TECHNISCHE UNIVERSITÄT DRESDEN

Heinrich Kießling and Martin Nowack

TU Dresden, Germany

KLEE's Solver Chain Revisited - *Opportunities for Improvement?*

London, 19. April 2018

Outline

Intro - Query Caching (in KLEE)

- Global Caching

- Parallel Portfolio Solving

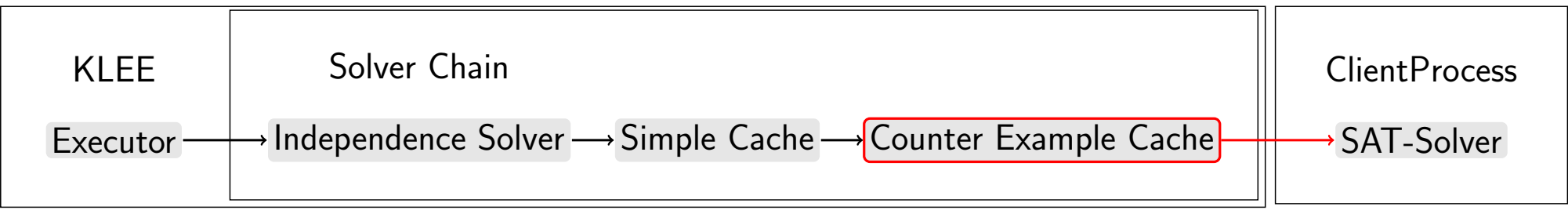
Query Caching (in KLEE)

- SAT (SMT) solving is NP-complete
  $\rightarrow$ 92% of runtime for SAT solving[1]

$\Rightarrow$ Caching as space-time trade-off
  $\searrow$ 41% of runtime for caching and SAT solving[1]

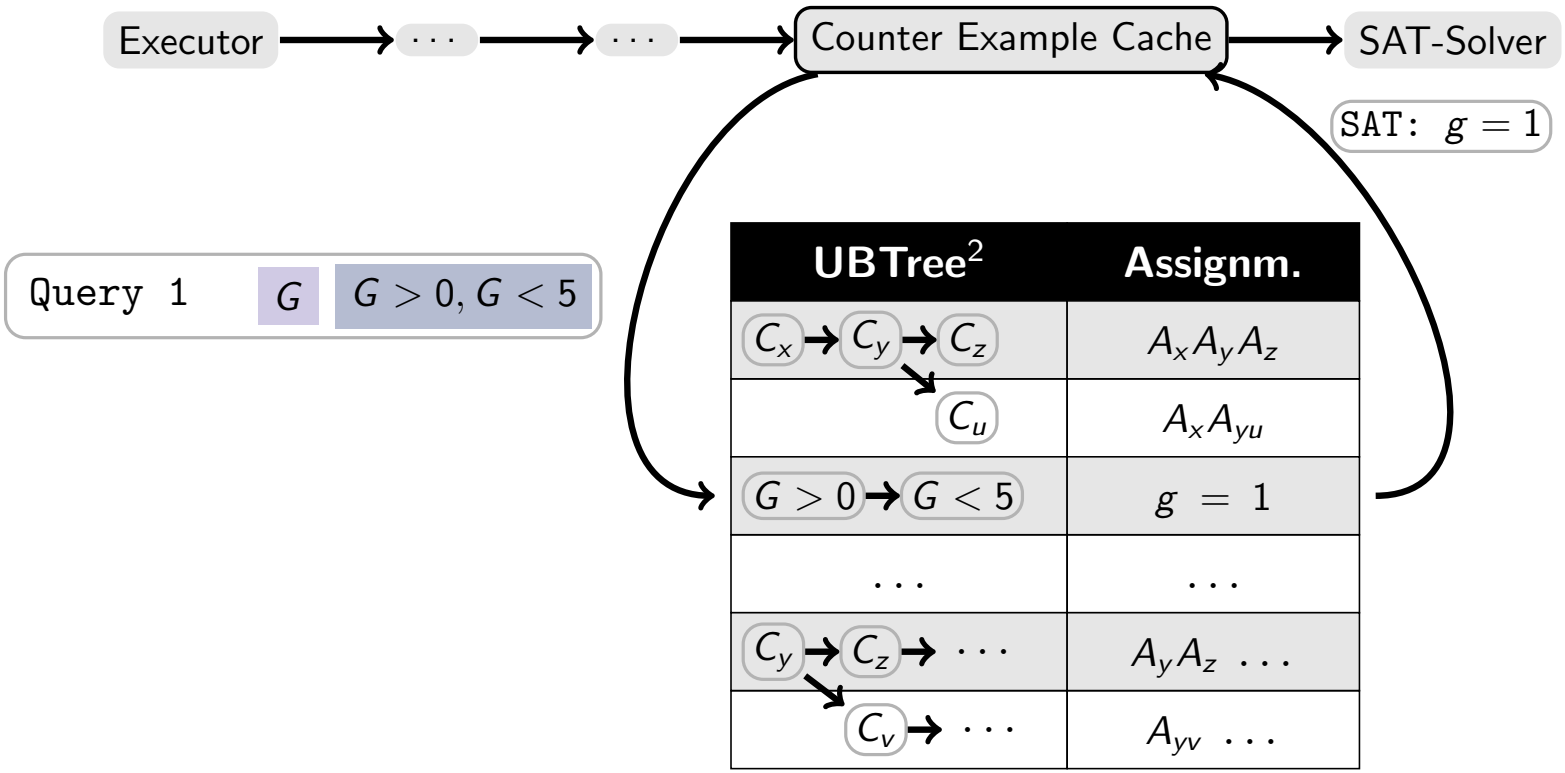
1) **KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs**, Proceedings of OSDI 2008, Cadar et al. [4]

