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**
  \[ \rightarrow 92\% \text{ of runtime for SAT solving} \]

\[ \Rightarrow \text{Caching as space-time trade-off} \]
\[ \downarrow 41\% \text{ of runtime for caching and SAT solving} \]

---

Query Caching (in KLEE)
Solver Chain

KLEE
Executor → Independence Solver → Simple Cache → Counter Example Cache

ClientProcess → SAT-Solver
Query Caching (in KLEE) Counter Example Cache

Reuse of concrete assignments via sub-/superset matching

Executor \(\rightarrow\) \(\rightarrow\) \(\rightarrow\) Counter Example Cache \(\rightarrow\) SAT-Solver

Query 1: \(G\) \(G > 0, G < 5\)

<table>
<thead>
<tr>
<th>UBT(e^2)</th>
<th>Assignm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_x\rightarrow C_y\rightarrow C_z)</td>
<td>(A_x A_y A_z)</td>
</tr>
<tr>
<td>(C_u)</td>
<td>(A_x A_y u)</td>
</tr>
<tr>
<td>(G &gt; 0 \rightarrow G &lt; 5)</td>
<td>(g = 1)</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(C_y\rightarrow C_z\rightarrow \ldots)</td>
<td>(A_y A_z \ldots)</td>
</tr>
<tr>
<td>(C_v\rightarrow \ldots)</td>
<td>(A_y v \ldots)</td>
</tr>
</tbody>
</table>
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 IPC³
  > Decouple cache and solver
  ⇒ Global reuse of concrete assignments (other literature⁴,⁵)

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

Mean: -66.54 %
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

\[ \text{ratio} = \frac{\text{CacheTime}}{\text{SolverTime}} \]
Parallel Portfolio Solving Idea

Annual ranking in SMT-COMP$^6$:
- STP$^7$
- Z3$^8$
- Boolector$^9$
- Yices$^{10}$
- CVC$^{11}$

---

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$^9$

---

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\textsuperscript{12,13}

12) \textit{metaSMT: A unified interface to SMT-LIB2}, Riener et al. [17]
13) \textit{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\(^{12,13}\)

\[ \Rightarrow \text{Parallel Portfolio Solving} \Rightarrow \text{More resource utilization} \]

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

Mean: -10.15%

Searcher: BFS
Parallel Portfolio S. & Global Caching Evaluation

Cumulative Number of Queries per Solver for tsort

1. without Global Cache
   - Cache+STP
   - Cache+Z3
   - STP
   - Z3

2. with Global Cache
   - Cache+STP
   - Cache+Z3
   - STP
   - Z3

KLEE's Solver Chain Revisited - Opportunities for Improvement? 
TU Dresden, Germany // Heinrich Kießling and Martin Nowack 
London, 19. April 2018

Slide 14 of 17
✓ 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


References II


