# 08332 Abstracts Collection
# Distributed Verification and Grid Computing
## — Dagstuhl Seminar —

Henri E. Bal[1], Lubos Brim[2] and Martin Leucker[3]

[1] Vrije Universiteit Amsterdam, NL
`bal@cs.vu.nl`
[2] Masaryk University, CZ
`brim@fi.muni.cz`
[3] TU München, DE
`leucker@in.tum.de`

**Abstract.** From 08/10/2008 to 08/14/2008 the Dagstuhl Seminar 08332 "Distributed Verification and Grid Computing" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

**Keywords.** Parallel Model Checking, Grid Computing, Verification

## 08332 Executive Summary – Distributed Verification and Grid Computing

The Dagstuhl Seminar on **Distributed Verification and Grid Computing** took place from 10.08.2008 to 14.08.2008 and brought together two groups of researchers to discuss their recent work and recent trends related to parallel verification of large scale computer systems on large scale grids. In total, 29 experts from 12 countries attended the seminar.

*Joint work of:* Bal, Henri E.; Brim, Lubos; Leucker, Martin

*Extended Abstract:* http://drops.dagstuhl.de/opus/volltexte/2008/1632

## Large-scale parallel computing on grids

*Henri E. Bal (Vrije Universiteit Amsterdam)*

Computational grids are interesting platforms for solving large-scale computational problems, because they consist of many (geographically distributed) resources. Thus far, grids have mainly been used for high-throughput computing on independent (or trivially parallel) jobs. However, advances in grid software (programming environments, schedulers) and optical networking technology make it more and more feasible to use grids for solving challenging large-scale problems.

The talk will first give a brief introduction to grid infrastructures, using the Dutch DAS-3 Computer Science grid as example. DAS-3 has a flexible and reconfigurable 40 Gb/s optical network called StarPlane between its five clusters and a 10 Gb/s dedicated optical link to the French Grid'5000 system. From a parallel programming point of view, grids like DAS-3 are characterized by a high-latency/high-bandwidth network and a hierarchical structure.

Next, the talk will discuss how algorithms and applications can be optimized to run in such an environment. It focusses on search applications like retrograde analysis, which, much like model checkers, analyze huge search spaces. As a case study, we have implemented an application that solves the game of Awari, which has 900 billion different states. Several optimizations were needed to obtain high performance on DAS-3/StarPlane. Next, the talk discusses promising preliminary results of running the DiVineE model checker on the wide-area DAS-3 system. These early results indicate that there are clear similarities in communication behaviour and performance between this model checker and Awari.

The last part of the talk will discuss research on programming environments that will make it easier to develop parallel applications for grids. Grid programmers often have to use low-level programming interfaces that change frequently, and they have to deal with heterogeneity, connectivity problems, security issues, and dynamically changing execution environments. The Ibis project aims to drastically simplify the whole programming and deployment process of high-performance grid applications. The philosophy of Ibis is that grid applications should be developed on a local workstation and simply be launched from there. Ibis uses middleware-independent Application Programming Interfaces with different abstraction levels, ranging from low-level message passing to high-level divide-and-conquer parallelism and group communication.

## Efficient Large-Scale Model Checking

*Henri E. Bal (Vrije Universiteit Amsterdam)*

Model checking is a popular technique to systematically and automatically verify system properties. Unfortunately, the well-known state explosion problem often limits the extent to which it can be applied to realistic specifications, due to the huge resulting memory requirements. Distributed memory model checkers

exist, but have thus far only been evaluated on small-scale clusters, with mixed results. We examine one well-known distributed model checker in detail, and show how a number of additional optimizations in its runtime system enable it to efficiently check very demanding problem instances on a large-scale, multi-core compute cluster. We analyze the impact of the distributed algorithms employed, the problem instance characteristics and network overhead. Finally, we show that the model checker can even obtain good performance in a high-bandwidth computational grid environment.

*Joint work of:* Verstoep, Kees; Bal, Henri; Barnat, Jiri; Brim, Lubos

*Extended Abstract:* http://drops.dagstuhl.de/opus/volltexte/2008/1630

## State Space Generation for $\mu$CRL: data structures and reusability.

*Stefan Blom (University of Twente)*

Besides the distributed state space generator for $\mu$CRL, there are two experimental ones: a batch scheduler based on-disk generator and a decision diagram based symbolic generator. (The latter is not distributed.) We will briefly introduce these three state space generators.

Subsequently, we will focus on the data structures and protocols used for communication and storage. Also, we will present the APIs of the reusable parts.

