Annual Report of the Chair of Computer Science 2 (Programming Systems)

Address: Martensstr. 3, 91058 Erlangen
Phone: +49-9131-85-27621
Fax: +49-9131-85-28809
E-Mail: info@i2.informatik.uni-erlangen.de

Ordinarius:
Prof. Dr. Michael Philippsen
Honorary Professor:
Hon.-Prof. Dr.-Ing. Bernd Hindel
Hon.-Prof. Dr.-Ing. Detlef Kips
Professor Emeritus:
Prof. em. Dr. Hans Jürgen Schneider
Secretary:
Margit Zenk (from May 1, 2013)
Scientific Staff:
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Daniel Brinkers
Dipl.-Inf. Georg Dotzler
Demian Kellermann, M. Sc. (from April 1, 2013)
Dipl.-Inf. Stefan Kempf
Dipl.-Math. Jakob Krainz
Andreas Kumlehn, M. Sc.
Dipl.-Inf. Christopher Mutschler
Dr.-Ing. Norbert Oster
Dipl.-Inf. Mykola Protsenko (from May 15, 2013)
Norbert Tausch, M. Eng.
PD Dr. Ronald Veldema
Dipl.-Inf. Tobias Werth
Guest:
Dr.-Ing. Josef Adersberger
Dipl.-Inf. Samir Al-Hilank
Dipl.-Inf. Ralf Ellner
Dr.-Ing. Martin Jung
Florian Lautenschlager, M. Sc.
Dr.-Ing. Stephan Otto
External Teaching Staff:
Dr.-Ing. Klaudia Dussa-Zieger
Dr.-Ing. Martin Jung
Dr.-Ing. Stephan Otto

The Chair of Computer Science 2 (programming systems) was founded in 1972 and is headed by Prof. Michael Philippsen (as the successor of Prof. H.-J. Schneider) since April 2002. Closely associated with the programming systems group is the professorship for Didactics of Computer Science.

1 Focus of research

The main research topics in the programming systems group are programming of parallel or distributed systems and programming of embedded or mobile systems. Software (and its development) for such systems should ideally be as complex, portable, maintainable and robust as existing software for single core systems and workstations. It is our long-term goal to allow applications to take full advantage of the available computing and network power. A particular focus lies on programming systems for multi-cores because more and more cheap multi-core high-performance parallel hardware (for example graphics cards or FPA-Hardware) is available. This will have an unpredictable impact on the future of the software landscape. Research results of the group are always evaluated by means of prototypes and demonstrators.

Important Research Areas

- **Exploit the available parallelization potential.** In the future the clock rate of multi-core systems will grow only slowly whereas the number of cores will grow. This makes it necessary to exploit the parallelization potential of already older, existing software to allow it to benefit from the new hardware. As a consequence, in most application areas a change to parallel computing is unavoidable. Therefore, the programming systems group develops tools to support the programmer interactively in reengineering existing sequential applications. It also develops architectural patterns for new software projects that scale automatically to support a growing number of cores.

- **Achieve portability in high-performance applications.** Up to the present, application programmers achieve the best possible performance results only if they handle latency issues and communications between different components of the system manually, optimize their code with hardware specific ”tricks” and split their application into multiple sections to outsource them to other hardware (for example graphics cards). To change this situation, the programming systems group researches the performance impact of higher programming abstraction layers that would improve programming productivity and software portability. The improvements are caused by generated code that allows the distribution of the program onto multiple heterogeneous system components to permit parallel execution. The higher abstraction layer makes the communication between the components transparent for the developer. To increase the efficiency of this approach it is necessary to give the programmer the possibility to express available domain knowledge in the programming language. For the higher abstraction layer, the details of the hardware architecture are hidden from the developer (for example by library functions or programming language extensions).

- **Adapt the degree of parallelism dynamically**. High-performance applications are often developed for a fixed number of cores. As requested cluster nodes of a batch system are statically assigned for a fixed time period, inefficient reservation gaps are unavoidable. Similar problems appear in multi-threaded applications on multi-core systems. The programming systems group works on the dynamic adaptation of the extent of parallelism by the means of code transformations (under consideration of the resulting data redistribution) and operating system interactions. As control flow based synchronization measures interfere with the necessary analyzes, the programming systems group researches new programming constructs that can replace the existing ones and allow to specify the synchronization in a data-centric way.

- **Develop Testing for Parallelism**. In software engineering, testing has always assumed an important role. Code coverage, test data generation, reliability assessment etc. are tools of the trade. Unfortunately, current research insufficiently covers the indeterminism caused by concurrency. To deal with that issue, the programming systems group develops tools that consider (based on the coverage criteria) interleavings of parallel threads in their test data generation. This topic also includes research on operating systems and schedulers. As concurrency considerably increases the search space of the test generation it is necessary to develop infrastructures that allow the test generation and execution on a cluster.

- **Improve of Software Development Processes**. The current development practice of complex, business or security critical software in global distributed teams (commonly found in the software industry) demands compliance with well-defined software development processes. To support the enforcement of this requirement, appropriate development tools are used. The Practical Software Engineering research group that is lead by the honorary professors Dr. Bernd Hindel and Dr. Detlef Kips cover the corresponding research area. Both possess long term experience in industrial software projects as managers of medium sized software companies. The goal of the Practical Software Engineering group is the development of a machine executable notation for modeling of software development processes. For that purpose the research group examines the semi-automatic retrieval of traceability information from the artefacts of different tools and notations as well as the model based development, integration and configurations of software components, used in the design of automotive embedded systems.

