



MICRO-50, Boston, October 2017

Tutorial:

Microarchitecture Level Reliability Assessment

Throughput and Accuracy

<http://micro50-tutorial.di.uoa.gr/>

Organizers/Presenters:

Athanasios Chatzidimitriou, Manolis Kaliorakis, Dimitris Gizopoulos



MICRO-50, Boston, October 2017

Part 4:

MeRLiN Fault Pruning

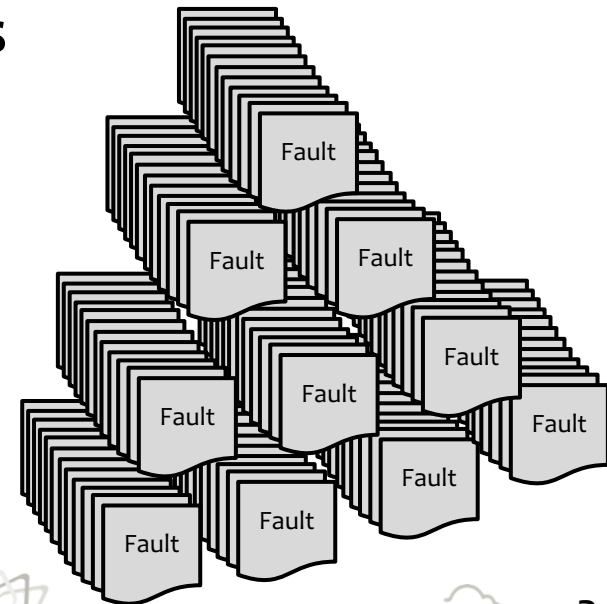
Motivation of Fault Pruning (1)

So far we investigated the speedup achieved by the **inherent characteristics** of the:

- 1) **Gem5 simulator:**
 - Early stop (invalid, overwrite)
 - checkpoint forwarding
- 2) **Machine** that runs the campaign:
 - parallel injections in multiple threads



Can we do better?
What about fault list
reduction?



Motivation of Fault Pruning (2)

Example:

- Small benchmark of **1B cycles**
- **1** hardware structure of (**32KB L1 data cache**)
- simulation throughput (microarchitecture) of **10^5 cycles/sec**
- **10 servers**

Exhaustive fault injection

(injection in **every cycle** and **bit** $\rightarrow \sim 2.6 \times 10^{14}$ injections)

$\sim 10^9$ years (!!!)

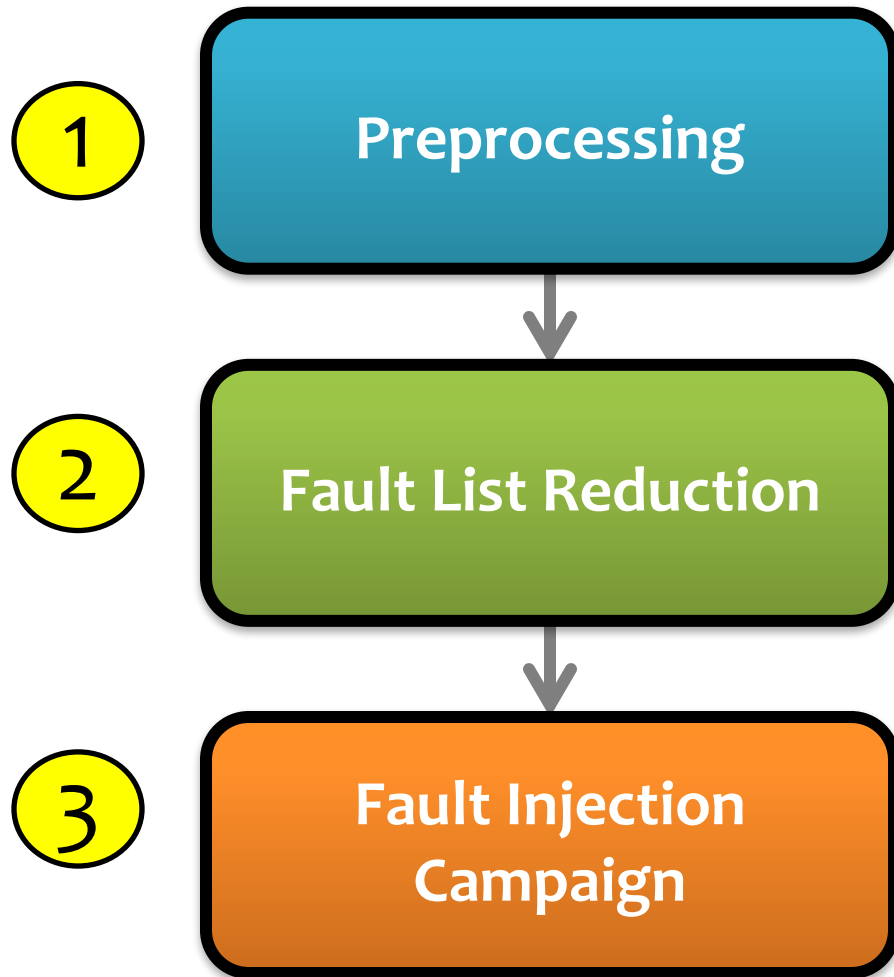
MeRLiN method:

