An Experience Report: Efficient Analysis using Soufflé

Bernhard Scholz Pavle Subotić Herbert Jordan
Padmanabhan Krishnan Raghavendra Kagalavadi Ramesh Cristina Cifuentes

Oracle Labs, Brisbane, Australia

Soufflé is an open-source programming framework for static program analysis. It enables the analysis designer to express static program analysis on very large code bases such as a points-to analysis for the Java Development Kit (JDK)[1] which has more than 1.5 million variables and 600 thousand call sites. Soufflé employs a Datalog-like language as a domain specific language for static program analysis. Its finite domain semantics lends to efficient execution on parallel hardware using various levels of program specializations. A specialization hierarchy is applied to a Datalog program. As a result, highly specialized and optimized C++ code is produced that harvests the computational power of modern shared-memory/multi-core computer architectures [2, 1].

We have been using Soufflé to explore and develop vulnerability detection analyses on the Java platform, using JDK 7, 8 and 9. These vulnerability detection analyses make use of points-to analysis (reusing parts of the DOOP framework), taint analysis, escape analysis, and other data flow-based analyses. In this talk we report on the types of analyses used, the sizes of the input relations and computed relations, as well as the the runtime and memory requirements for the analyses of such large codebases.

For the program specialization, we use several translation steps. In each translation step, new optimization opportunities open up that would not be able to exploit in the previous translation step. The first translation uses a Futamura projection to translate a declarative Datalog program to an imperative relational program for an abstract machine which we call the Relational Algebra Machine (RAM). The RAM program contains relational algebra operations to compute results produced by clauses, relation management operations to keep track of previous, current and new knowledge in the semi-naïve evaluation, and imperative constructs including statement composition for sequencing the operations, and loop construction with loop exit condition to express fixed-points computations for recursively-defined relations. It also has support for parallelism. The next translation steps, translates the optimized RAM program into a C++ program that uses meta-programming techniques with templates. The last translation step, is performed by a C++ program that compiles the C++ program to a executable binary. Operations for emptiness and existence checks, range queries, insertions and unions are highly efficient because portions of the operations are pushed from runtime to compile-time using meta-programming techniques.

We now outline some of the novel aspects that are in the implementation of Soufflé. The first is related to indices. Since indices are costly, a minimal set of indices for a given relation is desired. We employ a discrete optimization problem to minimize indices creating only the required indices for the execution is required and hence avoiding redundancies. The second is the choice of data-structures to represent large relations. Our experience showed that only b-trees and Tries are viable data-structures for large relations. These data-structures support membership tests and range queries in O(log(n)) time. The third aspect relates to exploiting caches and parallel cores. B-trees are cache efficient when keys are spatially close in memory. However, updating B-trees in parallel is non-trivial and involves locking when the tree needs to be rebalanced. Towards overcoming this we use optimistic locking techniques common in databases.

Based on our experience there are two items that we wish to explore further; viz., a query planner and support for better parallelism. Currently, Soufflé is not fully declarative. After finding a specification, the specification has to be altered to make it scalable for large code bases. This is achieved by either providing an efficient ordering of the atoms in the body of a clause and/or providing a schedule for recursive clauses.

[1] Java and JDK are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Without a deep insight into the intrinsics of the Datalog machinery, scheduling is not straightforward for the uninitiated. We observe as practitioners, automating the scheduling of clauses is of paramount importance to obtain high-performance Datalog program. Currently, we have implemented a preliminary scheduler that uses feedback-directed compilation but has insufficient performance metrics for its decision. Having a fully fledged scheduler that works for large-scale problems would reduce the effort for the programmer to fine-tune programs, i.e., the programmer can focus on the specification task rather than the implementation details.

The level of parallelism currently implemented in Soufflé is limited to shared memory architectures. Features that need further exploration is the parallelisation of Soufflé in distributed systems such as a map/reduce infrastructure for the cloud and heterogeneous cluster including GPGPUs.

References

[1] Bernhard Scholz, Herbert Jordan, and Pavle Subotić. Soufflé: On synthesis of Datalog for program analyzers. In *International Conference on Computer Aided Verification (CAV)*, To appear 2016.

[2] Bernhard Scholz, Herbert Jordan, Pavle Subotić, and Till Westmann. On fast large-scale program analysis in Datalog. In *Proceedings of the 25th International Conference on Compiler Construction (CC)*, pages 196–206. ACM, 2016.

Speaker Biography

Cristina Cifuentes is the Director of Oracle Labs Australia and an Architect at Oracle. Headquartered in Brisbane, the focus of the Lab is Program Analysis as it applies to finding vulnerabilities in software and enhancing the productivity of developers worldwide.

Prior to founding Oracle Labs Australia, Cristina was the Principal Investigator of the Parfait static code analysis project at Sun Microsystems, then Oracle. Today, Oracle Parfait has become the defacto tool used by thousands of Oracle developers for defect and vulnerability detection in real-world, commercially sized C/C++/Java applications. The success of the Parfait tool is founded on the pioneering work in advancing static program analysis techniques carried out by Cristinas team of Researchers and Engineers at Oracle Labs Australia.

Cristina's passion for tackling the big issues in the field of Program Analysis began with her doctoral work in binary decompilation at Queenslands University of Technology, followed by later work on binary translation in the UQBT and Walkabout projects. Prior to her work at Oracle and Sun Microsystems, Cristina held teaching posts at major Australian Universities, co-edited Going Digital, a landmark book on cybersecurity, and served on the executive committees of ACM SIGPLAN and IEEE Reverse Engineering.

Cristina continues to play an active role in the international programming language, compiler construction and software security communities. On the weekends, she channels her interests into mentoring young kids through the CoderDojo network.