2 Research projects

2.1 Design for Diagnosability

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Andreas Kumlehn, M. Sc.
Dr.-Ing. Josef Adersberger
Florian Lautenschlager, M. Sc.
Start: 15.5.2013
Sponsored by:
IuK Bayern
Contact:
Prof. Dr. Michael Philippsen
Phone: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Many software systems behave obtrusively during the test phase or even in normal operation. The diagnosis and the therapy of such runtime anomalies is often time consuming and complex - up to being impossible. There are several possible consequences for using the software system: long response times, inexplicable behaviors, and crashes. The longer the consequences remain unresolved, the higher is the accumulated economic damage.

”Design for Diagnosability” is a tool chain targeted towards increasing the diagnosability of software systems. By using the tool chain that consists of modeling languages, components, and tools, runtime anomalies can easily be identified and solved - ideally already while developing the software system. Our cooperation partner QAware GmbH offers a tool called Software EKG that enables developers to explore runtime metrics of software systems by visualizing them as time series.

The research project Design for Diagnosability enhances the existing Software EKG. A so-called Software Black Box measures technical and functional runtime values of a software system in a minimally intrusive way. FindPerformanceBugs automatically detects runtime anomalies. The link between the observed software system and the developed tool chain is the Performance Modeling Language (PML), a modeling language for performance relevant properties of a software system. The PML is used for the data collection in the Software Black Box. Additionally, the PML is the information base for the Runtime Anomaly Diagnosis Language (RADL) that describes rules to automatically detect runtime anomalies in the observed software system.

2.2 Efficient Software Architectures for Distributed Event Processing Systems

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dipl.-Inf. Christopher Mutschler
Duration: 15.11.2010–15.5.2014
Sponsored by:
Fraunhofer Institut für Integrierte Schaltungen

Localization Systems (also known as Realtime Location Systems, or RTLS) become more and more popular in industry sectors such as logistics, automation, and many more. These systems provide valuable information about whereabouts of objects at runtime. Therefore, processes can be traced, analyzed and optimized. Besides the research activities at the core of localization systems (like resilience and interference-free location technologies or methods for highly accurate positioning), algorithms and techniques emerge that identify meaningful information for further processing steps. We focus our research on automatic configuration methods for RTLSs as well as on the generation of dynamic moving models and techniques for event processing on position streams at runtime.

In 2011, we investigated whether events can be predicted after analyzing and learning event streams from the localization system at runtime. As a result, we are able to deduce models that represent the information buried in the event stream to predict future events.

We developed several methods and techniques in 2012 that process and detect events with low latency. Events (composite, complex) can be detected by means of a hierarchical aggregation of sub-events that themselves are detected by (several) event detectors processing sub-information in the event stream. This greatly reduces the complexity of the detection components and renders them fully maintainable. They even can use parallel or distributed cluster architectures more efficiently so that important events can be detected within a few milliseconds.

In 2013 we designed methods to further minimize detection latency in distributed event-based systems. A new migration technique modifies and optimizes the allocation of software components in a networked environment at runtime to minimize networking overhead and detection latencies. In addition we developed a speculative event processing technique that wraps conservative buffering techniques to exploit available system resources. As all the available resources can now be used to process events earlier detection latencies further shrink. We also created a representative data set (consisting of realtime position data and event streams) and a corresponding task description for the ACM DEBS Grand Challenge at the 7th Intl. Conf. on Distributed Event-Based System.

The project is a contribution of the Programming Systems Group to the [IZ ESI]http://www.esi.uni-erlangen.de/

2.3 Embedded Systems Institute

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Christopher Mutschler
Andreas Kumlehn, M. Sc.
Dr.-Ing. Norbert Oster
Demian Kellermann, M. Sc.
Start: 1.9.2007

In September 2007 the ESI – Embedded System Institute – was founded as an interdisciplinary center at the Friedrich-Alexander-University (FAU) with the goal to coordinate and organize research, teaching, and further education in the field of embedded systems.

ESI brings together existing skills within the university and interests, activities, and goals of large and medium sized companies in the field of embedded systems.

Companies obtain access to latest research results and the opportunity to develop common projects, to establish ties, and to find co-operation partners. The ESI concentrates the skills of the chairs of computer sciences and makes them usable for co-operation projects. Hence, the latest research results can be transferred into products in a speedy way. Finally, the ESI may serve as a platform for recruiting excellent students and highly qualified young academics at an early stage.

The chair of computer science department 2 (Prof. Philippsen) is one of the active founders of the ESI and carries out research projects within the ESI.

More information can be found at http://www.esi.uni-erlangen.de and http://www.esi-anwendungszentrum.de

2.4 ErLaDeF - Embedded Realtime Language Development Framework

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
PD Dr. Ronald Veldema
Dipl.-Inf. Thorsten Blaß
Dipl.-Math. Jakob Krainz
Dipl.-Inf. Daniel Brinkers
Start: 1.1.2012
Contact:
PD Dr. Ronald Veldema
Phone: +49-9131-85-27622
Fax: +49-9131-85-28809
E-Mail: ronald.veldema@fau.de

ErLaDeF is our test-bed for new programming language and compiler techniques. Our main focus is on building infrastructure for easier (hard + soft) real-time embedded parallel systems programming.

We focus on hard real-time embedded systems as they are about to go massively parallel in the near future.

Real-time and embedded systems also have hard constraints on resource usage. For example, a task should complete in a fixed amount of time, have guaranteed upper-limits on the amount of memory used, etc.