- **2 to 3 order** of magnitude acceleration
- **Small effect in accuracy**

Confidence	Error Margin	#Injections	Fault Injection Campaign Time
95%	5%	384	~4 days
99%	3%	1843	~3 weeks
99%	1%	16,587	~6 months
99.8%	1%	23,873	~9 months
99.8%	0.5%	95,493	~3 years



MeRLiN's Flow



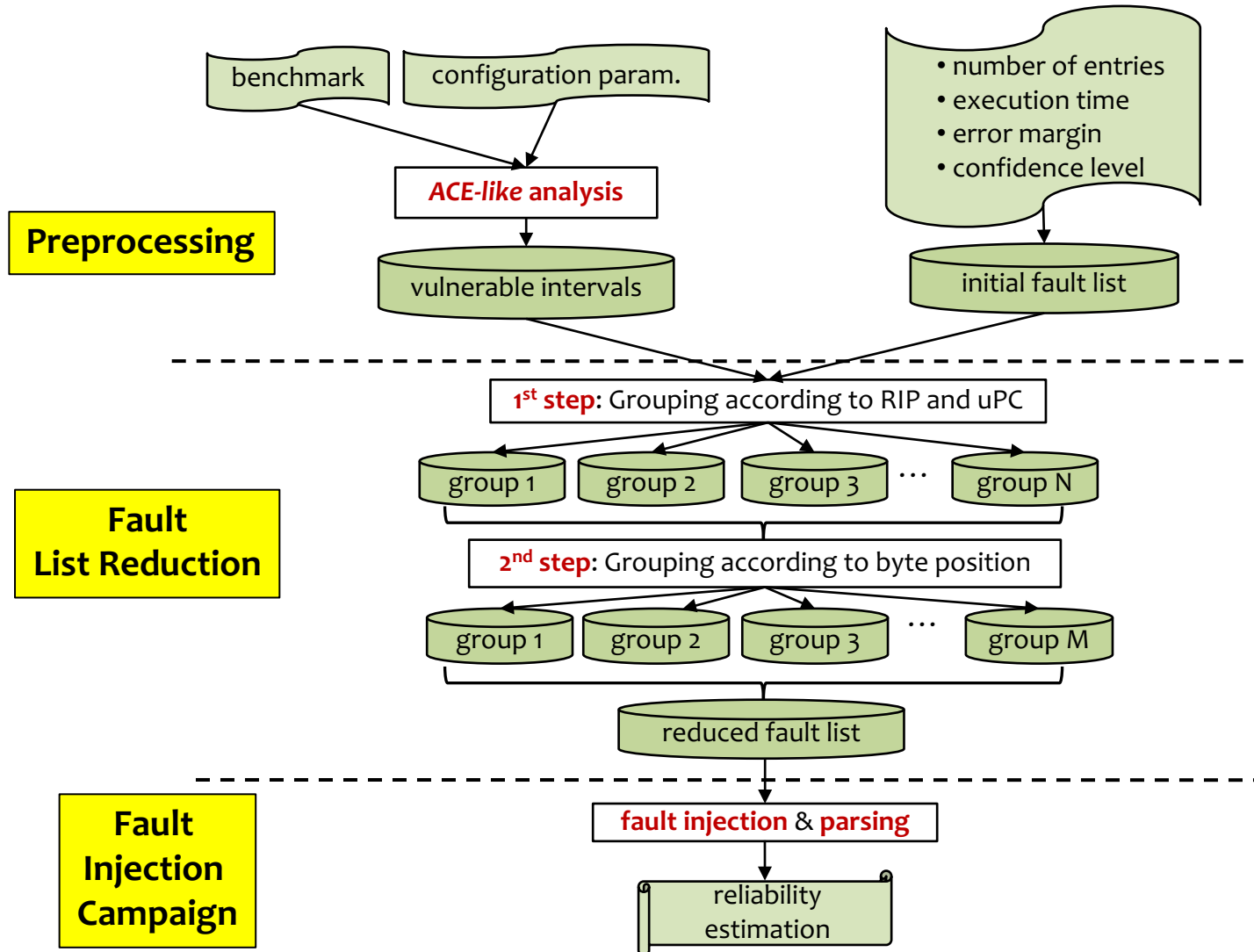
Two tasks:

