A GSoC 2013 Final Report

Enhancing Giri: Dynamic Slicing in LLVM

Mingliang Liu, *Tsinghua University*

Dynamic program slicing has been used in many applications. Giri was a research project from UIUC, which implemented the dynamic backward slicing in LLVM. It was selected as the Google Summer of Code 2013. I did several improvements to Giri directed by Dr. Swarup: *1*) Update the code to LLVM mainline and make it robust, *2*) Reduce the trace size, *3*) Make the Giri thread-aware (pthread only), *4*) Improve the performance of run-time.

Introduction

Dynamic program slice contains statements that influence the value of a variable occurrence for special program inputs [1]. The tradition program slicing was called static program slicing, which was firstly proposed by Weiser [7]. There are many applications that use (or could benefit from) dynamic slicing, both by research and industry organizations (e.g. Microsoft, IBM). For example, it's long been used in software debugging model [2, 4] and testing [3]. However, as far as I know, there is no public available dynamic slicing tool in GCC or Open64.

Sahoo et. al. from UIUC use dynamic program slicing to generate likely invariants for automated software fault localization [5]. They implemented the dynamic program slicing code, called Giri, in LLVM compiler infrastructure for research purpose based on LLVM 2.6. It also maps LLVM IR statements to source-level statements for its output using the debug metadata.

Progress

Our goal of GSoC is to make the Giri code update to the latest LLVM version and improve the performance, and/or reduce the tracing cost. There are several things we did in this summer:

1. **Update the code to LLVM mainline.**
   - Release a v3.1 version which works with LLVM 3.1
   - Make the code update to the latest LLVM code (3.4)
2. **Make the giri run-time library thread safe and the slicing thread aware**. The events of all processes and threads get thrown together in one trace file. Trace records indicate which thread is performing a particular operation using the thread id (`pthread_t`). The trace files are written with lockes so that the race condition will not break the sequences.
3. **Improving the giri run-time performance.** For example, fix the bug of `mmap` at the end of the tracing. We compute dynamically the cache size for how many trace records to hold in memory before flushing to disk.
4. **Write dozens of unit tests and test several real programs.** There is also a simple test framework which will try all the unit tests at the top level of `giri/test/`.
5. **Reduce the trace size.** We truncate the file at the end of tracer according to its real size.

6. **Write documents for the project.** There are several sample pages:
    - The Wiki Page Home.
    - How to Compile Giri.
    - Hello World!
    - Example Usage With An Example.

TODO List

1. Improve the performance of locking mechanism at the runtime
2. Add tool/Tracer.cpp to the make list. It was removed when upgrading to LLVM 3.4
3. Pass all the test programs of the test/ directory
4. Try large test cases like nginx, squid, etc
5. Consider more special function calls in TracingNoGiri::visitSpecialCall() function of the tracing pass
6. Parallel the code writing the entry cache to trace file and adding entry to the cache.

Contacts

The Giri code is still under active development. Dr. Swarup will direct me in the future to make the Giri code better. We publish our code at https://github.com/liuml07/giri [6]. For more information, please visit the homepage above.

References

[1] H AGRAWAL. Dynamic Program Slicing. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 1990.

[2] Hiralal Agrawal, Richard A Demillo, and Eugene H Spafford. Debugging with dynamic slicing and backtracking. *Software: Practice and Experience*, 23(6):589–616, 1993.

[3] Hiralal Agrawal, Joseph R Horgan, Edward W Krauser, and Saul A London. Incremental regression testing. In *Software Maintenance, 1993. CSM-93, Proceedings., Conference on*, pages 348–357. IEEE, 1993.

[4] Tibor Gyimóthy, Árpád Beszédes, and István Forgács. An efficient relevant slicing method for debugging. In *Software EngineeringESEC/FSE99*, pages 303–321. Springer, 1999.

[5] Swarup Kumar Sahoo, John Criswell, Chase Geigle, and Vikram Adve. Using likely invariants for automated software fault localization. In *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems*, ASPLOS '13, pages 139–152, New York, NY, USA, 2013. ACM.

[6] Swarup Kummar Sahoo and Mingliang LIU. Giri: Dynamic Program Slicing in LLVM. `https://github.com/liuml07/giri`, 2013.

[7] M. Weiser. Program slicing. In *Proceedings of the 5th International Conference on Software Engineering*, ICSE, pages 439–449. IEEE, 1981.