Program-Structure-Guided Approximation of Large Fault Spaces

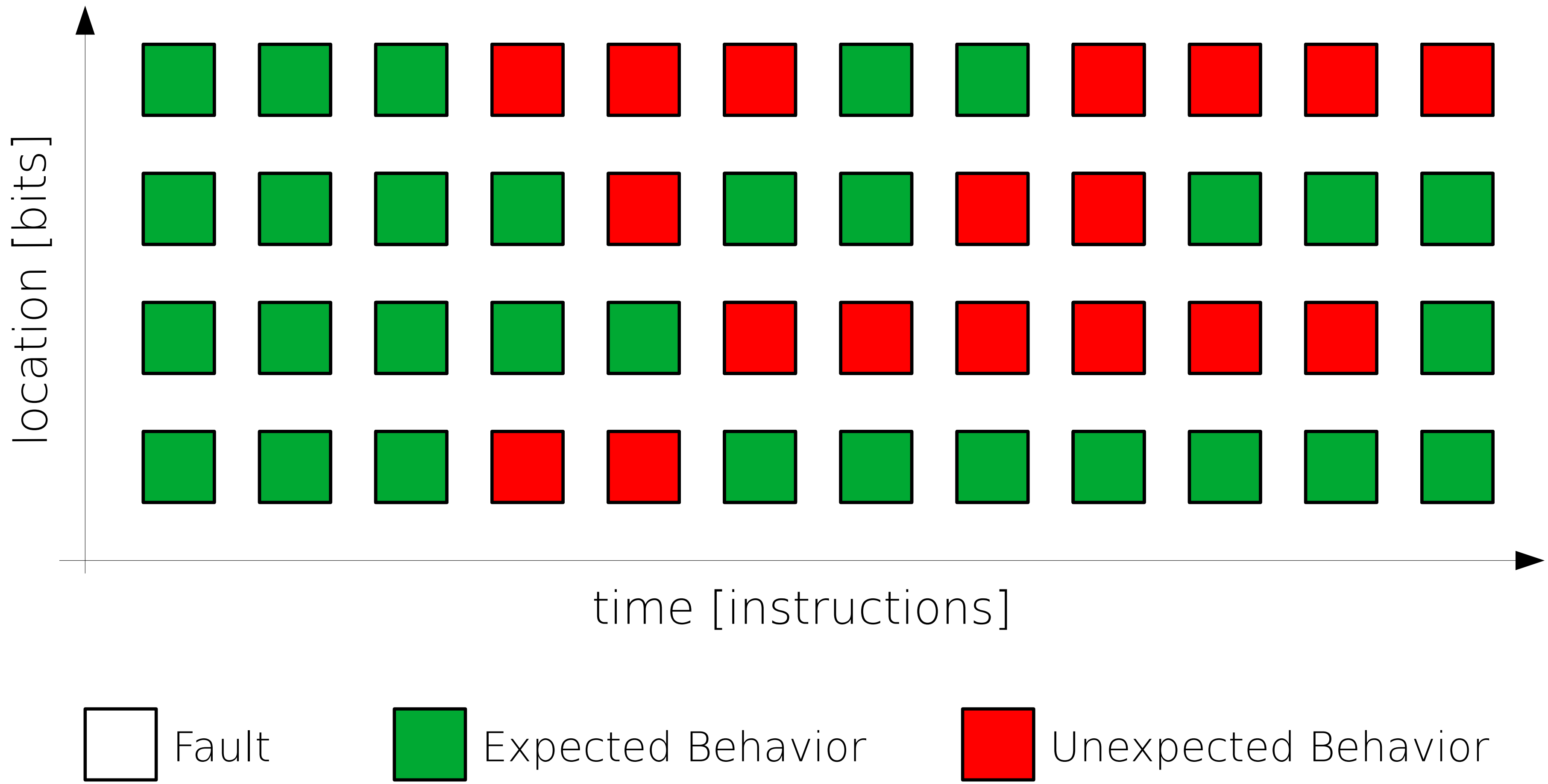
Oskar Pusz, Daniel Kiechle, Christian Dietrich, Daniel Lohmann
Systems Research and Architecture Group (SRA)
Leibniz University Hanover, Germany