- 1) Identification of faults that hit non-vulnerable intervals
- 2) Initial fault list generation

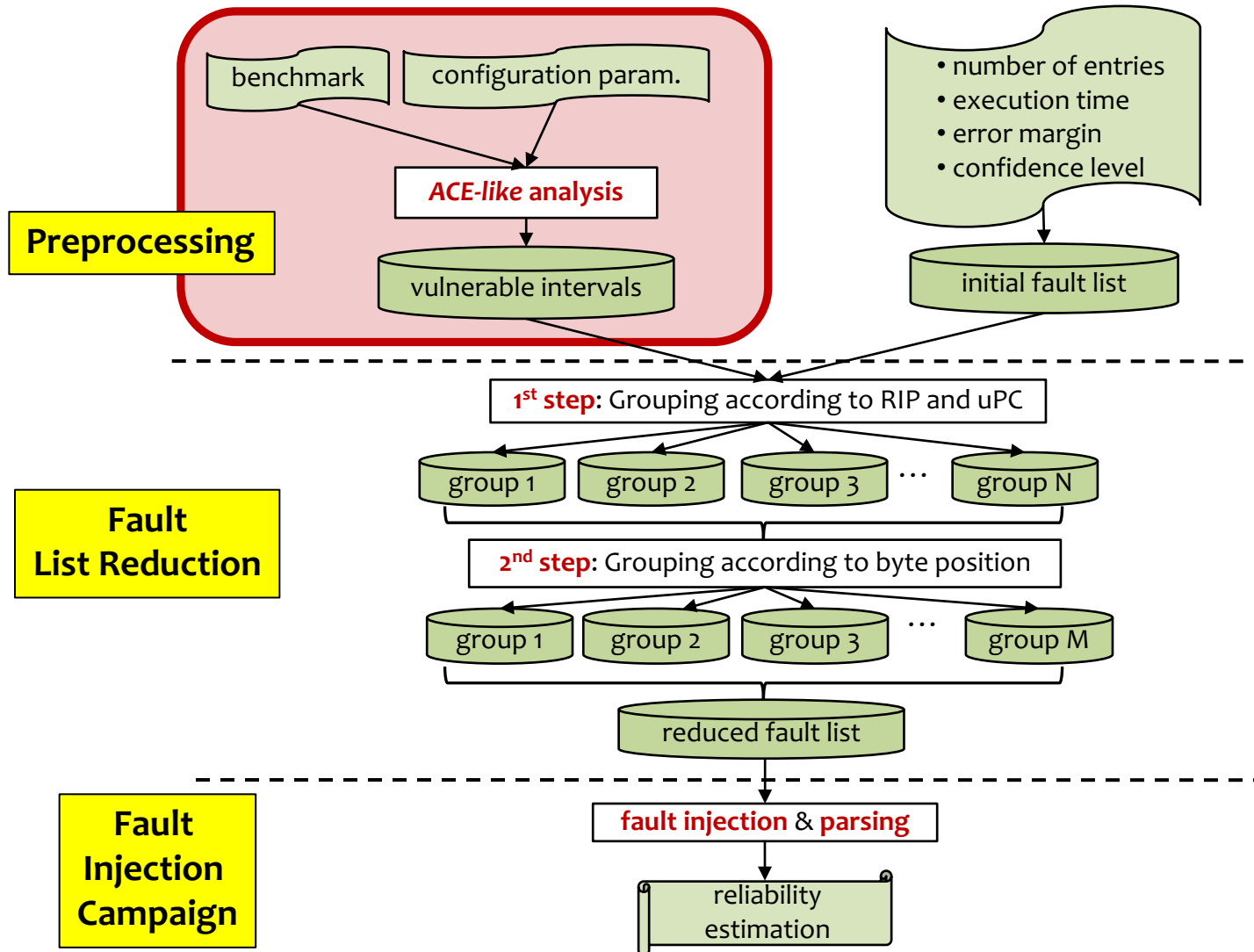
Implementation of a **two-step group creation algorithm** to reduce the remaining faults