## Where the Lions Gone?

*Lubos Brim (Masaryk University), Martin Leucker (TU München)*

In this introductionary talk paper we summarize fundamentals of parallel algorithms for enumerative and symbolic model checking of properties formulated in linear time temporal logic (LTL) as well as a fragments of the modal mu-calculus which naturally subsumes the branching time logics, like CTL (computation tree logic).

*Keywords:* Parallel model-checking

## Parallel Symbolic Algorithms: A Challenge

*Gianfranco Ciardo (Univ. of California - Riverside)*

We discuss sequential state-space generation first, both explicit and symbolic (decision-diagram based). Explicit state-space generation is easier to distribute or parallelize, and we discuss the general framework to do so.

Symbolic state-space generation is instead much harder to parallelize.

We focus on the saturation algorithm, which is by far better than traditional symbolic generation based on breadth-first search, but seems to be inherently sequential. We give a distributed implementation of saturation, which achieves excellent memory load balance, but provides no speedup at all.

Then, we discuss a speculative approach to improve this situation. However, the challenge to improve saturation implementations for a distributed environment remains an open problem, especially if we aim at a good scalability with respect to execution time.

*Keywords:* Parallel or distributed state space generation

## Inspect, ISP, and FIB: Tools for Dynamic Verification and Analysis of Concurrent Programs

*Ganesh Gopalakrishnan (University of Utah)*

We present ongoing work on practical methods for formally verifying message passing and thread programs. These programs are typically written in libraries such as MPI and PThreads. Since creating a formal model of programs in these notations is difficult, we opt for a dynamic verification approach in which API calls of MPI or PThreads made from a C program are intercepted by a special scheduler we include. For PThread program verification, we have a dedicated tool called Inspect whose scheduler implements a version of Dynamic Partial Order reduction, originally proposed by Flanagan and Godefroid. Our implementation includes numerous new ideas including state delta based partial determination of visited states, and specialized algorithms for race detection. For MPI program verification, we have a dedicated tool called ISP whose scheduler implements a new dynamic partial order reduction algorithm called Partial Order reduction avoiding Elusive interleavings (POE). The ISP tool is also able to determine whether MPI barriers used in a program are Functionally Irrelevant Barriers (FIB) or not. When a barrier is determined to be a FIB, it can be removed without changing the communication behavior of the overall MPI program. These works have appeared in a number of recent venues, and details may be found from our website http://www.cs.utah.edu/formal_verification. The authors thank Rajeev Thakur of Argonne and William Gropp of UIUC for their feedback and encouragement.

*Keywords:* MPI, PThreads, Dynamic Verification, Formal Verification, Dynamic Partial Order Reduction

*Joint work of:* Yu Yang; Sarvani Vakkalanka; Subodh Sharma; Anh Vo; Michael DeLisi; Ganesh Gopalakrishnan; Robert M. Kirby

*Full Paper:*
http://www.cs.utah.edu/formal_verification

*See also:* http://www.cs.utah.edu/formal_verification

## Real-Time Online Interactive Applications: Challenges and Prospects on the Grid

*Sergei Gorlatch (Universität Münster)*

Real-Time Online Interactive Applications (ROIA) contain such highly dynamic and commercially important applications as online computer games and e-learning based on simulation. We describe the specific problems of ROIA and present some approaches to their efficient implementation using Grid Computing.

## Demonstration of the mCRL2 toolset.

*Jan Friso Groote (Eindhoven Univ. of Technology)*

We demonstrate the mCRL2 toolset that is capable of modelling and analysing complex communicating systems. Typical examples of these are distributed algorithms and communication protocols. However, modelling of communication is also applied in the design of embedded systems (containing lots of embedded controllers) and factory automation (where passing a product is also modelled as passing a message).

Of course grid computers are huge message passing systems that can certainly benefit from these modelling techniques.

The mCRL2 language and tools originated in 1990 when a Common Representation Language (CRL) was being developed as an intermediate between high level behavioural specification languages (such as SDL, Chill, LOTOS, PSF) and tools. The language CRL was so complex that it was decided to make a micro version of it, micro-CRL, or $\mu$CRL. For this language theory, proof methodologies and tools have been constructed. A good description of it can be found in W. Fokkink. Modelling Distributed Systems. Springer-Verlag. 2007.

In approximately 2002 it was decided that $\mu$CRL needed a successor, called mCRL2. In nature both languages are the same, but mCRL2 has much more convenient built in data-types that makes the language more suitable for specification and analysis of complex systems.

By using an overly simple example (the alternating bit protocol) it is shown how the mCRL2 specification looks like, how states spaces are generated, reduced and visualized. It has been shown how modal formulas look like, and how these can verified using the toolset.