03.12.2019

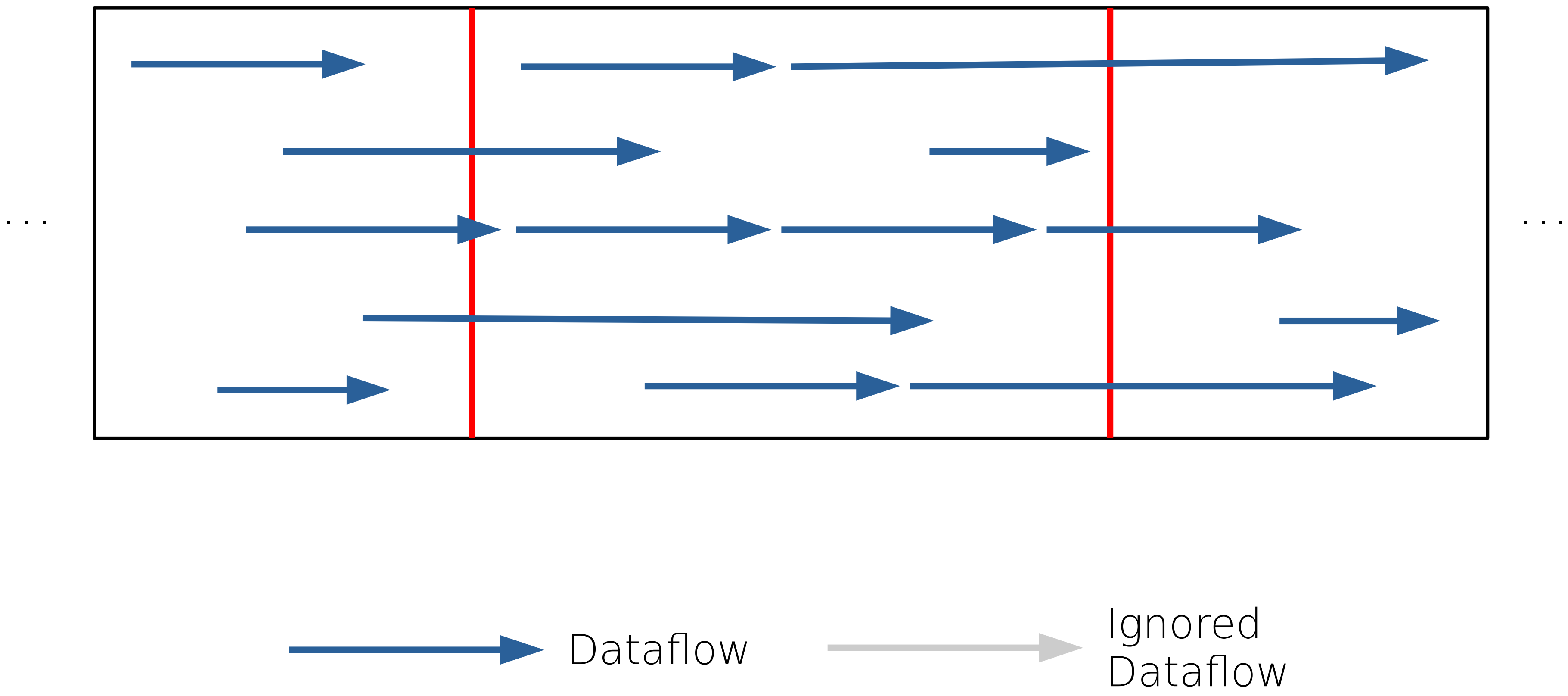
Motivation

- Devices with high responsibility need to be reliable
- Shrinking transistors make hardware more vulnerable
- Quantifying software safeness in presence of transient hardware errors: Simulation-based fault injection
- Problem: Fault injection campaign run time