Actual injections using the **GeFIN microarchitectural simulator**

MeRLiN's Flow in more details



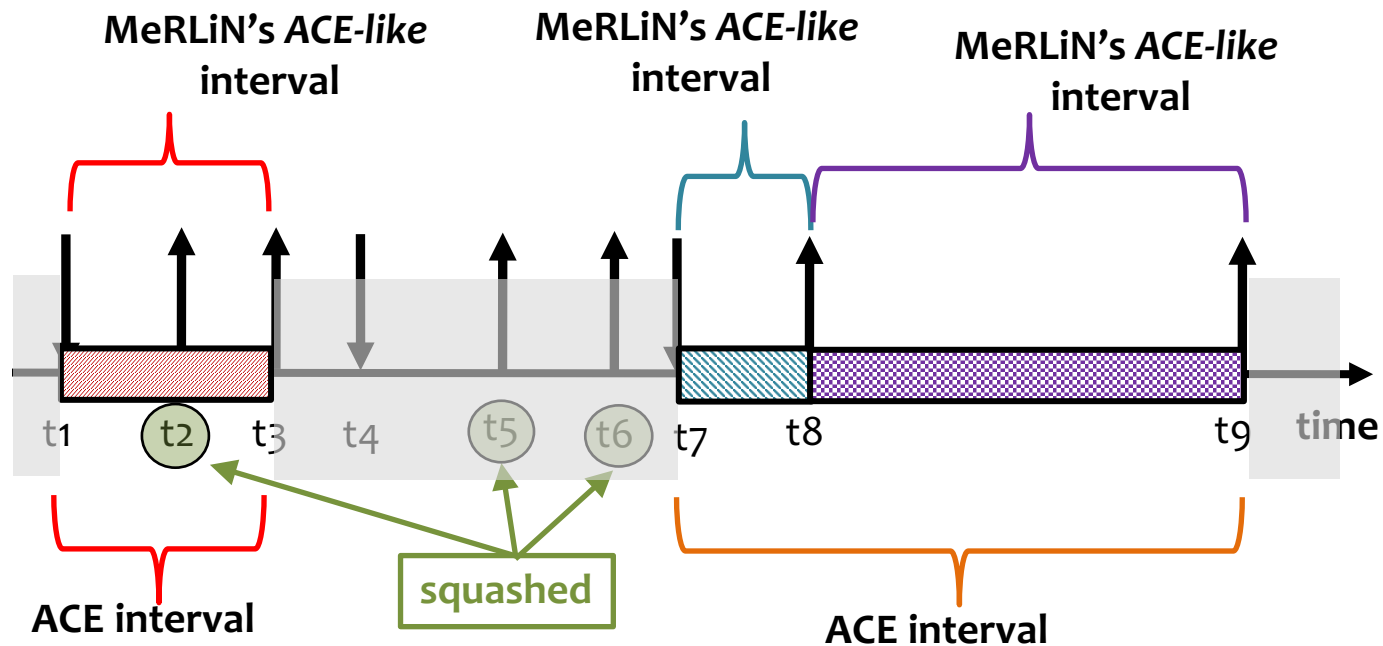
MeRLiN - Preprocessing (1)