Query Caching (in KLEE)
Solver Chain



Query Caching (in KLEE)
Counter Example Cache

Reuse of concrete assignments via sub-/superset matching[2]



Global Caching
Idea

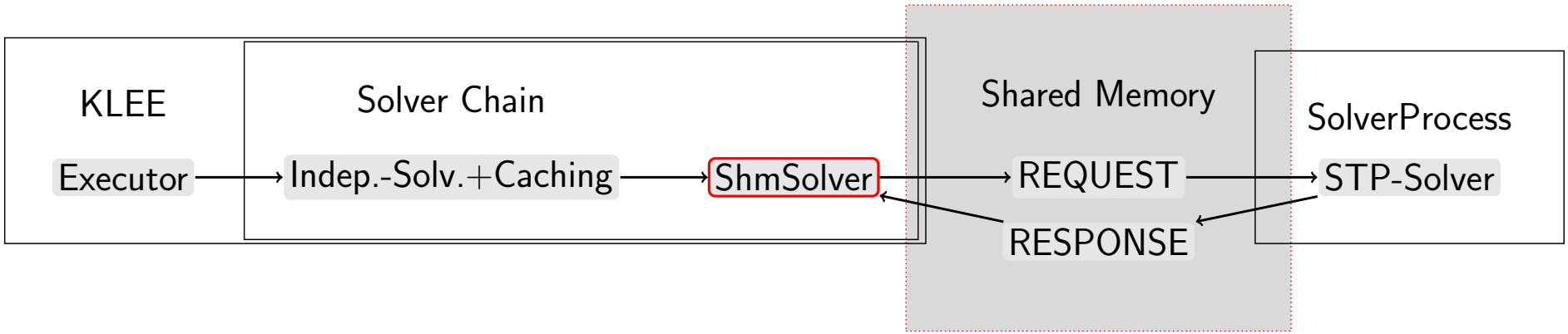
Reuse of cached assignments:

1. across runs? → Regression testing
2. across runs with different configurations of KLEE? → Coverage
3. across runs of different programs?

→ GNU coreutils reuse code for:

- argument parsing
- IO handling
- error handling
- configuration
- shared types

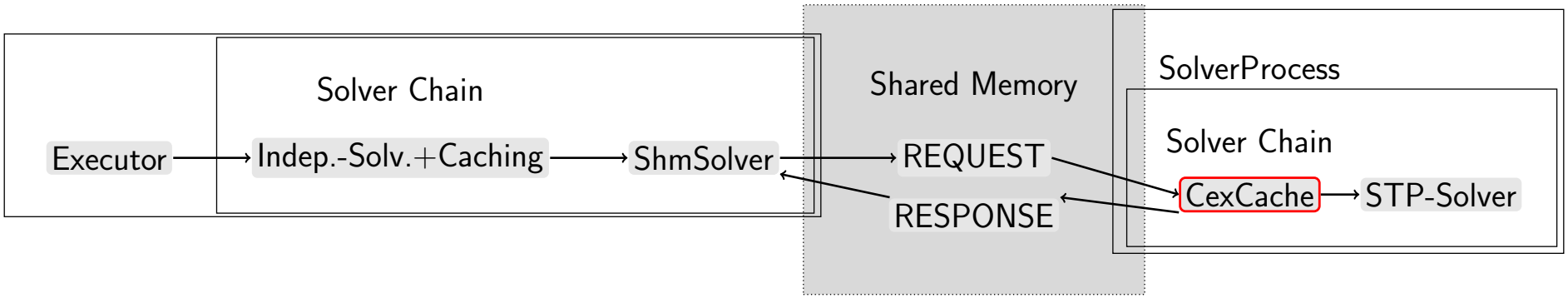
Global Caching
Architecture



- 3rd party solver connected via new IPC [3]

3) **New IPC-infrastructure based on cap'n'proto**, Nowack [14]
4) **EXE: automatically generating inputs of death**, Cadar et al. [5]
5) **Green: Reducing, Reusing and Recycling Constraints in Program Analysis**, Jia et al. [10], Visser et al. [18]

Global Caching
Architecture



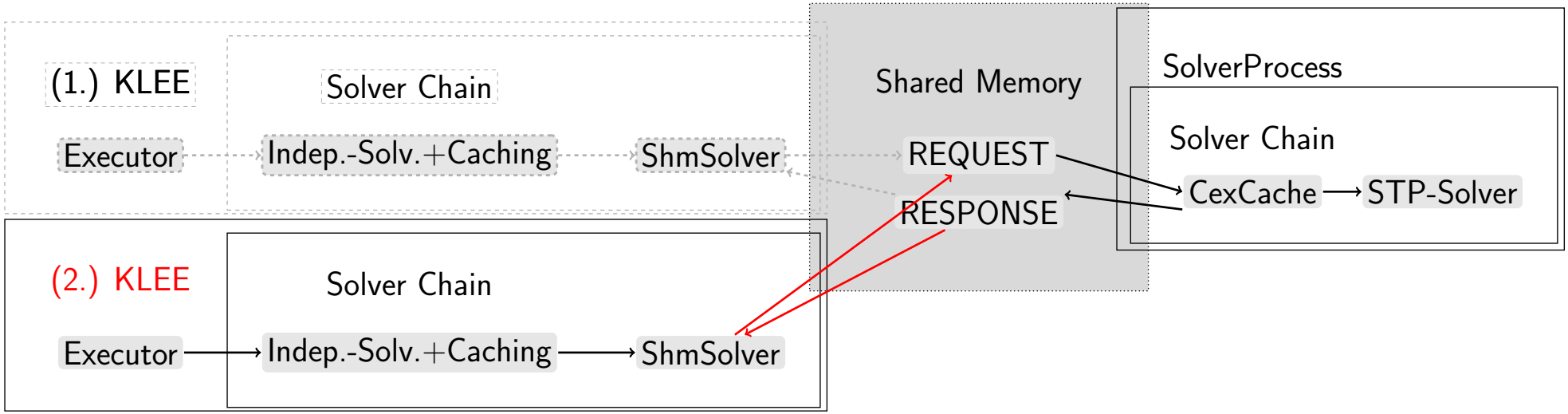
- 3rd party solver connected via new IPC [3]