We are developing different ways to manage this concurrency using a combination of strategies: simpler language features, automatic parallelization, libraries of parallel programming patterns, deep compiler analysis, model checking, and making compiler analysis fast enough for interactive use.

Parallelization through Language Features

Using simpler language features (compared to, for example, Java or C#) ensures that the compiler analysis finds the existing concurrency and real-time violation bugs. This forms a basis of the other techniques we pursue in this project. In 2013 we have explored alternatives to polymorphism and inheritance that may be easier to analyze. We have also examined alternative thread synchronization methods, for example transactional memory, implicit synchronization, remote procedure calls, etc.

Runtime Parallelization of Programs

Our automatic parallelization efforts are currently focused on dynamic parallelization. While a program is running, it is analyzed to find loops where parallelization can help performance. Our current idea is to run long-running loops three times. The first two runs analyze the memory accesses of the loop and can both run in parallel. The first run stores in a shared data structure for every memory address in which loop iteration a write access happens. We do not need any synchronization for this data structure, only the guarantee that one value is written to memory, when two concurrent writes happen. In the second pass we check for every memory access, if it has a dependency to one of the stored write accesses. A write access is part of any data-dependency, so we can find all types of data dependencies. If we do not find any, the loop is actually run in parallel. If we find dependencies, the loop is executed sequentially. We can execute the analyses in parallel to a modified sequential execution of the first loop iterations.

In 2013 we refined the analyses by exploiting that the sequential execution of the first iterations will happen before the parallel execution of the remaining iterations. By overlapping some sequential iterations with the analysis of the remainder of the loop, the analysis does not add (much) overhead in the case that it turns out that the loop cannot be parallelized after all.

Design Patterns for Parallel Programming

A library of parallel programming patterns allows a programmer to select well known parallelization and inter-core communication strategies from a well-debugged library. We are performing research into what (communication) patterns actually exist and when they can be applied. We have collected over 30 different patterns for parallel communication. In 2013 we investigated mechanisms to automatically determine the fitting implementation for a given software and hardware environment. We also added a set of distributed channels where cores can send data from one local memory to another. The distributed channels allow the library to be used to program modern Network-on-Chip (NoC) processors.

Model Checking

Model checking examines all interleavings of threads to determine if a concurrency bug can occur. Bugs that a model checker can find are race conditions, deadlocks, etc. A model checker can guarantee that a bug is found at the cost of long analysis times. In 2013 we made further progress in developing a language independent model checking framework. The model checker was optimized in two novel ways in 2013. First we discovered that we can reduce the model-checker's state space by exploiting a program's locality. Second we found that by using a data-centric synchronization programming model we can achieve coarse grained interleaving to reduce the model-checker's state space some more.

Interactive Program Analysis

To ensure that program design errors are caught early in the development cycle, it is necessary to find bugs while editing. This requires that any program analysis works at interactive speeds. We are following two approaches to this.

The first approach centers around algorithmic changes to program analysis problems. Making analysis problems lazy means that only those parts of a program should be examined that are pertinent to the question that is currently asked by the compiler. For example, if the compiler needs to know which functions access a certain object, it should not examine unrelated functions, classes, or packages. Making program analysis incremental means that a small change in the program should only require small work for the (re-)analysis.

To achieve that, a program is split recursively into parts. Then, for each of the parts, it is calculated which effect it would have during an execution of the program. For each part, a symbolic representation of its effects are saved.

These representations can then for one be used to find the errors that occur when two of the parts interact (concurrently or non-concurrently). Also, we can deduce the effects that a bigger part of the program has during it's execution by combining the effects of the smaller parts the bigger part consists of. This enables incremental analysis, because changes in one place do not cause the whole program to be reanalyzed, as the symbolic representations of the effects of unchanged parts of the program stay unchanged as well.

In 2013 our key focuses were twofold: Firstly, we developed data structures that can both precisely and efficiently describe the effects of a part of a program. Secondly, we developed both efficient and precise algorithms to create and use these data structures.

Our second approach to bring compiler analysis to interactive speed is to make the analysis itself parallel. In 2013 we continued to develop data-parallel formulations of basic compiler analyses. We have started to implement a generic data-parallel predicate propagation framework. Its data-parallel forms are then portably executable on many different multi-core architectures.

The ErLaDeF project is a contribution of the Chair of Computer Science 2 (Programming Systems) to the IZ ESI (Embedded Systems Initiative, http://www.esi.uni-erlangen.de)

2.5 Graphs and Graph Transformations

Project manager:
Prof. em. Dr. Hans Jürgen Schneider
Start: 1.10.2004
Contact:
Prof. em. Dr. Hans Jürgen Schneider
Phone: +49-9131-85-27620
Fax: +49-9131-85-28809
E-Mail: hans.juergen.schneider@fau.de

Graphs are often used as an intuitive aid for the clarification of complex matters. Examples of outside computer science include, e.g., chemistry where molecules are modeled in a graphical way. In computer science, data or control flow charts are often used as well as entity relationship charts or Petri-nets to visualize software or hardware architectures. Graph grammars and graph transformations combine ideas from the fields of graph theory, algebra, logic, and category theory, to formally describe changes in graphs.

Category theory is an attractive tool for the description of different structures in a uniform way, e.g., the different models for asynchronous processes: Petri-Nets are based on standard labeled graphs, state charts use hierarchical graphs, parallel logic programming can be interpreted in a graph-theoretical way using so-called jungles, and the actor systems can be visualized as graphs, whose labeling alphabet is a set of term graphs.