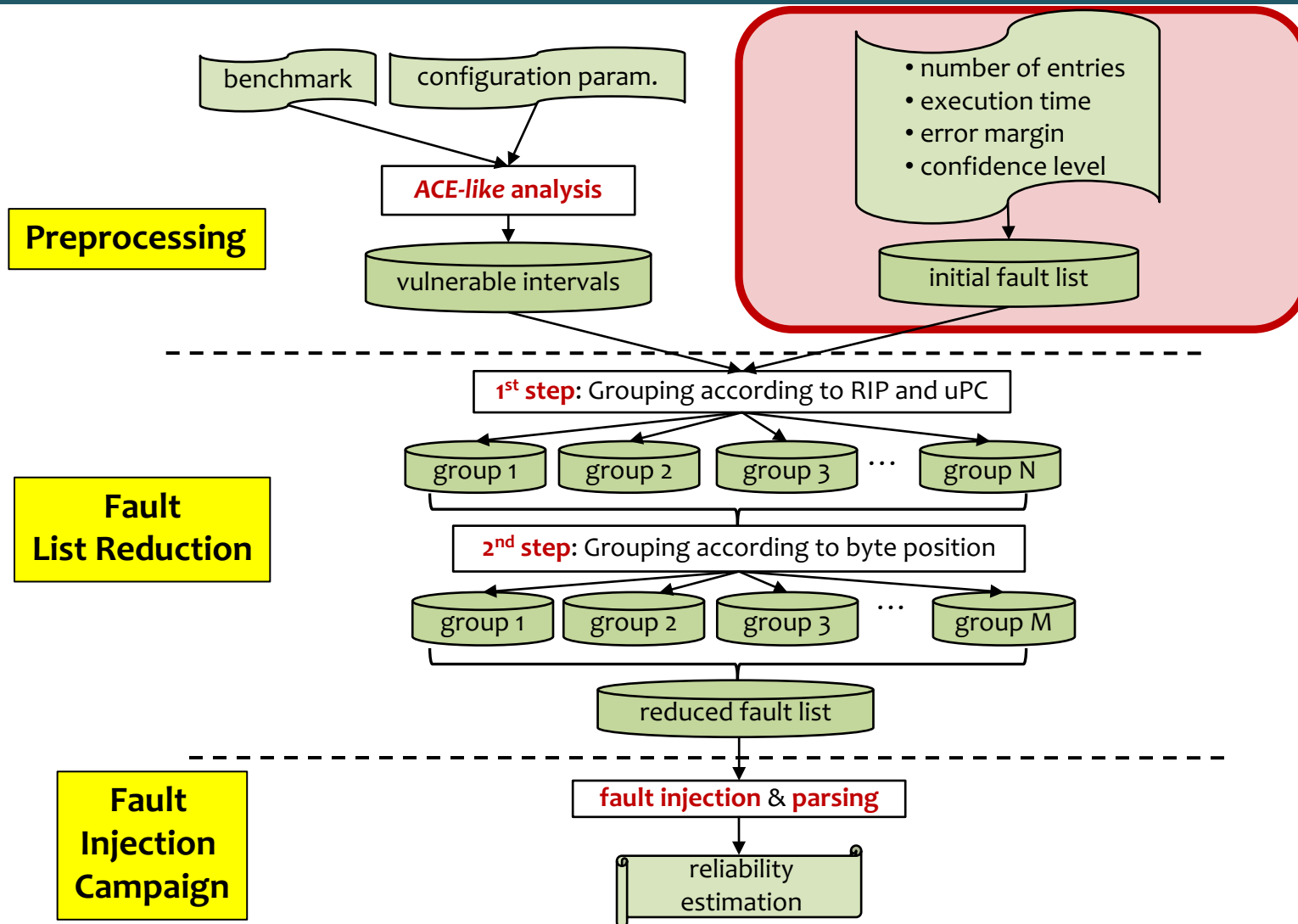
MeRLiN - Preprocessing (2)