  > Decouple cache and solver

---

3) **New IPC-infrastructure based on cap'n'proto**, Nowack [14]
4) **EXE: automatically generating inputs of death**, Cadar et al. [5]
5) **Green: Reducing, Reusing and Recycling Constraints in Program Analysis**, Jia et al. [10], Visser et al. [18]

Global Caching
Architecture



- 3rd party solver connected via new IPC [3]

  > Decouple cache and solver

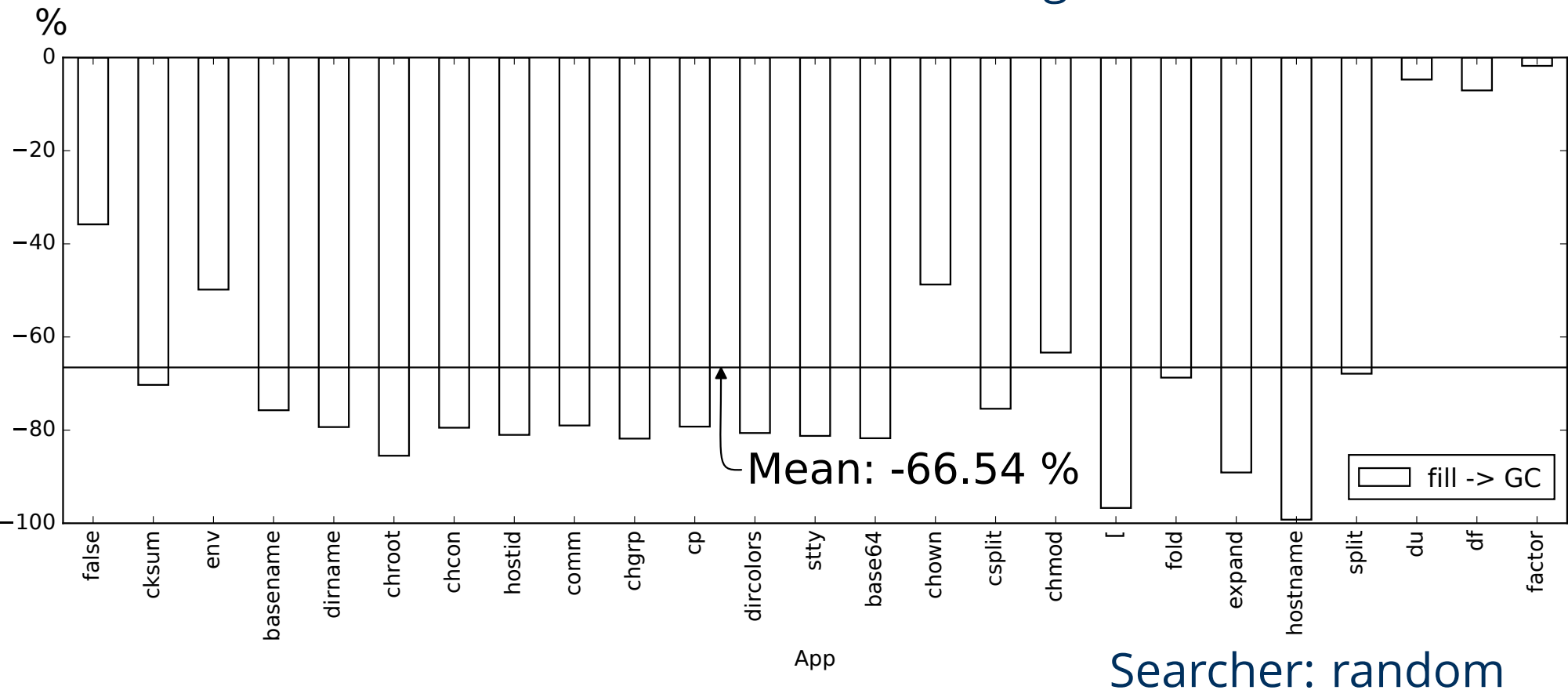
⇒ Global reuse of concrete assignments (other literature[4],[5])

3) **New IPC-infrastructure based on cap'n'proto**, Nowack [14]
4) **EXE: automatically generating inputs of death**, Cadar et al. [5]
5) **Green: Reducing, Reusing and Recycling Constraints in Program Analysis**, Jia et al. [10], Visser et al. [18]