Lately, we have concentrated our attention on a theoretical aspect.

Our work on graph transformation is based on notions borrowed from category theory. The so-called double-pushout approach represents a production by two morphisms starting at a common interface graph. One pushout glues the left-hand side of the production into the context, the other does with the right-hand side. Effectively constructing a derivation step, however, requires finding a pushout complement on the left-hand side. Some people consider this disadvantageous. In 1984, Raoult has proposed to model graph rewriting by a single pushout; Loewe has extensively studied this approach, but the discussion was mainly restricted to injective morphisms. Under this assumption, the approaches are equivalent. Some relevant applications such as term graph rewriting, however, lead to non-injective morphisms. We have examined these cases in detail, and we could show that the equivalence also holds for non-injective cases as long as the handle satisfies some reasonable conditions.

2.6 International Collegiate Programming Contest at the FAU

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Daniel Brinkers
Dipl.-Math. Jakob Krainz
Start: 1.11.2002
Contact:
Dipl.-Inf. Tobias Werth
Phone: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

The Association for Computing Machinery (ACM) has been hosting the International Collegiate Programming Contest (ICPC) for many years. Teams of three students try to solve nine to eleven programming problems within five hours. What makes this task even harder, is that there is only one computer available per team. The problems demand for solid knowledge of algorithms from all areas of computer science and mathematics, e.g. graphs, combinatorics, strings, algebra and geometry.

The ICPC consists of three rounds. First, each participating university hosts a local contest to find the three teams that are afterwards competing in one of the various regional contests. Erlangen lies in the catchment area of the Northwestern European Regional Contest (NWERC) where also teams from e.g. Great Britain, Benelux and Scandinavia. The winners of all regionals in the world (and some second place holders) advance to the world finals in spring of the following year.

In 2013 two local contests took place in Erlangen. During the winter semester a team contest was conducted with teams consisting of at most three students. The main goal of this contest was to interest new students in the contests. We had 19 FAU teams plus 30 more teams from universities all over Europe.

As in the previous years, in the summer term the seminar ”Hello World - Programming for the Advanced” served to prepare students from different disciplines in algorithms and contest problems. In the German-wide contest of the summer term we chose the students that represent the FAU at the NWERC 2013 in Delft. 14 teams with students of computer science, computational engineering, mathematics as well as informations and communication technology took the challenge. We selected ten students for the NWERC, forming three teams and one reserve. At the NWERC in Delft, the best FAU team reached a bronze medal by solving seven problems. The second and third team also did a great job solving only one problem less and finished on rank 10, 20, and 25 of 92 teams.

2.7 InThreaT - Inter-Thread Testing

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dr.-Ing. Norbert Oster
Start: 1.1.2012
Contact:
Dr.-Ing. Norbert Oster
Phone: +49-9131-85-28995
Fax: +49-9131-85-28809
E-Mail: norbert.oster@fau.de

In order to achieve higher computing performance, microprocessor manufacturers do not try to achieve faster clock speed anymore - on the contrary: the absolute number of cycles has even decreased, while the number of independent processing units (cores) per processor is continually increased. Due to this evolution, developers must learn to think outside the box: The only way to make their applications faster (in terms of efficiency) is to modularize their programs such that independent sections of code execute concurrently. Unfortunately, present-day systems have reached a level of functional complexity, such that even software for sequential execution is significantly error-prone - the parallelization for multiple cores adds yet another dimension to the non-functional complexity. Although research in the field of software engineering emerged several different quality assurance measures, there are still very few effective methods for testing concurrent applications, as the broad emergence of multi-core systems is relatively young.

This project aims to fill that gap by providing an automated test system. First of all, a testing criteria hierarchy is needed, which provides different coverage metrics tightly tailored to the concept of concurrency. Whilst for example branch coverage for sequential programs requires the execution of each program branch during the test (i.e. making the condition of an if-statements both true and false - even if there is no explicit else branch), a thorough test completion criterion for concurrent applications must demand for the systematic execution of all relevant thread interleavings (i.e. all possibly occurring orderings of statements, where two threads may modify a shared memory area). A testing criterion defines the properties of the ”final” test set only, but does not provide any support for identifying individual test cases. In contrast to testing sequentially executed code, test scenarios for parallel applications must also comprise control information for deterministically steering the execution of the TUT (Threads Under Test).

In 2012, a framework for Java has been developed, which automatically generates such control structures for TUT. The tester must provide the bytecode of his application only;

further details such as source code or restrictions of the test scenario selection are optional. The approach uses aspect-oriented programming techniques to enclose memory access statements (reads or writes of variables, responsible for typical race conditions) with automatically generated advices. After weaving the aspects into the SUT (System Under Test), variable accesses are intercepted at runtime, the execution of the corresponding thread is halted until the desired test scenario is reached, and the conflicting threads are reactivated in the order imposed by the given test scenario. In order to demonstrate the functionality, some naive sequence control strategies were implemented, e.g. alternately granting access to shared variables from different threads.

In 2013, the prototypical implementation of the InThreaT framework has been reengineered as an Eclipse plugin. This way, our approach can also be applied in a multi-project environment and the required functionality integrates seamlessly into the development environment (IDE) familiar to programmers and testers. In addition, the configuration effort required from the tester could be reduced, as e.g. selecting and persisting the interesting points of interleaving also became intuitively usable parts of the IDE. In order to automatically explore all relevant interleavings, we need an infrastructure to enrich functional test cases with control information, required to systematically (re)execute individual test cases. In 2013, such an approach for JUnit has been evaluated and implemented prototypically, which allows to mark individual test cases or whole test classes with adequate annotations.