The mCRL2 specification language has been used to analyse and design a whole plethora of systems, including controllers to steer cars, pace makers, large volume copiers, tv-controllers, component place machines, robots, etc. We generally find that designs that are made via thoroughly analysed models are of a substantial higher quality.

Tools, documentation and show cases can be found on www.mcrl2.org.

The tools are distributed under the BOOST license, that permits free, unlimited use.

*Keywords:* Behavioural specification, tools, mCRL2

*Full Paper:*
http://www.mcrl2.org

## Approaches to Distributing Bounded Model Checking

*Keijo Heljanko; Wieringa, Siert; Niemenmaa, Matti (Helsinki Univ. of Technology)*

Bounded model checking is a symbolic model checking method that employs efficient propositional satisfiability solvers to do symbolic model checking. In the talk we first describe the bounded model checking approach in general.

We then discuss ongoing research into different approaches for distributing bounded model checking. Our main focus is on an algorithm of Rintanen to solve artificial intelligence planning problems faster in the sequential setting by solving planning instances of different bounds in parallel in an interleaved fashion. We experiment with simulated variants of the approach in the distributed setting and find new variants of the approach that look promising and suitable for direct distributed implementation.

*Keywords:* Bounded model checking, SAT, distributed verfication

## Computing in the Mist: Writing Applications for Unknown Machines

*Thilo Kielmann (Vrije Universiteit Amsterdam)*

The landscape of computer architecture is changing dramatically.

Computer processors are no longer getting faster. Instead, we have to use several processors in parallel for future performance improvements.

This means that every computer, from mobile phones, via graphics processors, PC's, clusters, up to grids, clouds, and large data centers are becoming parallel environments.

What does this mean for application development? Obviously, sequential code will hardly be able to exploit near future machines. Not so obviously, neither will traditional parallel code.

In this presentation, I will briefly sketch processor architectures available today and expected soon, and will argue for virtualization, portability, scalability and fault-tolerance as the key ingredients for programming environments we will need from now on.

## Application specific support for grid jobs

*Josva Kleist (Nordic DataGrid Facility)*

In this talk I will present how the Nordic DataGrid Facility have worked together with two scientific communities on providing support for their specific applications.

The first example is targeted to one specific application for a quite specific scientific area: Carbon dioxide sequestration is one of the hot topics in climate research at the moment. The NDGF funded CO2 Community Grid project enables CO2 researches to use computational resources in grid in a transparent and easy manner by providing a simple command-line based environment for running the simulations, but like the command-line interface they are used to when executing on their own workstation.

The second is a more general framework: The BioGrid Community Grid project is an effort to establish a Nordic grid infrastructure for bioinformatics, aiming both to gridify computationally heavy tasks and to coordinate bioinformatic infrastructure efforts in order to use the Nordic resources more efficiently. The widely used bioinformatic software packages BLAST and HMMer have now been gridified in an optimised way and allowing for multicore support. The frequently used databases UniProtKB and UniRef have been made available on the distributed and cached storage system within the Nordic grid.

The focus in the talk will not be on the specific applications, it will be on how we have brought the applications to run on grid in such a way that the communities fell that grids can help them get their research done.

*Keywords:* Grid, BioInformatics, CO2, parallell

## Parallelising symbolic state-space generators: Frustration and hope

*Gerald Lüttgen (University of York)*

Due to the irregular nature of the task, efficient algorithms for symbolic state-space generation are notoriously hard to parallelise. This talk explores two different strategies for parallelising and implementing one such algorithm, Saturation, on multi-core PCs: one strategy employs the parallel language Cilk, while the other uses a thread pool implemented in Pthreads.

I will analyse the underlying parallel overheads, present experimental results and argue their relevance. The conclusions will give room for both frustration and hope regarding the parallelisability of symbolic model checkers on shared-memory architectures.

*Keywords:* Parallel state-space generation, decision diagrams, multi-core architectures

## Analysis of a quorum consensus protocol

*Simona Orzan (TU Eindhoven)*

We model in mCRL2 a basic protocol for Read and Write operations on a distributed replicated database.

Relying on the tool support, we analyze this basic model and design two extensions of it that tolerate faults. We also show how a model primarily built for checking correctness, can also be used for getting fast efficiency evaluations.

*Joint work of:* Groote, Jan Friso; Orzan, Simona

## Distributed Model Checking, @home?

*Fernando Schapachnik (University of Buenos Aires)*

This short work-in-progress presentation deals with asking the question of whether it makes sense or not to try an @home approach to timed model checking, where processing nodes can come and go.

