD, http://research.mupad.de/.

[56] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee and H. Casanova, *GridRPC: A Remote Procedure Call API for Grid Computing*, http://www.eece.unm.edu/~apm/ docs/APM_GridRPC_0702.pdf.

[57] Ninf, http://ninf.apgrid.org/.

[58] A. Norman and J. Fitch, Cabal: *Polynomial and power series algebra on a parallel computer*, in Procs. 2nd PASCO'97, M. Hitz and E. Kaltofen, eds., ACM press (1997), pp. 196–203.

[59] Octave, http://www.octave.org/.

[60] Otter, http://www.cs.orst.edu/~quinn/papers/hpdc7.ps.

[61] Parallel Computing Toolkit, http://www.wolfram.com/products/applications/parallel/ .

[62] ParMatlab, http://petrydpc.itm.mh.se/tools/.

[63] I. Patel and W. Mohiuddin, Matlab*p *on the grid* (2003), http://pompone.cs.ucsb.edu/~imran/matlabpg/.

[64] C. Pau and W. Schreiner, Distributed Mathematica (2000), http://www.risc.uni-linz.ac.at/software/distmath/.

[65] D. Petcu, PVMaple – *a distributed approach to cooperative work of* ‖Maple‖ *processes,* in Procs.EuroPVM/MPI'00, J. Dongarra et al. eds., LNCS 1908, Springer (2000), pp. 216–224.

[66] D. Petcu, D. Dubu, M. Paprzycki, *Extending Maple to the Grid: Design and Implementation*, in Procs. ISPDC'04, J.Morrison et al. eds., IEEE Computer Press, pp. 209-216

[67] R. Pirastu and K. Siegl, *Parallel computation and indefinite summation: A* ‖ Maple‖ *application for the rational case*, Journal of Symbolic Computation, 20:5-6 (1995), pp. 603–616, http://www.risc.uni-linz.ac.at/research/combinat/risc/publications/rpiratsu/CONPAR94.pdf.

[68] Pmi, ftp://ftp.mathworks.com/pub/contrib/v5/tools/PMI.

[69] Reduce, http://www.uni-koeln.de/REDUCE/.

[70] A. Reeves, *A parallel implementation of Buchberger's algorithm over* z_p *for* $p \geq 31991$, Journal of Symbolic Computation, 26:2 (1998), pp. 229–244.

[71] J. L. Roch, PAC: *Towards a parallel computer algebra co-processor*, Computer algebra and parallelism, ed. Della Dora and J.Fitch, Academic Press (1989), pp. 33–50.

[72] Sac, http://www.symbolicnet.org/systems/Systems.html.

[73] SAL, http://www.sai.msu.su/sal/A/1/index.shtml.

[74] E. Sibert, H.F. Mattson and P. Jackson, *Finite field arithmetic using the connection machine*, in Procs. 2nd Intern. Workshop Computer Algebra and Parallelism R.E. Zippel, ed., LNCS 584, Springer (1990), pp. 51–61.

[75] W. Schreiner, C. Mittermaier and. Bosa, Distributed Maple–*parallel computer algebra in networked environments*, J. Symb. Comp. 35:3, Academic Press (2003), pp. 305–347, ftp://ftp.risc.uni-linz.ac.at/pub/techreports/2002/02-19.ps.gz.

[76] P. Stewart, *Symbolic computation – a review* (1992), http://www.bham.ac.uk/ctimath/ reviews/nov92/symbol.pdf.

[77] D.Tepeneu and T.Ida, *MathGridLink – Connecting Mathematica to the Grid*, in Procs. 6th Intern. Mathematica Symposium (IMS 2004), Banff, Alberta, Canada, August 2004.

[78] P. S. Wang, Symbolic computation and parallel software (1991), http://icm.mcs.kent.edu/ reports/1991/ICM-9109-12-ab.pdf.

[79] P. S. Wang, S. Gray, N. Kajler, D. Lin and W. Liao, *IAMC architecture and prototyping: A progress report*, Proceedings of ACM ISSAC01, London-Ontario (2001), http:// icm.mcs.kent.edu/research/IAMC.icm/issac01.pdf.

[80] S. M. Watt, *A system for parallel computer algebra programs*, in Procs. Eurocal'85, B.F. Caviness ed., LNCS 204, Springer (1985), pp. 537–538.

[81] A. Weber, W. Küchlin, B. Eggers and V. Simonis, *Parallel computer algebra software as a web component*, http://www.cs.ucsb.edu/conferences/java98/papers/algebra.pdf.

[82] webMathematica, http://www.wolfram.com/products/webmathematica/.

[83] Y. Wu, W. Liao, P. Wang, D. Lin and G. Yang, *An Internet accessible grid computing system:* Grid-Elimino, Proceedings of IAMC 2003, Drexel University (2003), http:// www.symbolicnet.org/conferences/iamc03/grid.pdf.