2.8 OpenMP/Java

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
PD Dr. Ronald Veldema
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Duration: 1.10.2009–1.10.2015

JaMP is an implementation of the well-known OpenMP standard adapted for Java. JaMP allows one to program, for example, a parallel for loop or a barrier without resorting to low-level thread programming. For example:

```
class Test {
...void foo(){
......//#omp parallel for
......for (int i=0;i<N;i++) {
```

.........a[i]= b[i]+ c[i]

......}

...}

}

is valid JaMP code. JaMP currently supports all of OpenMP 2.0 with partial support for 3.0 features, e.g., the collapse clause. JaMP generates pure Java 1.5 code that runs on every JVM. It also translates parallel for loops to CUDA-enabled graphics cards for extra speed gains. If a particular loop is not CUDA-able, it is translated to a threaded version that uses the cores of a typical multi-core machine. JaMP also supports the use of multiple machines and compute accelerators to solve a single problem. This is achieved by means of two abstraction layers. The lower layer provides abstract compute devices that wrap around the actual CUDA GPUs, OpenCL GPUs, or multicore CPUs, wherever they might be in a cluster. The upper layer provides partitioned and replicated arrays. A partitioned array automatically partitions itself over the abstract compute devices and takes the individual accelerator speeds into account to achieve an equitable distribution. The JaMP compiler applies code-analysis to decide which type of abstract array to use for a specific Java array in the user's program.

In 2013, we examined how to better support Java objects in OpenMP parallel code, regardless of where the code is executed. We found that we needed to restrict the language slightly by forbidding inheritance of objects used in a parallel block. This ensures that the objects will not be of a different type than what is seen at compile time. We use this property to, for example, allow object inlining into arrays to occur naturally. With the added inlining, communication of objects and arrays over the network and to the compute devices was accelerated enormously, including a small performance increase on the devices themselves.

2.9 PATESIA - Parallelization techniques for embedded systems in automation

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Start: 1.6.2009
Sponsored by:
ESI-Anwendungszentrum

Contact:
Dipl.-Inf. Stefan Kempf
Phone: +49-9131-85-27624
Fax: +49-9131-85-28809
E-Mail: stefan.kempf@fau.de

This project was launched in 2009 to address the refactorization and parallelization of applications used in the field of automation. The programs are executed on specially designed embedded systems. This hardware forms an industry standard and is used worldwide. As multicore-architectures are increasingly used in embedded systems, existing sequential software must be parallelized for these new architectures in order to gain an improvement of performance. As these programs are typically used in the industrial domain for the control of processes and factory automation they have a long life cycle. Because of this, the programs are often not being maintained by their original developers any more. Besides that, a lot of effort was spent to guarantee that the programs work reliably. For these reasons, the software is only extended in a very reluctant way.

Therefore, a migration of these legacy applications to new hardware and a parallelization cannot be done manually, as it is too error prone. Thus, we need tools that perform these tasks automatically or aid the developer with the migration and parallelization.

Research on parallelization techniques

We developed a special compiler for the parallelization of existing automation programs. First, we examined industry automation applications with respect to automatic parallelizability. We found that it is hard to perform an efficient automatic parallelization with existing techniques. Therefore, we shifted our focus to let our compiler generate feedback about code sequences that are hard to parallelize and leave the final parallelization to the developers. However, the proper synchronization of the parallel threads was performed by the compiler. We used atomic blocks and transactional memory as synchronization techniques and implemented those in a prototypical runtime environment. In 2013, we developed a new parallelization technique that works well for industry automation codes. Our new approach uses program slicing and graph coloring to extract parallel threads of execution from a sequential program. We developed a graphical tool that presents the results of the analysis to developers so that they can easily spot those locations in their codes that prevent a parallelization. This helps programmers to refactor their code in order to improve th results of the analysis. Additionally, we developed another novel compiler analysis that combines STM and lock inference, which are the currently two most popular techniques to implement critical sections. Our compiler divides critical sections into distinct sequences of code and for every sequence, it chooses the technique that leads to a more fine-grained synchronization. Since lock inference typically has a lower runtime overhead than STM but often leads to coarse-grained synchronization, we enhanced the former technique with optimizations that now can implement more critical sections with fine-grained lock inference than was previously possible.

Research on migration techniques

Our research on the migration of legacy applications originally consisted of having a tool that automatically replaces suboptimal code constructs with better code. The code sequences that had to be replaced as well as the replacement codes were specified by developers by means of a newly developed pattern description language. However, we found this approach to be too difficult for novice developers.

This led us to the development of a new tool that automatically learns patterns from source code archives, recognizes them in other projects, and presents recommendations to developers. The foundation of our tool lies in the comparison of two versions of the same program. It extracts the changes that were made between two versions, derives the patterns of suboptimal and better code from these changes, and saves the patterns in a database. Our tool then uses these patterns to perform similar changes on the source code of different programs. In 2013 the transformation tool that automatically extracts patterns from version control systems was completed. Our tool continues to extract changes from software archives, but since the detected changes contained too much irrelevant low-level information, we added a new preprocessing pass that extracts high-level information from these changes that are then used for further optimization steps. Then we added a new step in which our tool encodes the high-level changes between two source codes as a string and then compares all strings from the source code archive with the Needleman-Wunsch algorithm. This comparison results in similarity scores between the strings. If two strings have a high similarity score, then the source code changes that are represented by those two strings are almost identical, which means that the changes belong to the same group of code modification. Finally, the tool classifies the groups and processes them as before in order to generate the patterns of suboptimal and better code.