Global Caching

Evaluation - same configuration twice

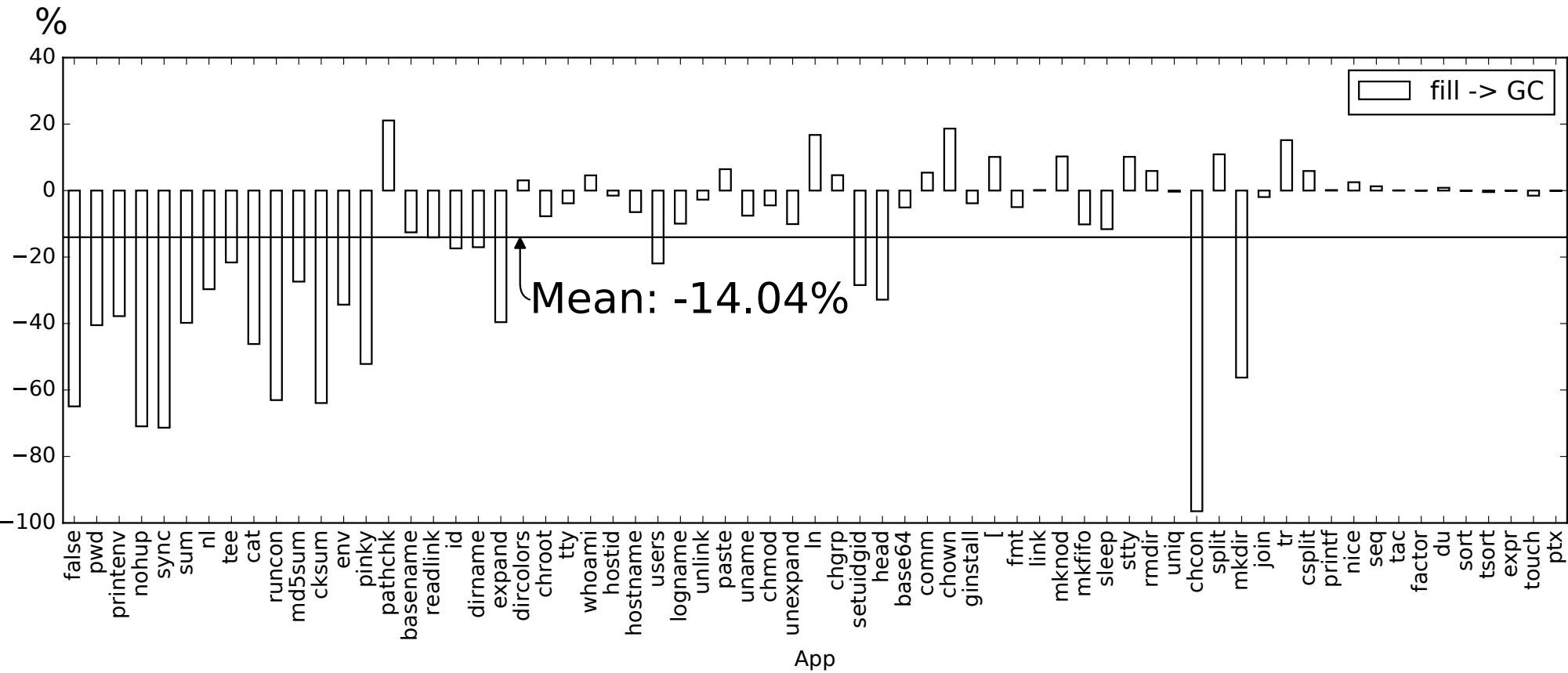
Relative Solver Time savings



Searcher: random

Global Caching

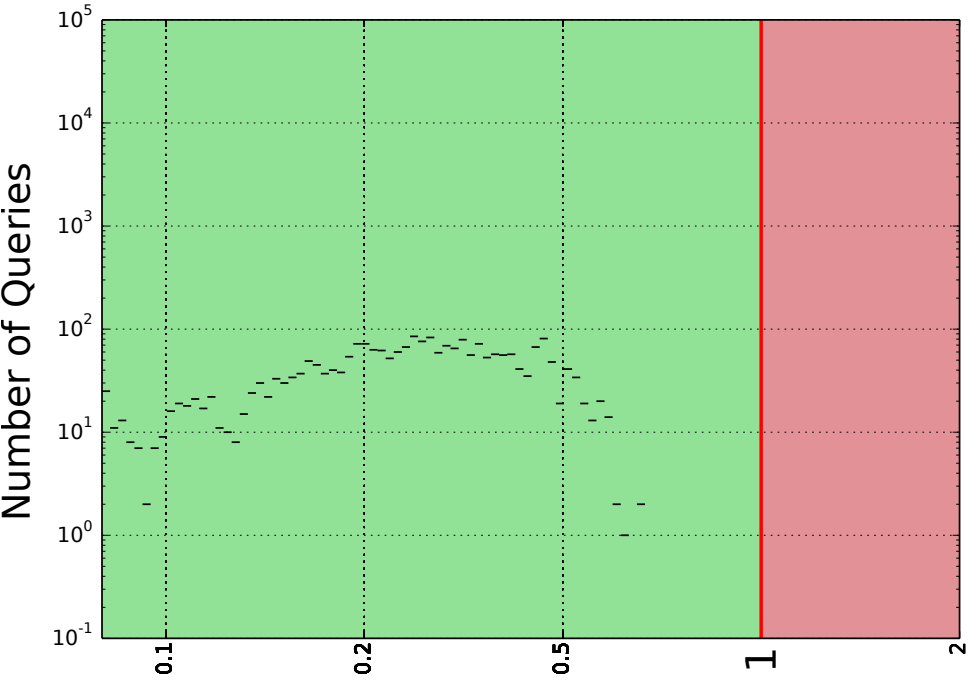
Evaluation - 1. BFS → 2. DFS searcher



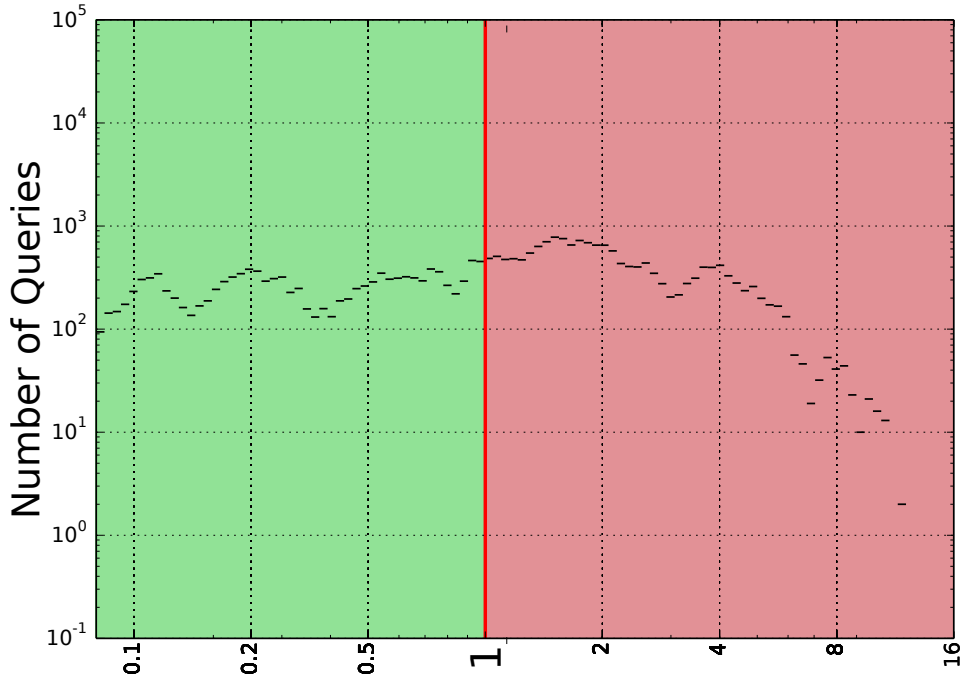
Relative Solver Time savings of second run compared to first run .

Global Caching
+ Parallel Portfolio Solving
Motivation

pathchk



tr



ratio = CacheTime / SolverTime

Parallel Portfolio Solving
Idea

Annual ranking in SMT-COMP[6]:

- STP[7]
- Z3[8]
- Boolector[9]
- Yices[10]
- CVC[11]

---

6) **SMT-COMP'16**, `http://smtcomp.sourceforge.net/2016` Conchon et al. [6]
7) **A Decision Procedure for Bit-Vectors and Arrays**, (STP) Ganesh and Dill [8]
8) **Z3: An Efficient SMT Solver**, Moura and Bjørner [13]