Fault Space Equivalence Sets

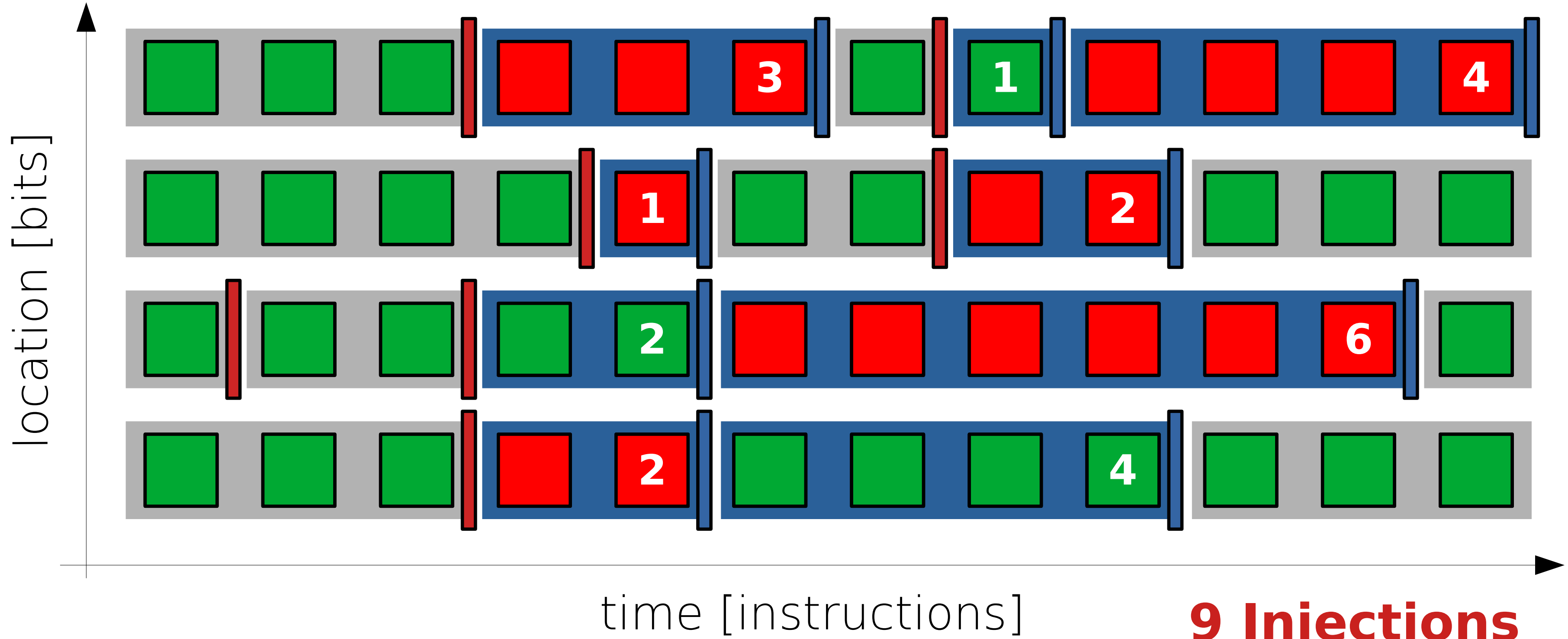


Approach in a nutshell

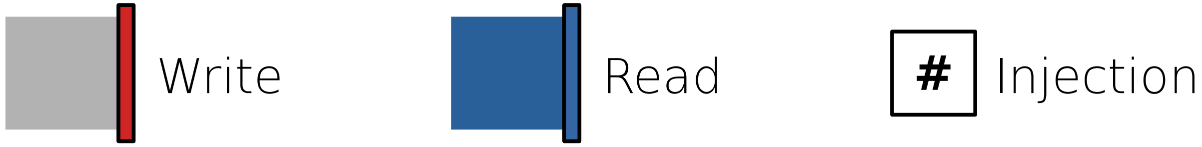


Fault Space Equivalence Sets

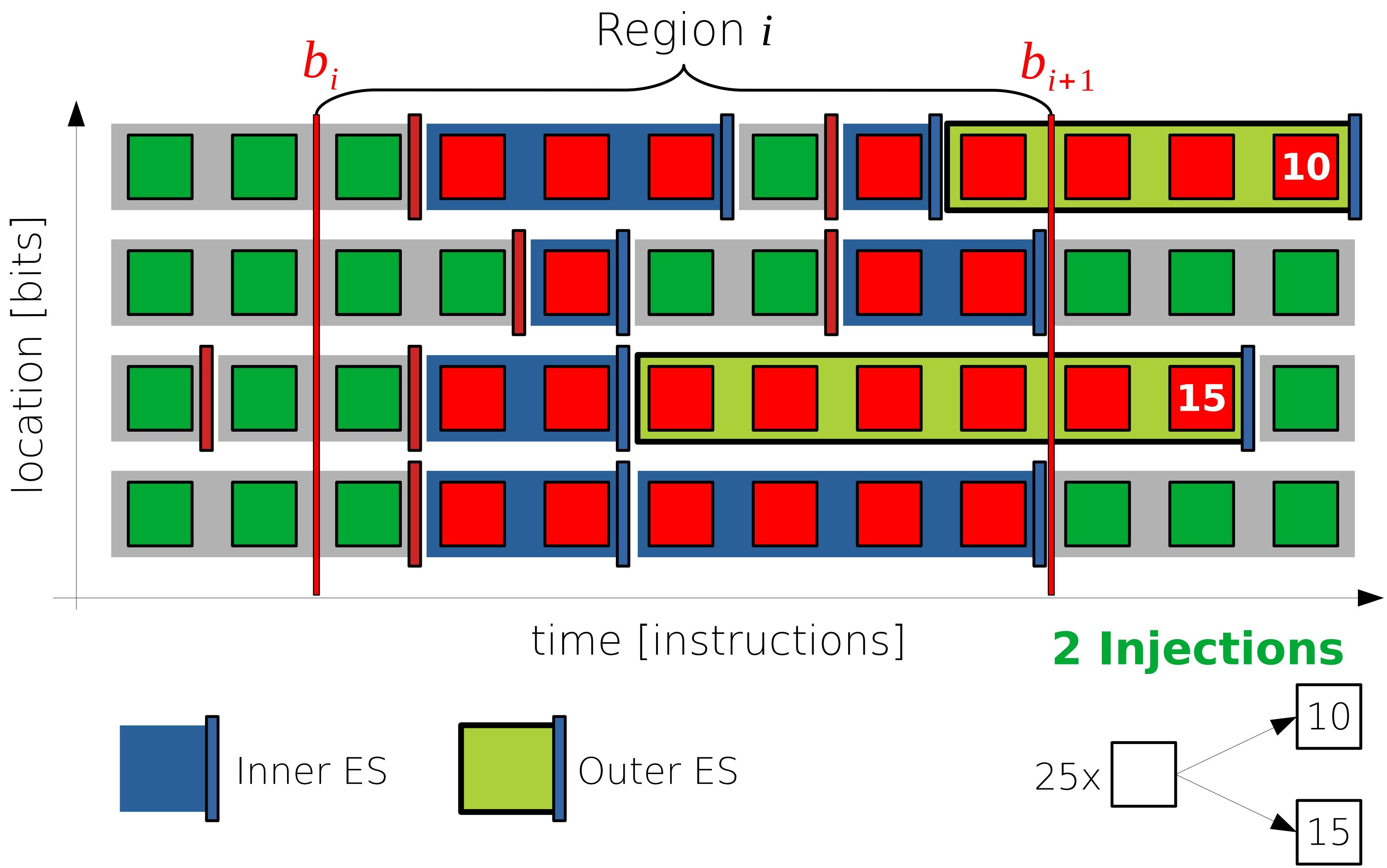
Def/Use-Pruning



9 Injections

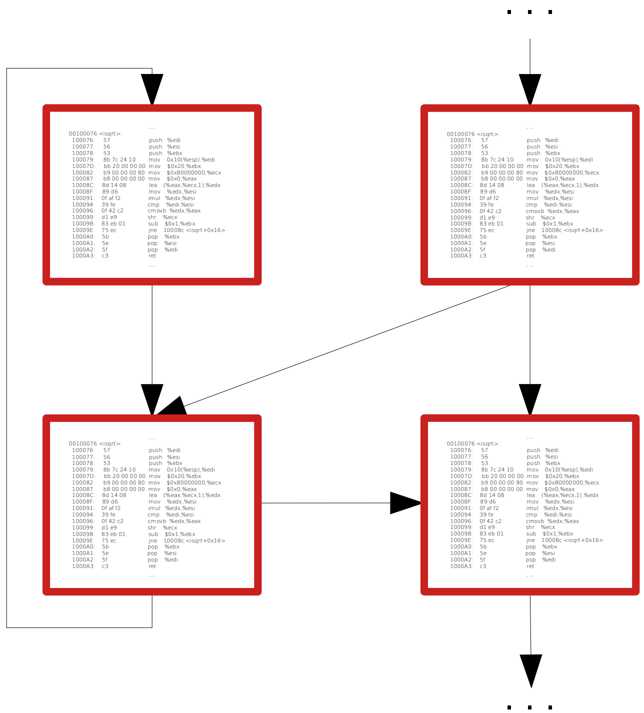


Fault Space Regions

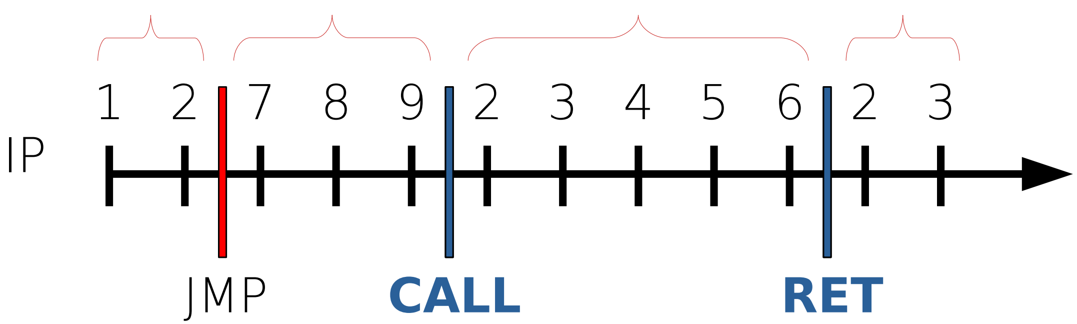


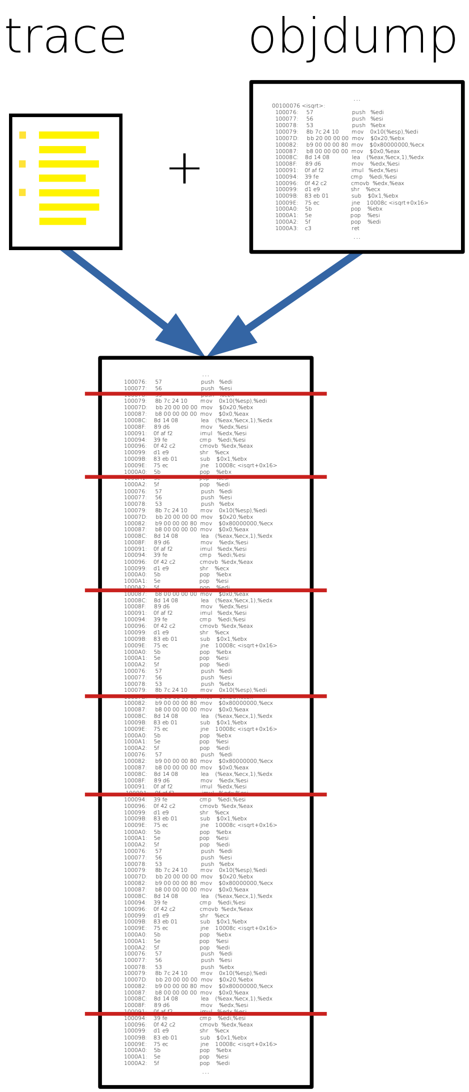
Realization of Program Structure

Static Basic Blocks



Dynamic Basic Blocks





Evaluation