*Keywords:* Distributed model checking, @home, at home, timed automata, Zeus, @office

*Joint work of:* Schapachnik, Fernando; Vaquier, José Ignacio

## Verification of MPI-based Computations

*Stephen Siegel (University of Delaware)*

The Message Passing Interface is a widely-used parallel programming model and is the effective standard for high-performance scientific computing. It has also been used in parallel model checkers, such as DiVinE. In this talk we discuss the verification problem for MPI-based programs. The MPI is quite large and the semantics complex. Nevertheless, by restricting to a certain subset of MPI, the verification problem becomes tractable. Certain constructs outside of this subset (such as wildcard receives) can lead to a rapid blowup in the number of states, but MPI-specific reduction techniques have led to progress in combating this state explosion. Specifying correctness is another challenge. One approach is to use a trusted sequential version of the program as the specification, and use model checking and symbolic execution techniques to establish the functional equivalence of the sequential and parallel versions. This approach is supported in MPI-SPIN, an extension to the model checker SPIN for verifying MPI-based programs.

*Keywords:* MPI, Spin, model checking, MPI-Spin, symbolic execution

*Extended Abstract:* http://drops.dagstuhl.de/opus/volltexte/2008/1631

## GXP—a minimalist's user-level tool for distributed computing

*Kenijro Taura (University of Tokyo)*

GXP is a parallel shell that is specifically designed as an infrastructure to support parallel computing on large-scale distributed resources (Grid). Its specific design goals include: (1) make installation almost effortless, (2) coordinate resources accessed via heterogeneous protocols (SSH, SGE, TORQUE, etc.) and provide a uniform interface to these resources, and (3) make deployment of parallel programs on a large number of nodes fast and easy, and (4) provide a simple workflow engine via parallel make. These features are integrated to a command line similar to parallel shells. In this talk I show how its design is and must be different from other similar tools.

*Keywords:* Parallel shell, distributed shell, parallel workflow

## How to Survive Developing Multi-Core Algorithms?

*Michael Weber (University of Twente)*

We describe a parallel algorithm for solving parity games, with applications in, e.g., modal mu-calculus model checking with arbitrary alternations, and (branching) bisimulation checking. The algorithm is representative for a larger class of algorithm which perform fixpoint computations, (graph-based) value propagation, etc.

The algorithm presented here is designed for parallel shared-memory architectures. We will highlight some of the issues encountered during the development, which complicate reasoning about the correctness of the actual implementation.

*Keywords:* Parity games, parallel, multi-core, shared memory, lock-free, wait-free

## (Parallel) Transient Solution Methods for Transition Class Models

*Verena Wolf (EPFL - Lausanne)*

Transition class models represent well-structured continuous-time Markov chains with a state space of possibly infinite size. Their numerical analysis is in many cases computationally expensive or even infeasible, especially for transient measures.

We introduce the sliding window method, which computes an approximate solution of the CME by performing a sequence of local analysis steps. In each step, only a manageable subset of states is considered, representing a “window” in the state space. In subsequent steps, the window follows the direction in which the probability mass moves until the time period of interest has elapsed. We construct the window based on a deterministic approximation of the future behavior of the system, where we estimate upper and lower bounds on the state variables. We present a parallel version of our approach, where the window is split into smaller parts. Each such part defines the initial conditions of a subproblem, which is passed to a worker thread. For a given time step, each worker computes transient measures of a subsystem. The master combines all solutions of the workers and distributes the problem of the next iteration step in the same way.

This approach allows to analyse transition class models with state spaces of arbitrary size, which occur in various application domains such as systems biology.

*Keywords:* Biological Networks - Stochastic Modeling - Markov Chains - Structured Model Representation - Numerical Analysis

## A Typical Verification Challenge for the GRID

*Jaco van de Pol (University of Twente)*

A typical verification challenge for the GRID community is presented. The concrete challenge is to implement a simple recursive algorithm for finding the strongly connected components in a graph. The graph is typically stored in the collective memory of a number of computers, so a distributed algorithm is necessary.

The implementation should be efficient and scalable, and separate synchronization and implementation details from the purely algorithmic aspects. In the end, a framework is envisaged for distributed algorithms on very large graphs. This would be useful to explore various alternative algorithmic choices.

A more detailed description of the challange is part of the proceedings of this seminar.

*Keywords:* Strongly connected components, distributed algorithms, breadth first search

*Full Paper:* http://drops.dagstuhl.de/opus/volltexte/2008/1629

*Full Paper:*
http://dx.doi.org/10.1016/j.entcs.2008.02.001