1st task (**ACE-like analysis**)

- Single run to identify *vulnerable intervals*



MeRLiN - Preprocessing (3)



MeRLiN – Preprocessing (4)

2nd task (*Initial Fault List Creation*)

- Input of the statistical formula*:
 - **number of entries** of the targeted structure
 - **execution time in cycles** (known from the ACE-like task)
 - **error margin**
 - **confidence level**

- 60,000 faults* for all our fault injection campaigns:
 - ~0.63% error margin
 - 99.8% confidence level

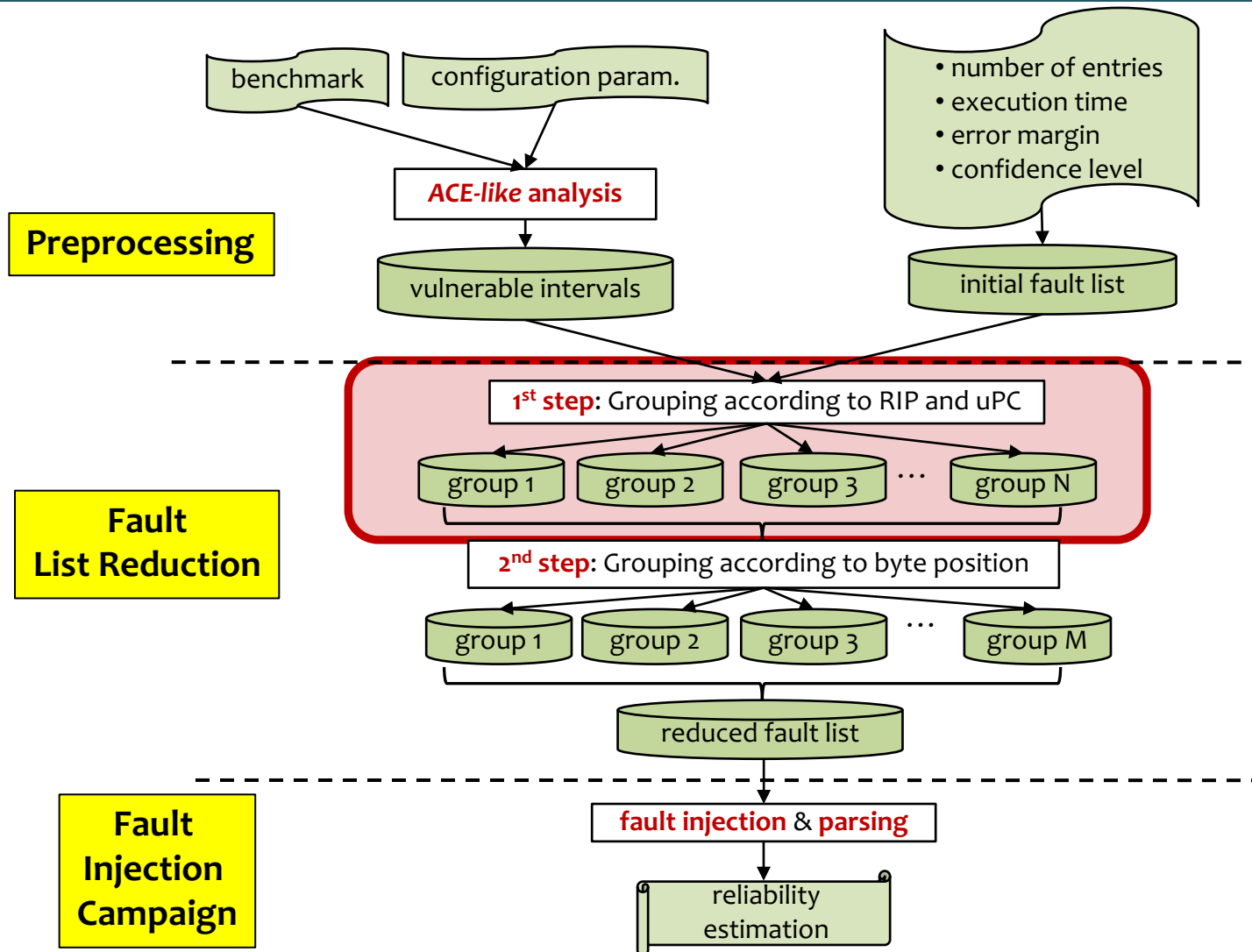
* R.Leveugle, et. al. “Statistical fault injection: Quantified error and confidence”, DATE 2009.