- Fault injection framework **FAIL*** is extended and emulates on the IA-32 simulator Bochs

- Used hardware:
  - Def/Use + FSR calculation: Intel i5-7400 @ 3Ghz (4 cores)
  - FI Campaign: 17 Intel Xeon @ 2.67 Ghz (12 cores each)

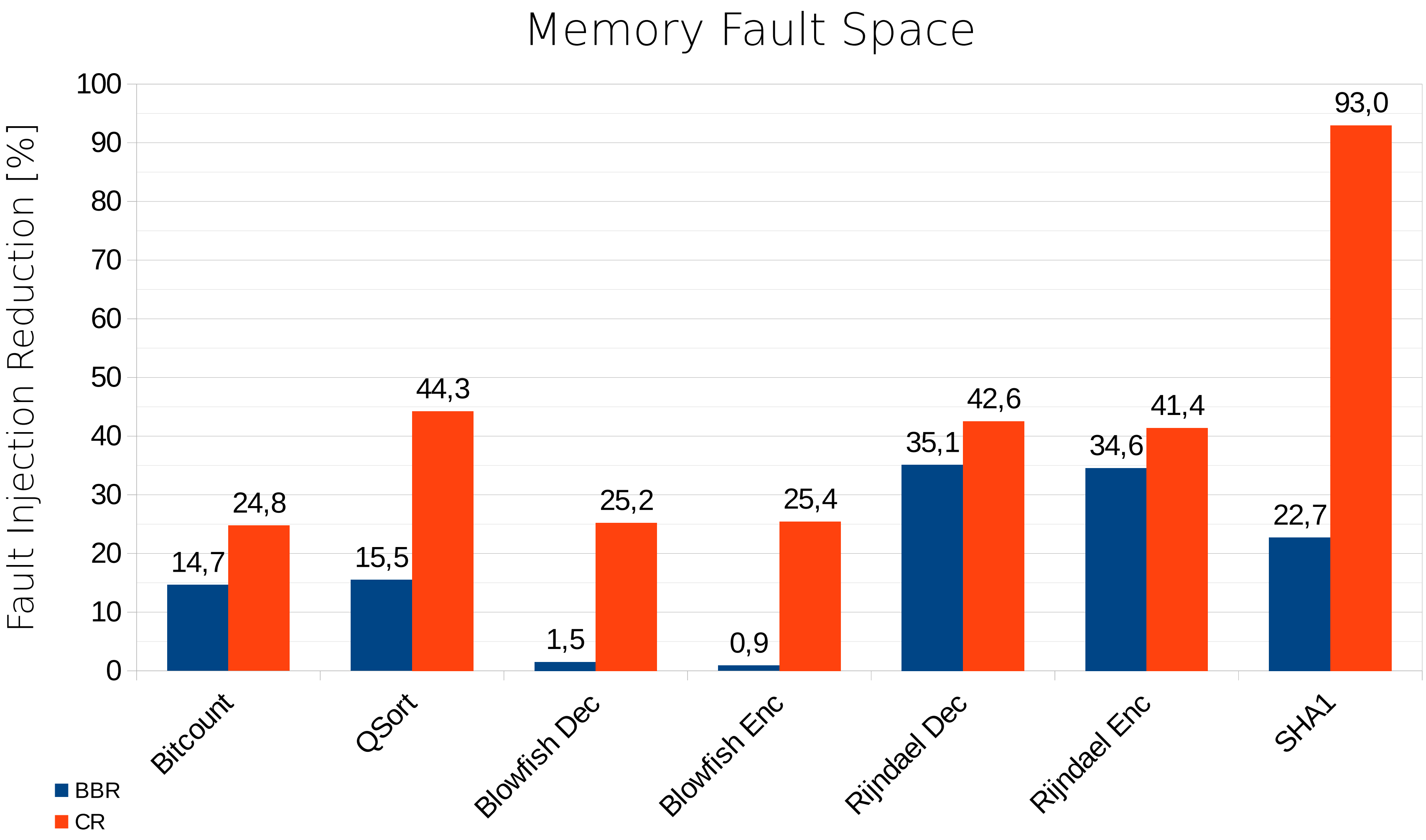
- Dimensions of the evaluation
  - Seven selected Programs of the automotive and security branch of the MiBench benchmark suite
  - Two instantiations of the FSRs: basic block and call regions
  - Three fault models: memory, GP registers, combined