Parts of the project are funded by the http://www.esi-anwendungszentrum.de/ .

2.10 Software Project Control Center

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dr.-Ing. Josef Adersberger
Norbert Tausch, M. Eng.
Duration: 1.11.2009–31.12.2014

Sponsored by:
Bundesministerium für Wirtschaft und Technologie
Contact:
Prof. Dr. Michael Philippsen
Phone: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Prototypical implementation of a new tool for quality assurance during software development

Modern software systems are getting increasingly complex with respect to functional, technical and organizational aspects. Thus, both the number of requirements per system and the degree of their interconnectivity constantly increase. Furthermore the technical parameters, e.g., for distribution and reliability are getting more complex and software is developed by teams that are not only spread around the globe and also suffer from increasing time pressure. Due to this, the functional, technical, and organizational control of software development projects is getting more difficult.

The ”Software Project Control Center” is a tool that helps the project leader, the software architect, the requirements engineer, or the head of development. Its purpose is to make all aspects of the development process transparent and thus to allow for better project control. To achieve transparence, the tool distills and gathers properties from all artifacts and correlations between them. It presents/visualizes this information in a way suitable for the individual needs of the users.

The Software Project Control Center unifies the access to relations between artifacts (traceability) and to their properties (metrics) within software development projects. Thus, their efficiency can be significantly increased. The artifacts, their relations, and related metrics are gathered and integrated in a central data store. This data can be analyzed and visualized, metrics can be computed, and rules can be checked.

For the Software Project Control Center project we cooperate with the QAware GmbH, Munich. The AIF ZIM program of the German Federal Ministry of Economics and Technology funded the first 30 months of the project.

The Software Project Control Center is divided into two subsystems: The integration pipeline that gathers traceability data and metrics from a variety of software engineering tools, and the analysis core, that allows to analyze the integrated data in a holistic way. Each subsystem is developed in a separate subproject.

The project partner QAware GmbH implements the integration pipeline. The first step was to define TraceML, a modeling language for traceability information in conjunction with metrics. The language contains a meta-model and a model library. TraceML allows to define customized traceability models in an efficient way. The integration pipeline is realized using TraceML as lingua franca in all processing steps: From the extraction of traceability information to its transformation and integrated representation. We used the Eclipse Modeling Framework to define the TraceML models on each meta-model level. Furthermore, we use the Modeling Workflow Engine for model transformations and Eclipse CDO as our model repository. A set of wide-spread tools for software engineering are connected to the integration pipeline including Subversion, Eclipse, Jira, Enterprise Architect and Maven.

The main research contribution of our group to this project is the analysis core, i.e., the design and realization of a domain-specific language for graph-based traceability analysis. Our Traceability Query Language (TracQL) significantly reduces the effort that is necessary to implement traceability analyses. This is crucial for both industry and the research community as lack of expressiveness and inefficient runtimes of other known approaches hinder the implementation of traceability analysis. TracQL eases not only the extraction, but also the analysis of traceability data using graph traversals that are denoted in a concise functional programming style. The language itself is built on top of Scala, a multi-paradigm programming language, and was successfully applied to several real-world industrial projects. In 2013, we increased its expressiveness by improving modularity and static typing.

2.11 Compiler-supported parallelization for multi-core architectures

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Dipl.-Inf. Tobias Werth
Start: 1.3.2007
Contact:
Dipl.-Inf. Tobias Werth
Phone: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

Several issues significantly retard the development of quicker and more efficient computer architectures. Traditional technologies can no longer contribute to offer more hardware speed. Basic problems are the divergent ratio of the latencies of memory access and CPU speeds as well as the heat and waste of energy caused by increasing clock rates.

Homogeneous and heterogeneous multi-core architectures were presented as a possible answer and offer enormous performance to the programmer. The decreasing clock rates help avoid most of the above problems, while the multiplied hardware can still deliver high performance since more arithmetic operations can be executed per time unit with less energy. Potentially, performance can increase even further by specialization of some hardware components. For example, often the latency problem is attacked with a multi-tiered memory hierarchy and lots of caches.

But there is no free lunch. It seems to be quite difficult to make multi-core architectures deliver their theoretically available performance into applications. Only with a lot of expertise in both the application domain and the specifics of the multi-core platform at hand and only with enough time to invest into tuning endeavors, one can make multi-core programs run fast.

From the point of view of a programming systems research group, there are - among others - the following open questions: What kind of support can a modern compiler offer to the programmer that develops applications for multi-core architectures? How much context knowledge is necessary in order to make reasonable decisions for parallelization? Which part of the available performance can be used by the programmer with a reasonable amount of effort without detailed knowledge about the features and quirks of the underlying architecture? Which tools are necessary for debugging and for finding bottlenecks in applications that run on multi-core architectures? How can they be designed?

It is the intention of this research project to answer these questions for a restricted application domain. We have selected the Lattice-Boltzmann-Method (LBM) that is mostly used in computational fluid dynamics as our problem domain. Caused by its lattice structure and its manageable number of data dependencies between the single lattice points, it is comparatively straightforward how to parallelize it. Hence, our compiler research can focus on the above questions.