9) **Boolector: An efficient SMT solver for bit-vectors and arrays**, Brummayer and Biere [3]
10) **The yices smt solver**, Dutertre and De Moura [7]
11) **CVC4**, in *Computer Aided Verification* Barrett et al. [2]

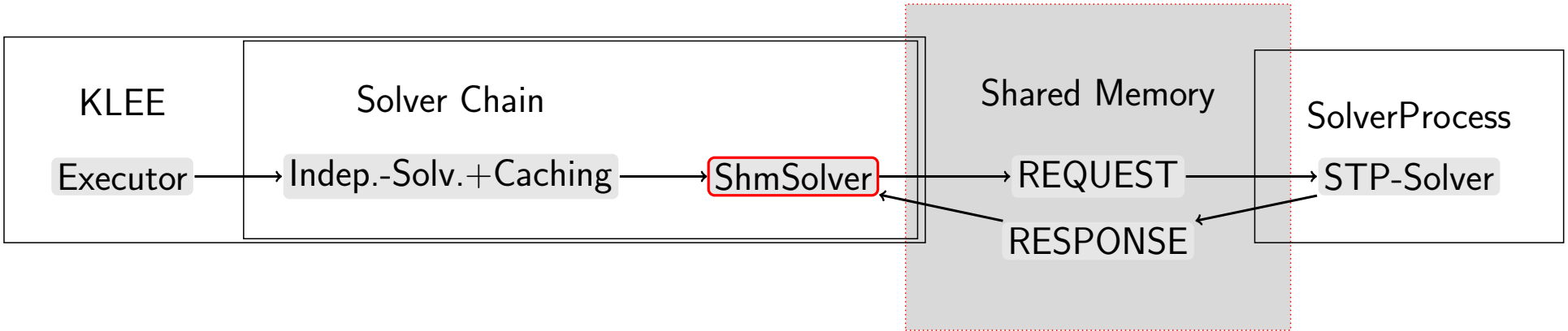
Parallel Portfolio Solving
Idea

Annual ranking in SMT-COMP[6]:

- STP[7]
- Z3[8]
- Boolector[9]

6) **SMT-COMP'16**, `http://smtcomp.sourceforge.net/2016` Conchon et al. [6]
7) **A Decision Procedure for Bit-Vectors and Arrays**, (STP) Ganesh and Dill [8]
8) **Z3: An Efficient SMT Solver**, Moura and Bjørner [13]
9) **Boolector: An efficient SMT solver for bit-vectors and arrays**, Brummayer and Biere [3]
10) **The yices smt solver**, Dutertre and De Moura [7]
11) **CVC4**, in *Computer Aided Verification* Barrett et al. [2]