Evaluation - Reduction

- Number of injections after the precise and complete Def/Use pruning is the baseline
- FSR calculation took end-to-end no more than 4.1 seconds

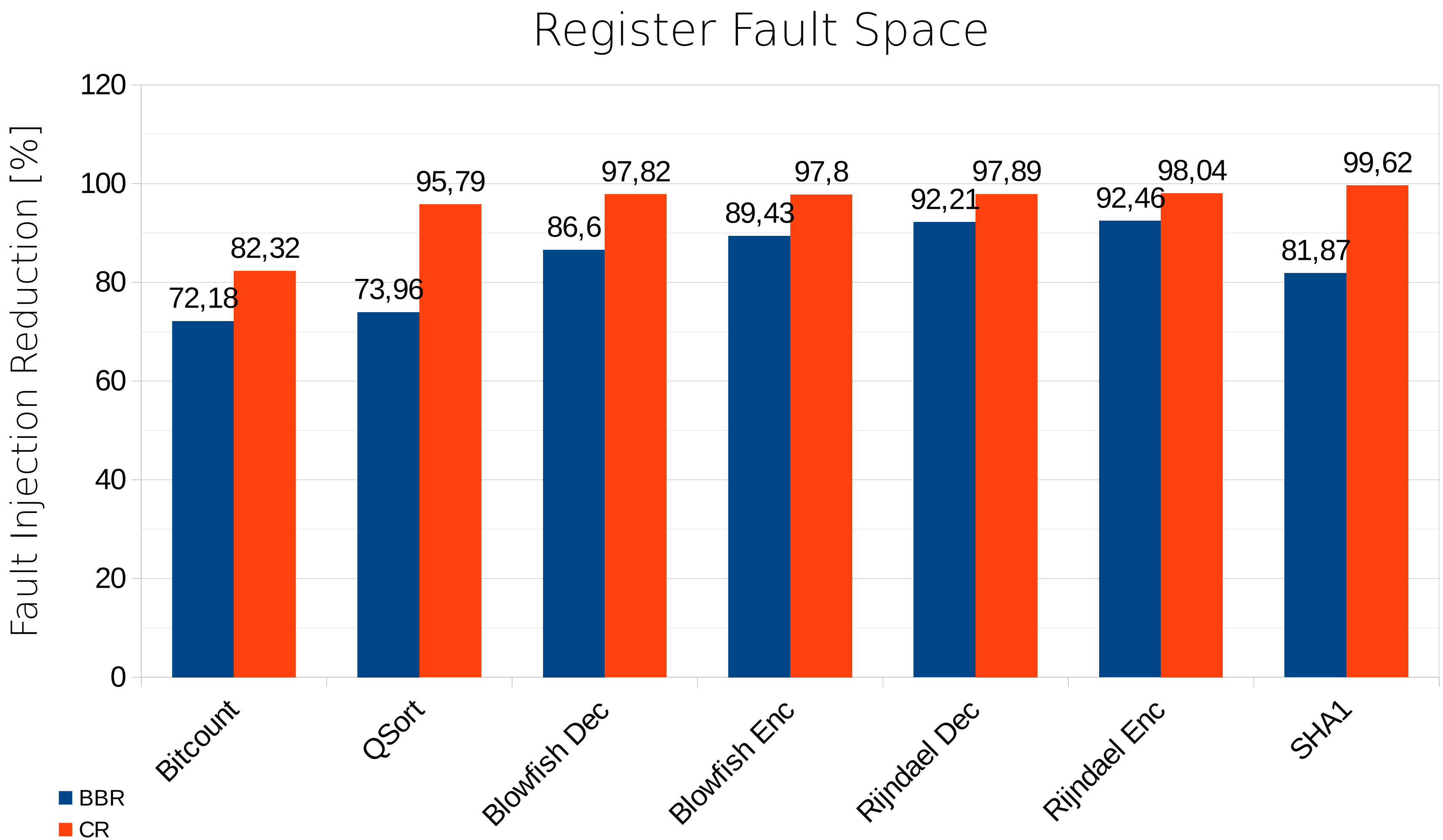
fault model		Memory	Register	Combined
#injections	BBR[%]	−9.79	−83.73	−69.91
	CR [%]	−38.03	−95.44	−83.87