The heterogeneous CellBE architecture is selected as target architecture due its good performance on a single chip. It consists of a PowerPC core (PPU) and eight Synergistic Processing Units (SPUs), which can do computations in parallel. The programming model Cilk was further developed to allow a robust and efficient execution on the SPUs. We made it much easier to debug the execution while stealing functions from remote SPUs. Beside that, the source to source transformation was rewritten to produce code for both PPU and SPU in a simpler and more generic way.

In 2012, we focused on graphic cards (GPUs) as a second target architecture. GPUs offer a lot more performance than ordinary CPUs, however achieving peak performance may be difficult. For data parallel problems, the performance can be achieved using Cuda (NVidia) or OpenCL (AMD) relatively easy. However, it is much more difficult to port task parallel problems with reasonable performance to the GPU, which is one of the goals on our roadmap. Thus, we design, implement and compare various load balancing algorithms. In 2012 we designed a first approach with hierarchical queues under the principle of work donation.

In 2013, while further developing the load balancing algorithms for the GPU, we also targeted our work towards the Intel XeonPhi processor. With its many-core architecture and large register sets (and thus the ability to issue vector instructions on multiple data), the XeonPhi processor is a new challenge for load balancing algorithms. In practice, we extended and adopted Cilk for the XeonPhi such that we can automatically merge functions during the source-to-source transformation. This increases the Intel compiler's chances to automatically parallelize. We implemented several analyses that not only increase the number of candidate functions for merging but also avoid (or at least handle) divergence in those merged functions.

2.12 WEMUCS - Techniques and tools for iterative development and optimization of software for embedded multicore systems

Project manager:
Prof. Dr. Michael Philippsen
Project participants:
Demian Kellermann, M. Sc.
Dr.-Ing. Norbert Oster
PD Dr. Ronald Veldema
Start: 15.10.2012
Sponsored by:
IuK Bayern
Contact:
Prof. Dr. Michael Philippsen
Phone: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

The importance of multicore processors in embedded systems is rising, as these processors offer high performance while maintaining low power consumption. Developing parallel software for these platforms poses new challenges for many industrial sectors, because established tools and software are not aware of parallel processing. The efficient development, optimization and testing of multicore-software are still open research problems.

The multi-partner WEMUCS project provides new tools and methods for efficient, iterative development, optimization and testing of multicore software. Innovative tools and technologies for modelling, simulation, visualization, tracing, and testing are developed and integrated into a tool chain. Using case studies from different industries (automotive, telecommunications, industry automation), these tools are evaluated and improved.

As part of the WEMUCS project, we are developing new ways to model parallel processes, making it possible to identify problems and reproduce them in a testing environment. We also explore what chances a compiler analysis has to identify and report problematic parallelism to the test environment.

The project is a contribution of the Programming Systems Group to the IZ ESI (http://www.esi.uni-erlangen.de/).

3 Publications 2013

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Combining Lock Inference with Lock-Based Software Transactional Memory . In: Springer (Ed.) : Proceedings of the 26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) (26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) Santa Clara, California, USA). 2013, pp -.

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Compiler-Guided Identification of Critical Sections in Parallel Code . In: De Bosschere, Koen ; Jhala, Ranjit (Ed.) : Proceedings of the 22nd International Conference on Compiler Construction (International Conference on Compiler Construction Italy, Rome 21.-22.03.2013). 2013, pp 204-223. - ISBN 978-3-642-37050-2

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Reduktion von False-Sharing in Software-Transactional-Memory . In: GI (Ed.) : Proceedings of the 25th Workshop on Parallel Systems and Algorithms (PARS 2013) (25th Workshop on Parallel Systems and Algorithms (PARS 2013) Erlangen, Germany 11.-12.04.2013). 2013, pp 70-79.

– Lautenschlager, Florian: Design for Diagnosability: Wie mache ich Software diagnostizierbar? In: the coaches (Org.) : German Testing Day 2013 (German Testing Day 2013 Munich, Germany 12.11.2013). 2013, pp -.

– Loeffler, Christoffer ; Mutschler, Christopher ; Philippsen, Michael: Evolutionary Algorithms that use Runtime Migration of Detector Processes to Reduce Latency in Event-Based Systems . In: IEEE Computer Society (Ed.) : Proceedings of the 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)

(2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013) Torino, Italy June 25-27, 2013). 2013, pp 31-38. - ISBN 978-1-4673-6382-2

– Mutschler, Christopher: Apparatus, Method, and Computer Program for Processing Out-of-Order Events . Schutzrecht EP13153525.4 patent application (31.01.2013)

– Mutschler, Christopher ; Philippsen, Michael: Distributed Low-Latency Out-of-Order Event Processing for High Data Rate Sensor Streams . In: IEEE Computer Society (Ed.) : Proceedings of 27th International Parallel and Distributed Processing Symposium (27th IEEE International Parallel & Distributed Processing Symposium (IPDPS) Boston, Massachusetts, USA May 20-24, 2013). 2013, pp 1133-1144. - ISBN 978-0-7695-4971-2

– Mutschler, Christopher ; Witt, Nicolas ; Philippsen, Michael: Do Event-Based Systems have a Passion for Sports? (Best Poster/Demo Audience Award) . In: ACM (Ed.) : Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, pp 331-332. - ISBN 978-1-4503-1758-0

– Mutschler, Christopher ; Philippsen, Michael: Dynamic Low-Latency Distributed Event Processing of Sensor Data Streams . In: Gesellschaft für Informatik e.V. (Ed.) : Proceedings of the 25th Workshop on Parallel Systems and Algorithms (PARS 2013), ISSN 0177-0454 (25th Workshop on Parallel Systems and Algorithms (PARS 2013) Erlangen, Germany 11.-12.04.2013). 2013, pp 5-14.