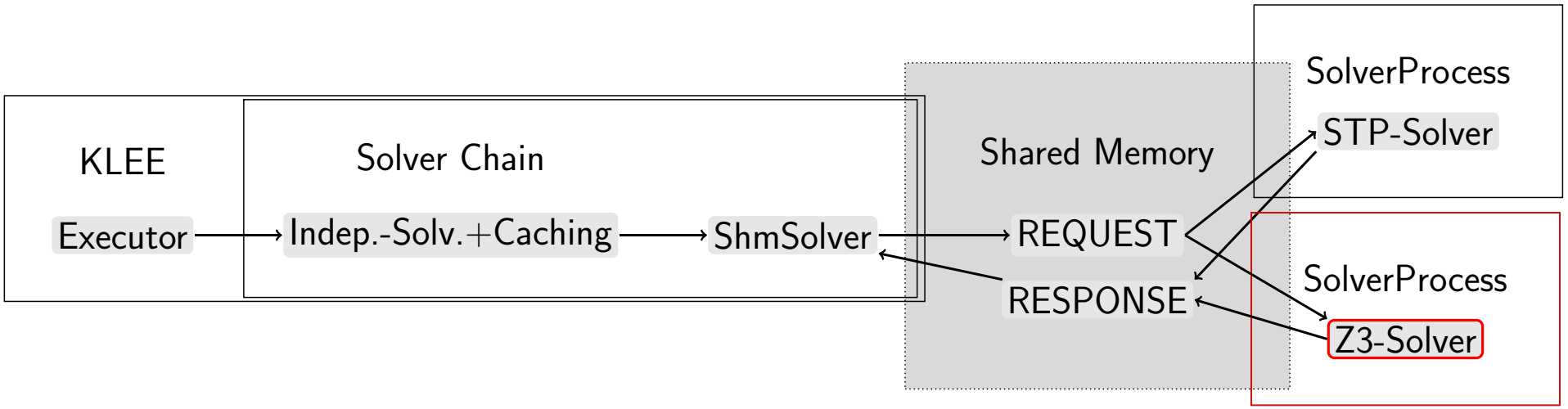
Parallel Portfolio Solving
Architecture



- ✓ SHM-based IPC protocol
- ✓ Handle solver crashes / timeouts.
- ✓ Portfolio of Solvers[12,13]

12) **metaSMT: A unified interface to SMT-LIB2**, Riener et al. [17]
13) **Multi-solver Support in Symbolic Execution**, (KLEE), Palikareva and Cadar [16]

Parallel Portfolio Solving
Architecture



- ✓ SHM-based IPC protocol
- ✓ Handle solver crashes / timeouts.
- ✓ Portfolio of Solvers[12,13]

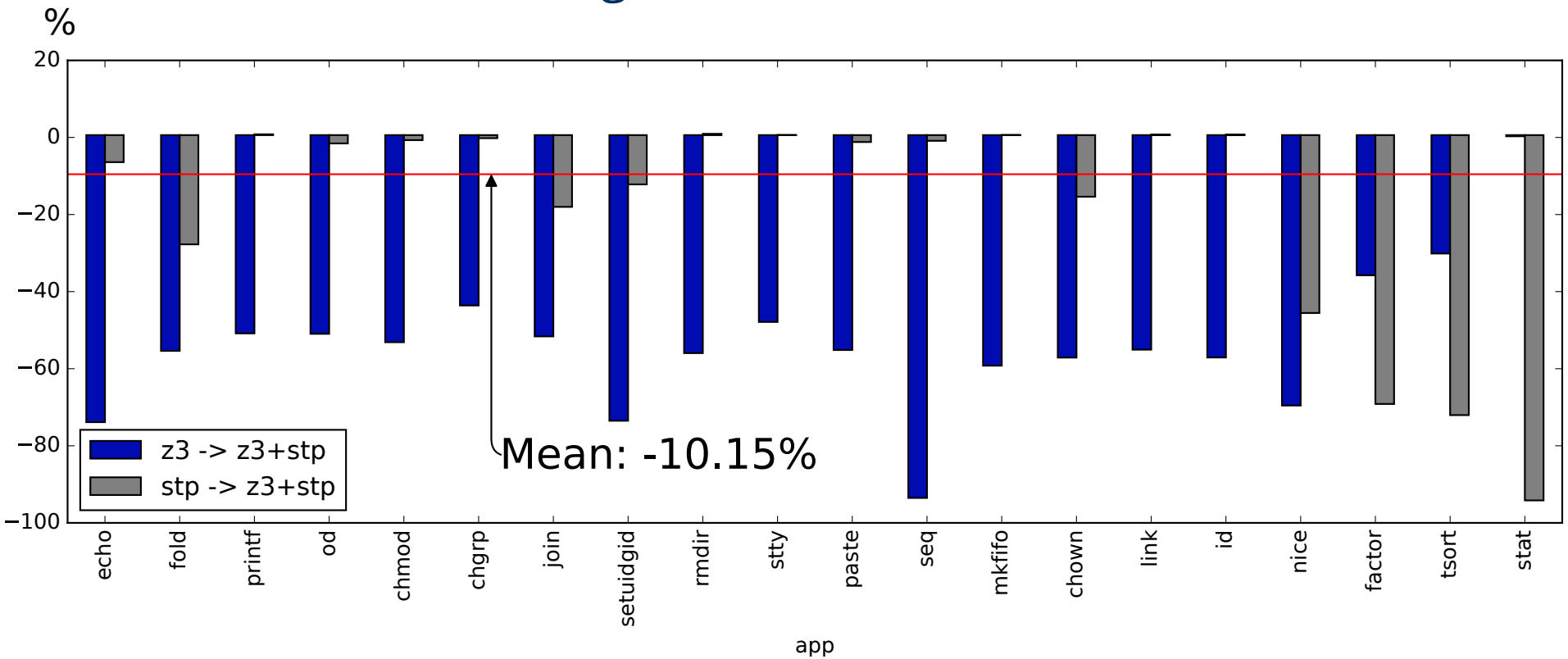
⇒ Parallel Portfolio Solving ⇒ More resource utilization

