Demonstrating the Portfolio Analysis in CPAchecker

Thomas Bunk

LMU Munich, Germany

October 01, 2021
Outline

- Motivation
- The CPAchecker Portfolio Analysis
- Evaluation
- Outlook
Emergence of large cloud service providers in recent years that provide large amounts of computing capacities.
Motivation

- Emergence of large cloud service providers in recent years that provide large amounts of computing capacities
- These make it feasible to run evaluations based on wall-time measurement
Motivation

- Emergence of large cloud service providers in recent years that provide large amounts of computing capacities

- These make it feasible to run evaluations based on wall-time measurement

- This talk presents a *portfolio solver* implemented in CPAchecker that exploits the availability of such large computation environments.
Outline

▶ Motivation

▶ The CPAchecker Portfolio Analysis

▶ Evaluation

▶ Outlook
Run CPAchecker using the portfolio-configuration:

```
scripts/cpa.sh -portfolio <path/to/program> \  
   -spec <path/to/specification>
```
The Portfolio Analysis: An Overview (2/3)

- The portfolio algorithm runs the sub-analyses in parallel, possibly on large distributed machines.

Main instance

Icons taken from Wikimedia Commons
The portfolio algorithm runs the sub-analyses in parallel, possibly on large distributed machines.

Icons taken from Wikimedia Commons
Based on the results of the sub-analyses, the main instance concludes with a final result.
The analysis starts as regular {\texttt{CPAchecker}}-instance
Requirements

- The analysis starts as regular \texttt{CPAchecker}-instance
- That runs the sub-analyses in parallel
Requirements

- The analysis starts as regular CPAchecker-instance
- That runs the sub-analyses in parallel
- ... not only on the same machine, but also on large distributed systems!
Requirements

- The analysis starts as regular \texttt{CPAchecker}-instance
- That runs the sub-analyses in parallel
- ... not only on the same machine, but also on large distributed systems!

Approach:

The \textit{Message Passing Interface} (MPI) standard
The MPI Standard

- Standardized interface for passing messages between processes
- Supports both point-to-point and collective communication (e.g., multicasts or broadcasts)
- Supports blocking and non-blocking communication

Figure: Example for broadcast communication (Image from Wikimedia commons)
MPI defines a standard for launching MPI programs: 
*mpiexec*

Invokes processes on a specified list of hosts

Highly configurable – allows mapping of processes
- e.g., to nodes, sockets, cores, L1cache, L2cache, numa, board, and more

Example command line:
```
> mpiexec -hostfile <hostfile> -np 4 -map-by core <executable>
```
MPI in the Portfolio Analysis

Main instance

Subanalysis 1

Subanalysis 2

Subanalysis 3

Subanalysis n

Icons taken from iconarchive.com

Thomas Bunk 
LMU Munich, Germany
MPI in the Portfolio Analysis

Main instance

Subanalysis 1

Subanalysis 2

Subanalysis 3

Subanalysis n

Main instance

Icons taken from iconarchive.com

Thomas Bunk

LMU Munich, Germany
MPI in the Portfolio Analysis

Subanalysis 1

Subanalysis 2

Subanalysis 3

Subanalysis n

Main instance

Icons taken from iconarchive.com

Thomas Bunk  
LMU Munich, Germany
Choosen Sub-analyses in Portfolio Analysis

- Current selected analyses (specified in config/portfolio.properties):
  1. ValueAnalysis with CEGAR
  2. ValueAnalysis
  3. Predicate-Analysis
  4. $k$-Induction
  5. BMC
  6. BAM
  7. PDR

- Analyses perform an additional CEX-check if concluding with a FALSE result
Live Demo
Outline

- Motivation
- The CPACHECKER Portfolio Analysis
- Evaluation
- Outlook
Evaluation (1/3)

See results in wall-time demo track from SV-COMP’20:

▶ https://sv-comp.sosy-lab.org/2020/results-demo/results-verified
### Evaluation (2/3)

<table>
<thead>
<tr>
<th></th>
<th>CPA-Seq</th>
<th></th>
<th>CPA-Par</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>status</td>
<td>raw score</td>
<td>walltime (s)</td>
<td>status</td>
</tr>
<tr>
<td>total</td>
<td>1 567</td>
<td>1 492</td>
<td>440 000</td>
<td>1 579</td>
</tr>
<tr>
<td>correct results</td>
<td>935</td>
<td>1 588</td>
<td>25 000</td>
<td>937</td>
</tr>
<tr>
<td>correct true</td>
<td>653</td>
<td>1 306</td>
<td>18 000</td>
<td>667</td>
</tr>
<tr>
<td>correct false</td>
<td>282</td>
<td>282</td>
<td>7 100</td>
<td>270</td>
</tr>
<tr>
<td>incorrect results</td>
<td>6</td>
<td>−96</td>
<td>400</td>
<td>3</td>
</tr>
<tr>
<td>incorrect true</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>0</td>
</tr>
<tr>
<td>incorrect false</td>
<td>6</td>
<td>−96</td>
<td>400</td>
<td>3</td>
</tr>
</tbody>
</table>

Score (SV-COMP): - 1 746

---

Categories: ReachSafety – Arrays, BitVectors, ControlFlow, Floats, Heap, Loops
Evaluation (3/3)

sv-comp.sosy-lab.org/2020/results-demo/results-verified/META_-ReachSafety.table.html#/quantile?hidden=2,3,4,5&selection=walltime
Outline

▶ Motivation

▶ The CPAchecker Portfolio Analysis

▶ Evaluation

▶ Outlook
Outlook

- Run analysis on the whole benchmark set
- Minimize ramp-up time
- Tweak sub-analyses more towards features of program (loops, floats, recursion, etc.)
Thank you for your attention!