Boosting Resilience Efficiency in the Face of Frequent, Clustered, and Diverse Faults

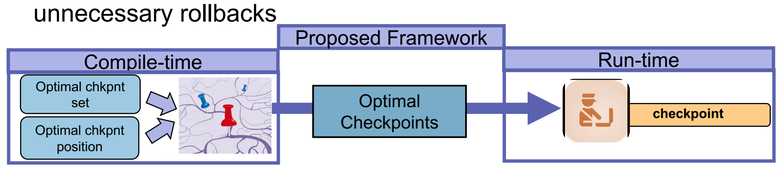
Chengmo Yang
Department of Electrical and Computer Engineering, University of Delaware

Gem5-based Integrated Fault tolerant Framework

Adaptive Fault Detection and Checkpoint

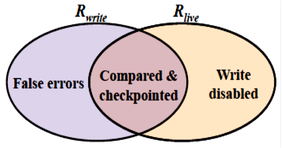
Why?

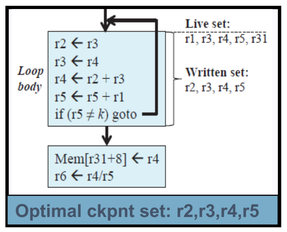
✓Minimize checkpointing and detection overhead and prevent unnecessary rollbacks



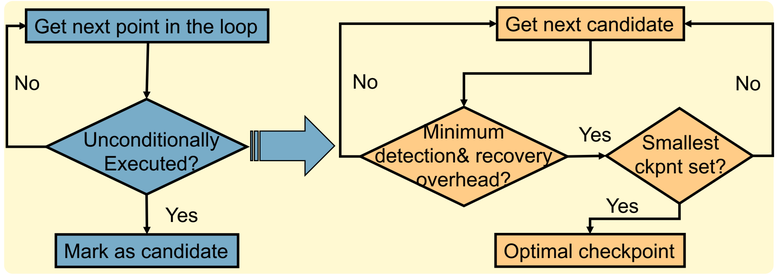
❑**Optimal checkpointed set**

$$R_{ckpt(i)} = R_{ckpt(i)} \cap R_{write(loop\ body)}$$





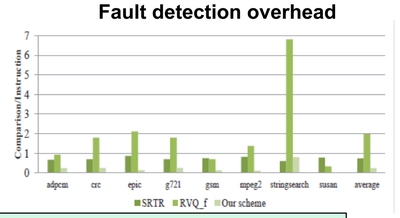
❑**Optimal Checkpoint position**



Ratio of various types of faults

	RVQ_f			Our scheme		
	Detect	Mask	Miss	Detect	Mask	Miss
adpcm	0.835	0.165	0	0.749	0.251	0
crc	0.800	0.200	0	0.799	0.201	0
epic	0.897	0.103	0	0.794	0.206	0
g721	0.855	0.145	0	0.588	0.412	0
gsm	0.676	0.324	0	0.645	0.355	0
mpeg2	0.796	0.204	0	0.769	0.231	0
stringsearch	0.786	0.214	0	0.600	0.400	0
susan	0.572	0.428	0	0.115	0.885	0
average	0.777	0.223	0	0.632	0.368	0

Fault detection overhead



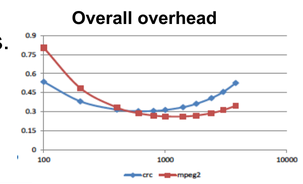
Conclusions:

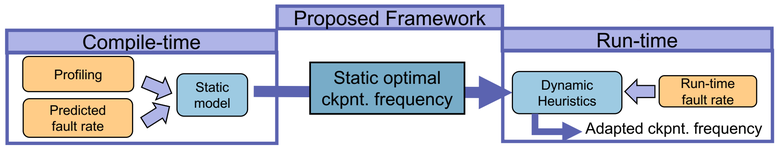
✓ Full fault coverage
✓ Reduce 88% of the comparison overhead
✓ Mask over 38% of the total injected faults.

Application-specific Checkpoint Frequency

Why?

✓Fault rate is unpredictable in most cases.
✓The higher the fault rate, the higher the checkpoint frequency
✓Optimal checkpoint frequency is application specific





❑**Static Model**

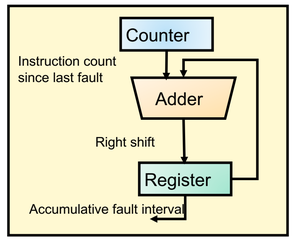
$$optimal\ checkpointing\ frequency = \alpha \cdot \sqrt{\frac{1}{\lambda}}$$

α: Application specific; λ: Fault rate

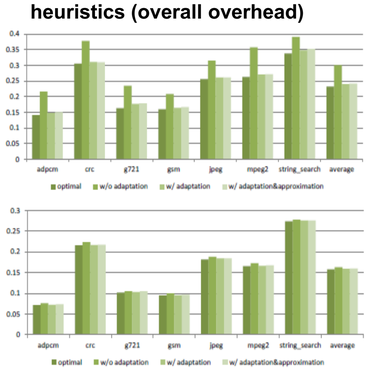
❑**Dynamic Heuristics**

✓Run-time fault rate = number of instructions executed between two consecutive faults.
✓Adaptation: Using the approximation of S

$$\frac{t_{new}}{t_{static}} = \sqrt{\frac{\Delta Fault_{acc}}{\Delta Fault_{static}}}$$