12) **metaSMT: A unified interface to SMT-LIB2**, Riener et al. [17]

13) **Multi-solver Support in Symbolic Execution**, (KLEE), Palikareva and Cadar [16]

Parallel Portfolio Solving
Evaluation

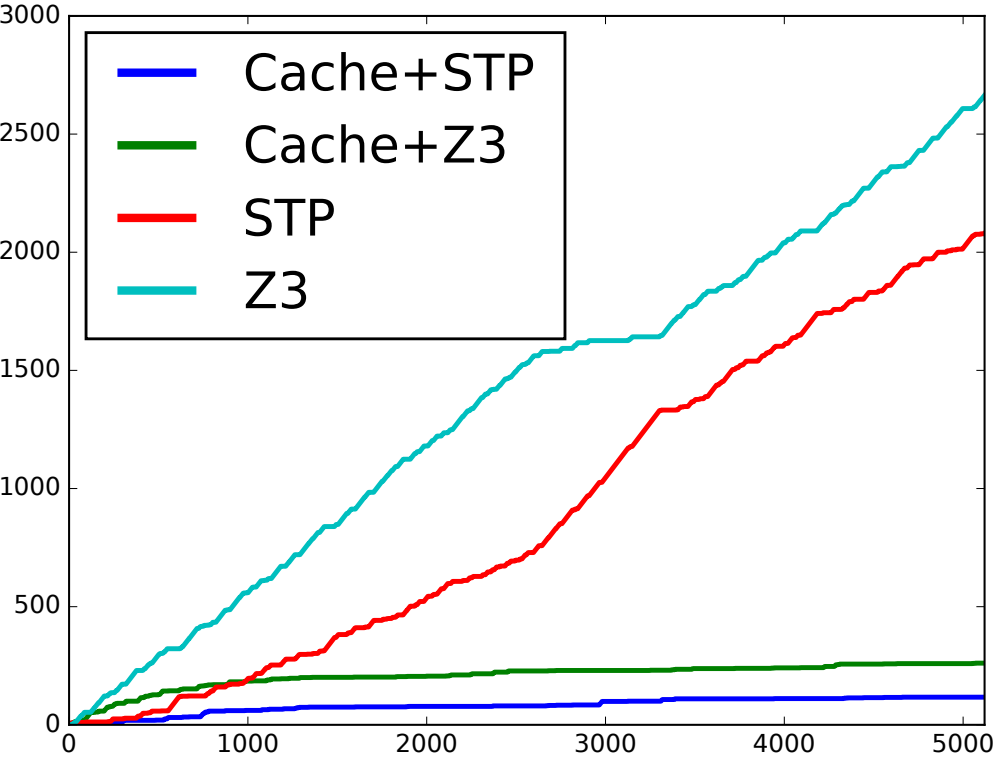
Solver Time savings STP+Z3-Portfolio vs. STP vs. Z3