Evaluation - Reduction



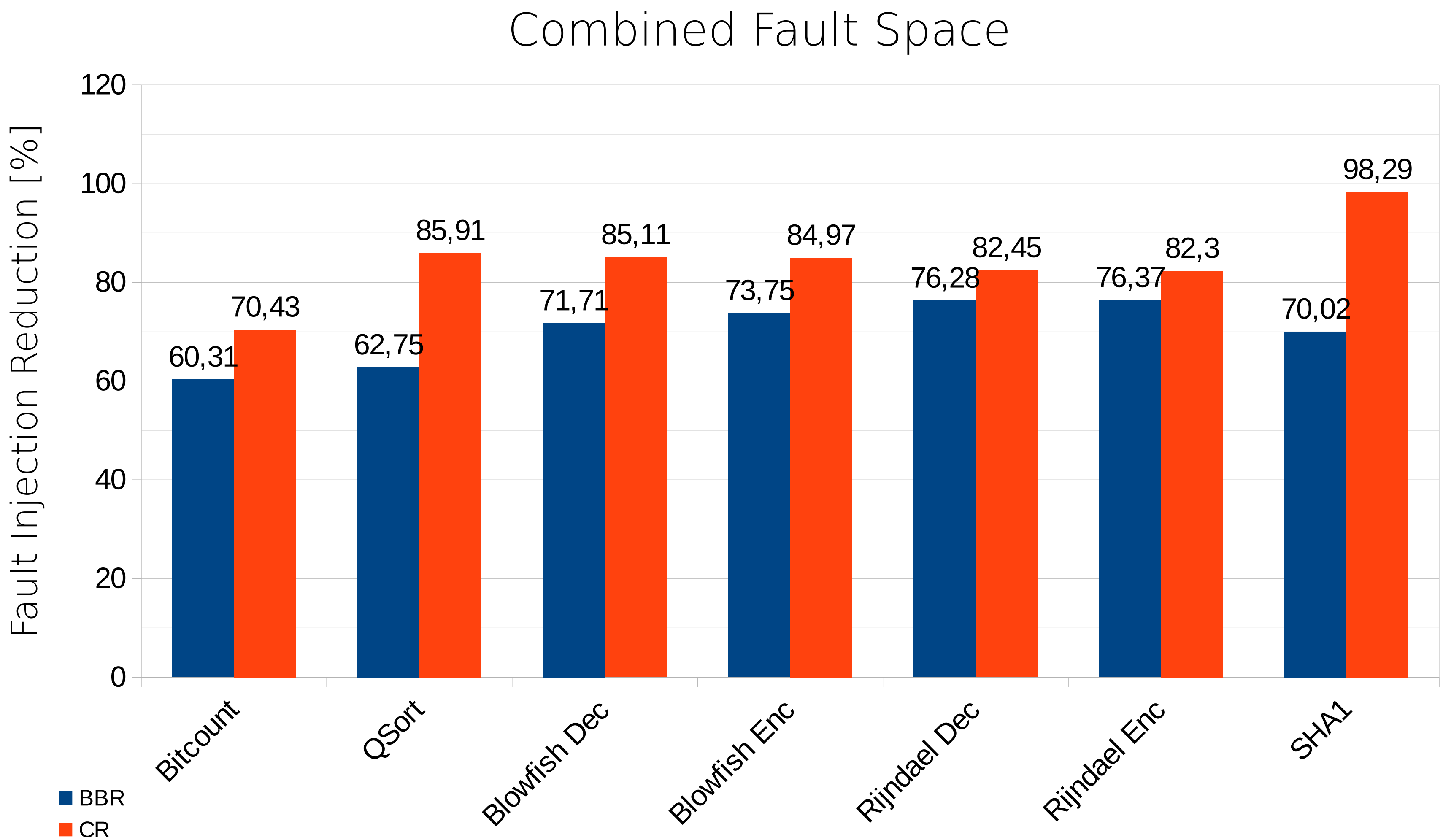
Deviation for BBR below 0.2%

Evaluation - Reduction



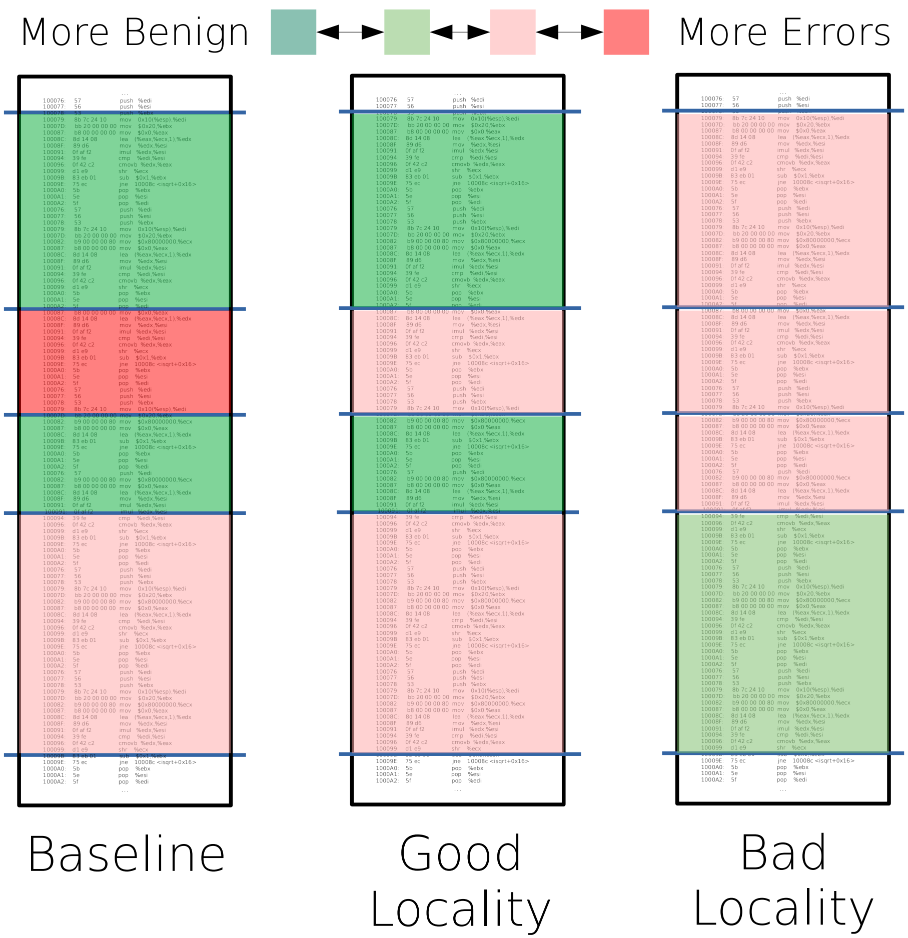
High Reduction → High Deviation
BBR up to 41.3 %, CR up to 108.8 %