– Mutschler, Christopher ; Philippsen, Michael: Reliable Speculative Processing of Out-of-Order Event Streams in Generic Publish/Subscribe Middlewares . In: ACM (Ed.) : Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, pp 147-158. - ISBN 978-1-4503-1758-0

– Mutschler, Christopher ; Philippsen, Michael: Runtime Migration of Stateful Event Detectors with Low-Latency Ordering Constraints . In: IEEE (Ed.) : Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (9th International Workshop on Sensor Networks and Systems for Pervasive Computing San Diego, CA, USA 18.-22.03.2013). 2013, pp 609-614. - ISBN 978-1-4673-5075-4

– Mutschler, Christopher ; Ziekow, Holger ; Jerzak, Zbigniew: The DEBS 2013 Grand Challenge . In: ACM (Ed.) : Proceedings of the 7th ACM International

Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, pp 289-294. - ISBN 978-1-4503-1758-0

– Otto, Stephan ; Edelhäußer, Thorsten ; Witt, Nicolas ; Völker, Matthias ; Voll, David ; Mutschler, Christopher: Apparatus, Method and Computer Program for Providing a Virtual Boundary . Schutzrecht EP13156961.8 patent application (27.02.2013)

– Philippsen, Michael ; Tillmann, Nikolai ; Brinkers, Daniel: Double inspection for run-time loop parallelization . In: Rajopadhye, S. ; Strout, M. Mills (Ed.) : Proceedings of the 24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) (24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) Fort Collins, Colorado, USA 08.-10.09.2011). Berlin : Springer-Verlag Berlin Heidelberg, 2013, pp 46-60. (Lecture Notes in Computer Science (LNCS) Vol. 7146) - ISBN 978-3-642-36035-0

– Veldema, Ronald ; Philippsen, Michael: Language and Runtime Techniques for better Model Checking Efficiency of Parallel Programs . In: Springer (Ed.) : Proceedings of the 26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013), Poster Session (26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) Santa Clara, California, USA 25.09. - 27.09.2013). 2013, pp -.

– Werth, Tobias ; Schreier, Silvia ; Philippsen, Michael: CellCilk: Extending Cilk for heterogeneous multicore platforms . In: Rajopadhye, S. ; Strout, M. Mills (Ed.) : Proceedings of the 24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) (24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) Fort Collins, Colorado, USA 08.-10.09.2011). Berlin : Springer-Verlag Berlin Heidelberg, 2013, pp 91-105. (Lecture Notes in Computer Science (LNCS) Vol. 7146) - ISBN 978-3-642-36035-0

– Wolf, Carolin ; Dotzler, Georg ; Veldema, Ronald ; Philippsen, Michael: Object Support for OpenMP-style Programming of GPU Clusters in Java . In: IEEE Computer Society (Ed.) : Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013) (27th International Conference on Advanced Information Networking and Applications Workshops Barcelona, Spain 25.03.-28.03.2013). 2013, pp 1405-1410. - ISBN 978-1-4673-6239-9

4 Exam theses (german only)

- Master Thesis: Konzeption und prototypische Realisierung eines Integrationsframeworks für Entwicklungswerkzeuge. Bearbeiter: Julia Mailova (beendet am 24.4.2013); Betreuer: Dr.-Ing. Martin Jung; Hon.-Prof. Dr.-Ing. Detlef Kips
- Master Thesis: Entfernung überflüssiger Thread-Synchronisationen in NGAPL. Bearbeiter: Philipp Weissmann (beendet am 26.4.2013); Betreuer: PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen
- Diplomarbeit: Re-Engineering des Werkzeugs .gEAr zum Eclipse-Plugin für die aktuelle Sprachversion von Java. Bearbeiter: Waldemar Krawtschuk (beendet am 15.08.2013); Betreuer: Dr.-Ing. Norbert Oster; Prof. Dr. Michael Philippsen
- Bachelor Thesis: Evaluation und prototypische Implementierung eines Parsers für Ereignisbeschreibungssprachen in Echtzeitlokalisierungssystemen. Bearbeiter: Cerny Patrick (beendet am 11.11.2013); Betreuer: Dipl.-Inf. Christopher Mutschler; Prof. Dr. Michael Philippsen; Dr.-Ing. Thorsten Edelhäußer
- Bachelor Thesis: Entwicklung einer adaptiven HMI-Schnittstelle zur Erkennung von Gesten auf Basis hochpräziser Echtzeitlokalisierung. Bearbeiter: Dennis Salzner (beendet am 2.12.2013); Betreuer: Dipl.-Inf. Christopher Mutschler; Dr.-Ing. Stephan Otto; Prof. Dr. Michael Philippsen
- Master Thesis: Entwicklung eines Werkzeugs zur Identifizierung vergleichbarer Code-Modifikationen in Software-Archiven. Bearbeiter: Christoph Romstöck (beendet am 3.12.2013); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen
- Studienarbeit: Entwurf und Implementierung eines Visualisierungs- und Explorationskonzeptes für multi-relationale Pseudographen am Beispiel der Software-Traceability. Bearbeiter: Peter Kranz (beendet am 12.12.2013); Betreuer: Norbert Tausch, M. Eng.; Prof. Dr. Michael Philippsen