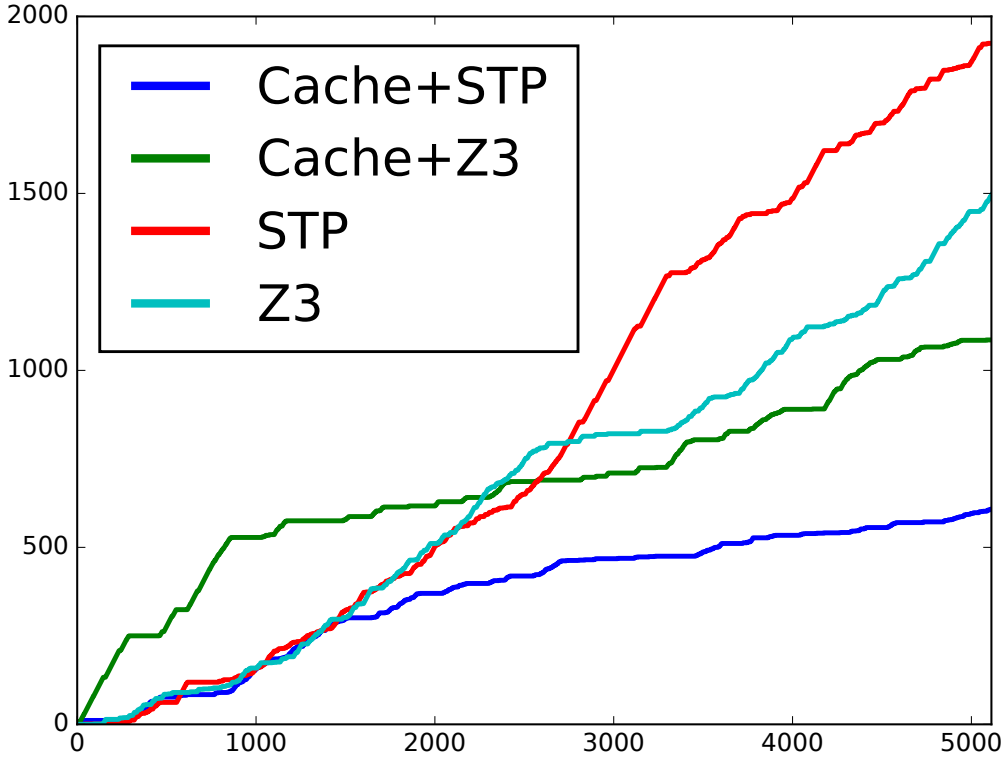
Searcher: BFS

Parallel Portfolio S. & Global Caching
Evaluation

Cumulative Number of Queries per Solver for `tsort`



1. without Global Cache



2. with Global Cache

Summary

- ✓ Global Caching ⇒ prototype
  - ✓ across multiple runs (same configuration)
    ⇒ -66% Solver Time
  - ✓ across multiple runs (using different strategies)
    ⇒ -14% Solver Time
- ✓ Parallel Portfolio Solving ⇒ -10% to -16% Solver Time

References I

[1] coreutils, 2017. URL https://www.gnu.org/software/coreutils/. Last accessed: 20 March 2017.

[2] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4, pages 171–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-22110-1. doi: 10.1007/978-3-642-22110-1_14. URL http://dx.doi.org/10.1007/978-3-642-22110-1_14.

[3] Robert Brummayer and Armin Biere. Boolector: An efficient smt solver for bit-vectors and arrays. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pages 174–177. Springer, 2009.

[4] Cristian Cadar, Daniel Dunbar, and Dawson Engler. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs. In Proceedings of the 8th USENIX conference on Operating systems design and implementation, 2008.

[5] Cristian Cadar, Vijay Ganesh, Peter M Pawlowski, David L Dill, and Dawson R Engler. Exe: automatically generating inputs of death. ACM Transactions on Information and System Security (TISSEC), 12(2):10, 2008.

[6] Sylvain Conchon, David Déharbe, Matthias Heizmann, , and Tjark Weber. STM-COMP 2016 - QF_BV (Main Track), 2016. URL http://smtcomp.sourceforge.net/2016/results-QF_AUFBV.shtml. Last accessed: 20 March 2017.

[7] Bruno Dutertre and Leonardo De Moura. The yices smt solver. Tool paper at http://yices.csl.sri.com/tool-paper.pdf, 2(2), 2006.

[8] Vijay Ganesh and DavidL. Dill. A decision procedure for bit-vectors and arrays. In Werner Damm and Holger Hermanns, editors, Computer Aided Verification, volume 4590 of Lecture Notes in Computer Science, pages 519–531. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73367-6. doi: 10.1007/978-3-540-73368-3_52. URL http://dx.doi.org/10.1007/978-3-540-73368-3_52.

[9] Jörg Hoffmann and Jana Koehler. A new method to index and query sets. In IJCAI, volume 99, pages 462–467, 1999.

[10] Xiangyang Jia, Carlo Ghezzi, and Shi Ying. Enhancing reuse of constraint solutions to improve symbolic execution. CoRR, abs/1501.07174, 2015. URL http://arxiv.org/abs/1501.07174.

References II

[11] Heinrich Kießling. Towards optimizing cache-utilization of symbolic execution. A thorough analysis of KLEEs caching implementation, March 2016.

[12] Jim Meyering, Pádraig Brady, Bernhard Voelker, Eric Blake, Paul Eggert, and Assaf Gordon. include-graph of yes.c from gnu core utilities v. 8.26. URL `https://fossies.org/dox/coreutils-8.27/yes_8c.html`. Last accessed: 20 March 2017.

[13] Leonardo Moura and Nikolaj Bjørner. Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, chapter Z3: An Efficient SMT Solver, pages 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78800-3. doi: 10.1007/978-3-540-78800-3_24. URL `http://dx.doi.org/10.1007/978-3-540-78800-3_24`.

[14] Martin Nowack. (under submission) new ipc infrastructure for klee's solver chain. 2018.

[15] Martin Nowack, Katja Tietze, and Christof Fetzer. Parallel symbolic execution: Merging in-flight requests. In Nir Piterman, editor, Hardware and Software: Verification and Testing, volume 9434 of Lecture Notes in Computer Science, pages 120–135. Springer International Publishing, 2015. ISBN 978-3-319-26286-4. doi: 10.1007/978-3-319-26287-1_8. URL `http://dx.doi.org/10.1007/978-3-319-26287-1_8`.

[16] Hristina Palikareva and Cristian Cadar. Multi-solver support in symbolic execution. In Natasha Sharygina and Helmut Veith, editors, Computer Aided Verification, volume 8044 of Lecture Notes in Computer Science, pages 53–68. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39798-1. doi: 10.1007/978-3-642-39799-8_3. URL `http://dx.doi.org/10.1007/978-3-642-39799-8_3`.

[17] Heinz Riener, Mathias Soeken, Clemens Werther, Görschwin Fey, and Rolf Drechsler. metasmt: A unified interface to smt-lib2. In Specification and Design Languages (FDL), 2014 Forum on, volume 978, pages 1–6. IEEE, 2014.

[18] Willem Visser, Jaco Geldenhuys, and Matthew B. Dwyer. Green: Reducing, reusing and recycling constraints in program analysis. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE '12, pages 58:1–58:11, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1614-9. doi: 10.1145/2393596.2393665. URL `http://doi.acm.org/10.1145/2393596.2393665`.