Evaluation of dynamic heuristics (overall overhead)



Optimal checkpoint intervals, analytical v.s experimental

	$\lambda = 10^{-4}$		$\lambda = 10^{-5}$		$\lambda = 10^{-6}$	
Benchmarks	Pred	Real	Pred	Real	Pred	Real
adpcm	833	800	2669	2500	8543	9000
crc	670	600	2104	2000	6614	6000
g721	951	1000	3098	3500	10058	10500
gsm	901	1200	2905	3000	9394	10500
jpeg	753	800	2405	3000	7658	9000
mpeg2	1144	1200	3654	4000	11615	12000
stringsearch	871	1000	2870	3000	9421	10500

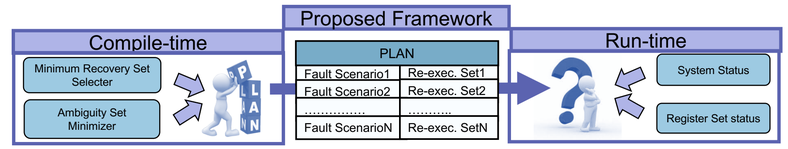
Conclusion:

✓ Static model: <6% deviation from actual optimal values
✓ Dynamic method is effective when fault rate is unknown *a priori*

Adaptive Fault Recovery

Why?

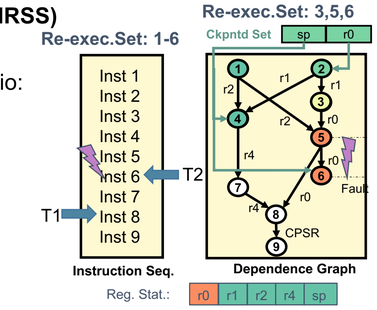
✓In face of elevated fault rate, systems need to recover faster
✓Traditional recovery: overhead independent of the fault scenario



❑**Minimum Recovery Set Selector (MRSS)**

✓ Classify Instructions in each scenario:

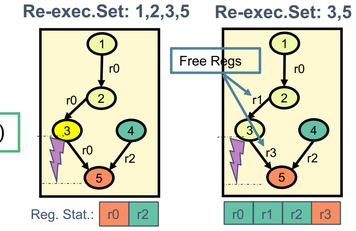
Faulty: Valid and Faulty
Clean: Valid and Clean
Ambiguous: overwritten



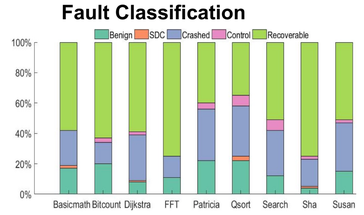
❑**Ambiguity Set Minimizer (ASM)**

✓ Convert original code to pseudo-SSA code using free registers.

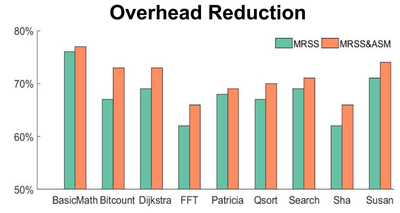
$$R_{FreeSet} = allregs - (R_{used(L)} \cup R_{Def(L)})$$



Fault Classification



Overhead Reduction



Conclusion:

✓Provide fault coverage
✓Require 16KB of storage
✓Applicable to 60% of faults
✓72% recovery overhead reduction

Publications

1) F. S. Hosseini, P. Fotouhi, C. Yang, and G. R. Gao. “Leveraging compiler optimizations to reduce runtime fault recovery overhead,” DAC, 2017.
2) H. A. Khouzani and C. Yang, "Towards a Scalable and Write-Free Multi-version Checkpointing Scheme in Solid State Drives," *DSN*, 2016.
3) C. Liu and C. Yang. “Secure and Durable (SEDURA): An Integrated Encryption and Wear-leveling Framework for PCM-based Main Memory,” LCTES, 2015.
4) C. Yang and M. Ruiz Varela, "Qualifying non-volatile register files for embedded systems through compiler-directed write minimization and balancing," *VLSI-SoC*, 2015.
5) L. A. Rozo Duque and C. Yang, “Improving MPSoC reliability through adapting runtime task schedule based on time-correlated fault behavior,” DATE, 2015.
6) L. A. Rozo Duque and C. Yang, “Guiding fault-driven adaption in multicore systems through a reliability-aware static task schedule," ASPDAC, 2015.
7) C. Liu and C. Yang, "Improving multi-level PCM reliability through age-aware reading and writing strategies," *ICCD, 2014.*
8) H. Chen and C. Yang. “Fault detection and recovery efficiency co-optimization through compile-time analysis and runtime adaptation,” CASES, 2013.
9) H. Chen and C. Yang. “Boosting efficiency of fault detection and recovery through application-specific comparison and checkpointing,” LCTES, 2013.

This work is supported by NSF grant #1253733