Evaluation - Reduction



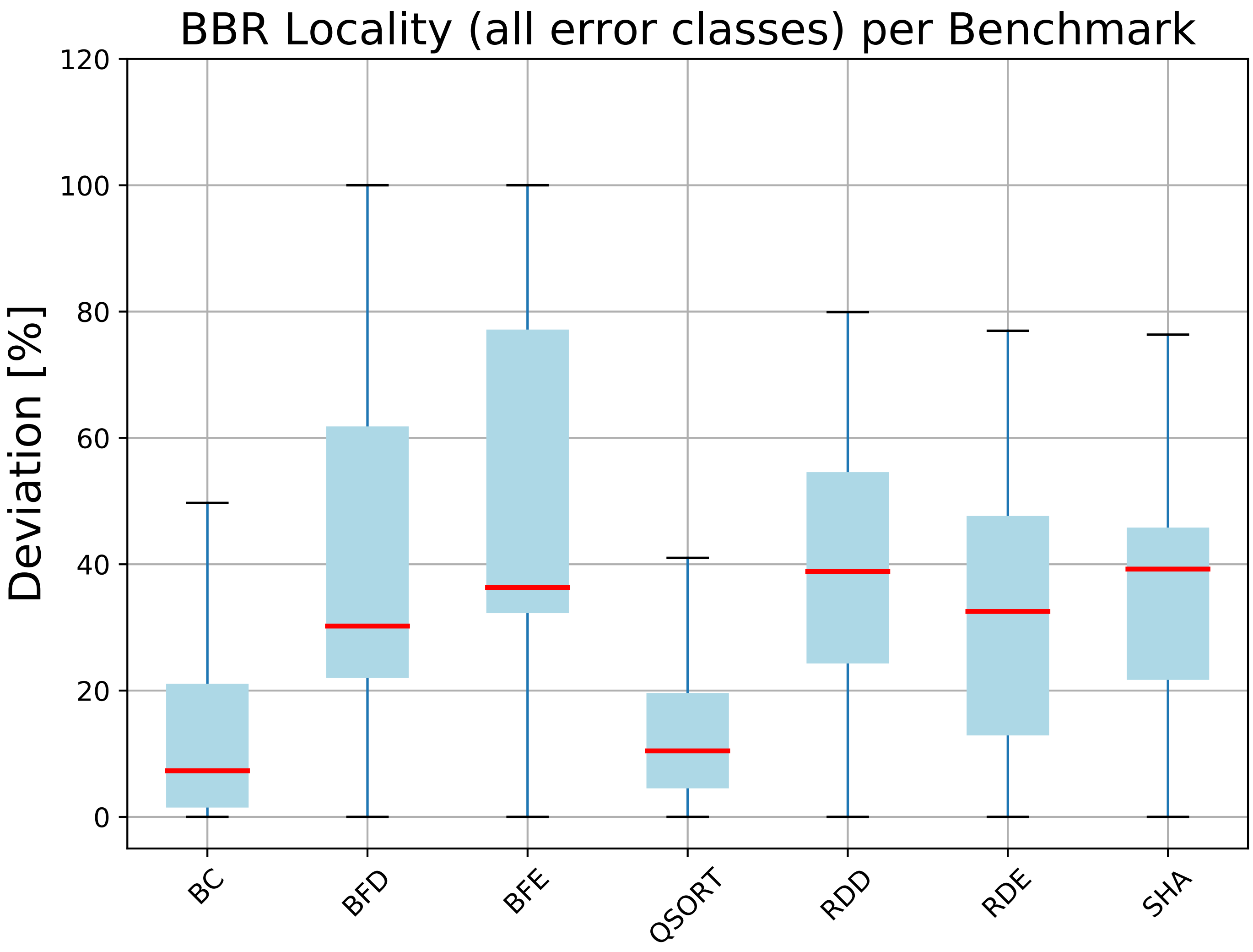
Deviation BBR 0.2% - 2.7%

Evaluation - Locality

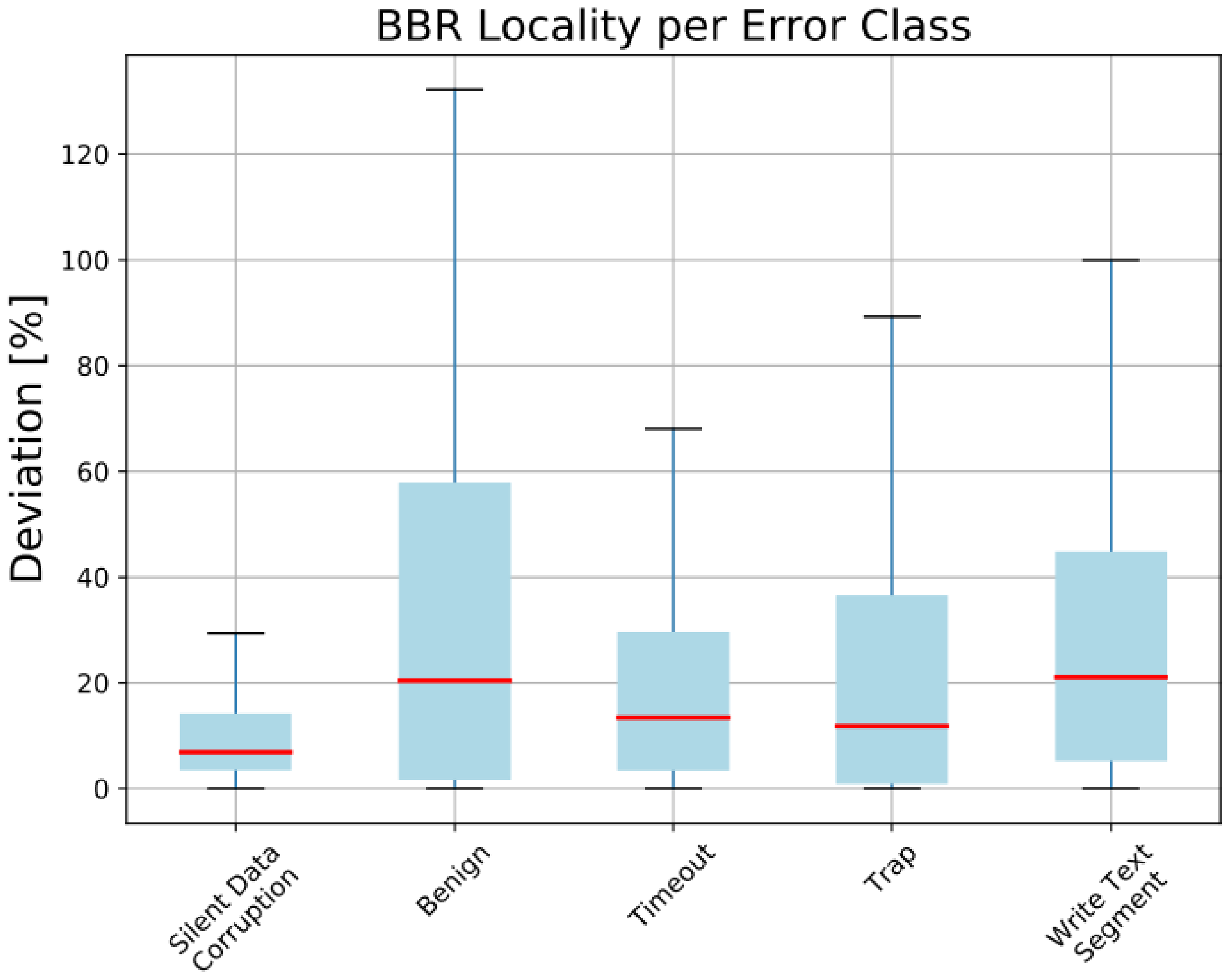


- Dimensions of the evaluation
  - The seven programs from MiBench
  - For every error class: Silent Data Corruption, Benign, Timeout, Trap, Write Text Segment
- Comparison of two error-rate vectors
  - Precise vector from Def/Use
  - Approximated one from BBRs
  - Geometric mean over the class deviations

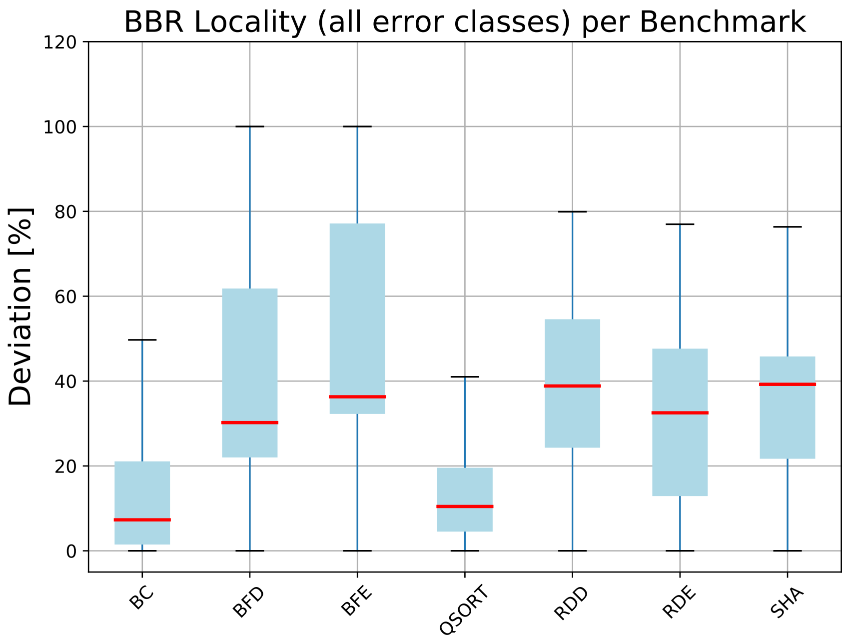
Evaluation - Locality

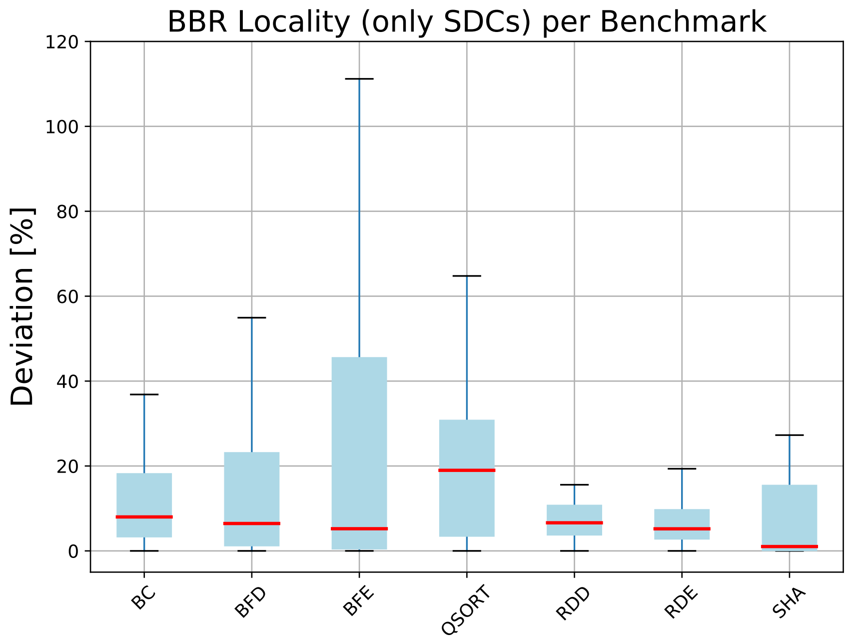


Evaluation - Locality



Evaluation - Locality





BBRs keep locality of the results for SDCs in combined FM

Memory fault space: Median 0%, 75-quantile mostly 0%

Register fault space: As expected, bad locality of the results

Conclusion

- What we did
  - We extract the program structure from a program trace
  - The extracted structure leads to fault space regions
  - Dataflows which cross regions borders will be injected
- We could reduce the number of required faults
  - Combined fault space -76% with 2.7% deviation
  - Even more precise for the memory fault space
  - Locality of the results kept regarding silent data corruption
- Check it out: **FAIL*** - Fault Injection Leveraged (https://github.com/danceos/fail)