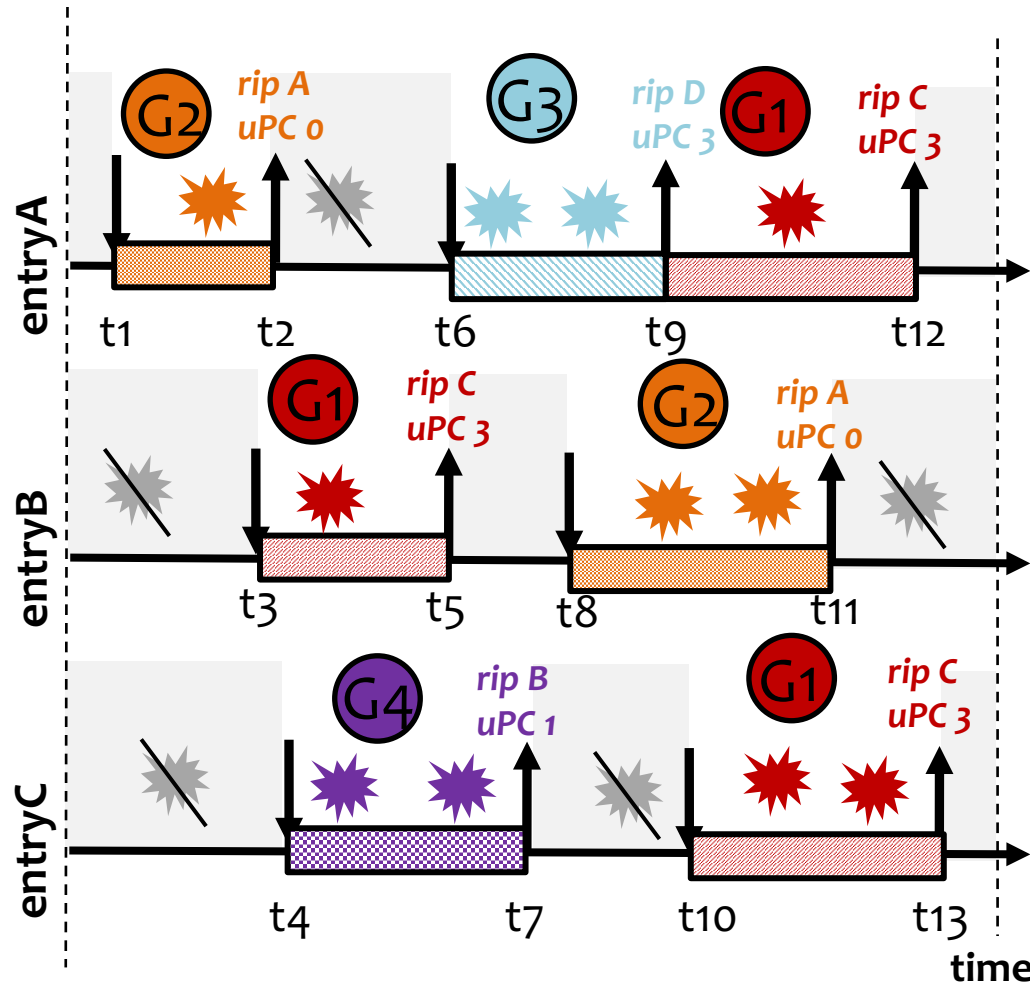
MICRO-50, Boston



MeRLiN - Fault List Reduction (1)



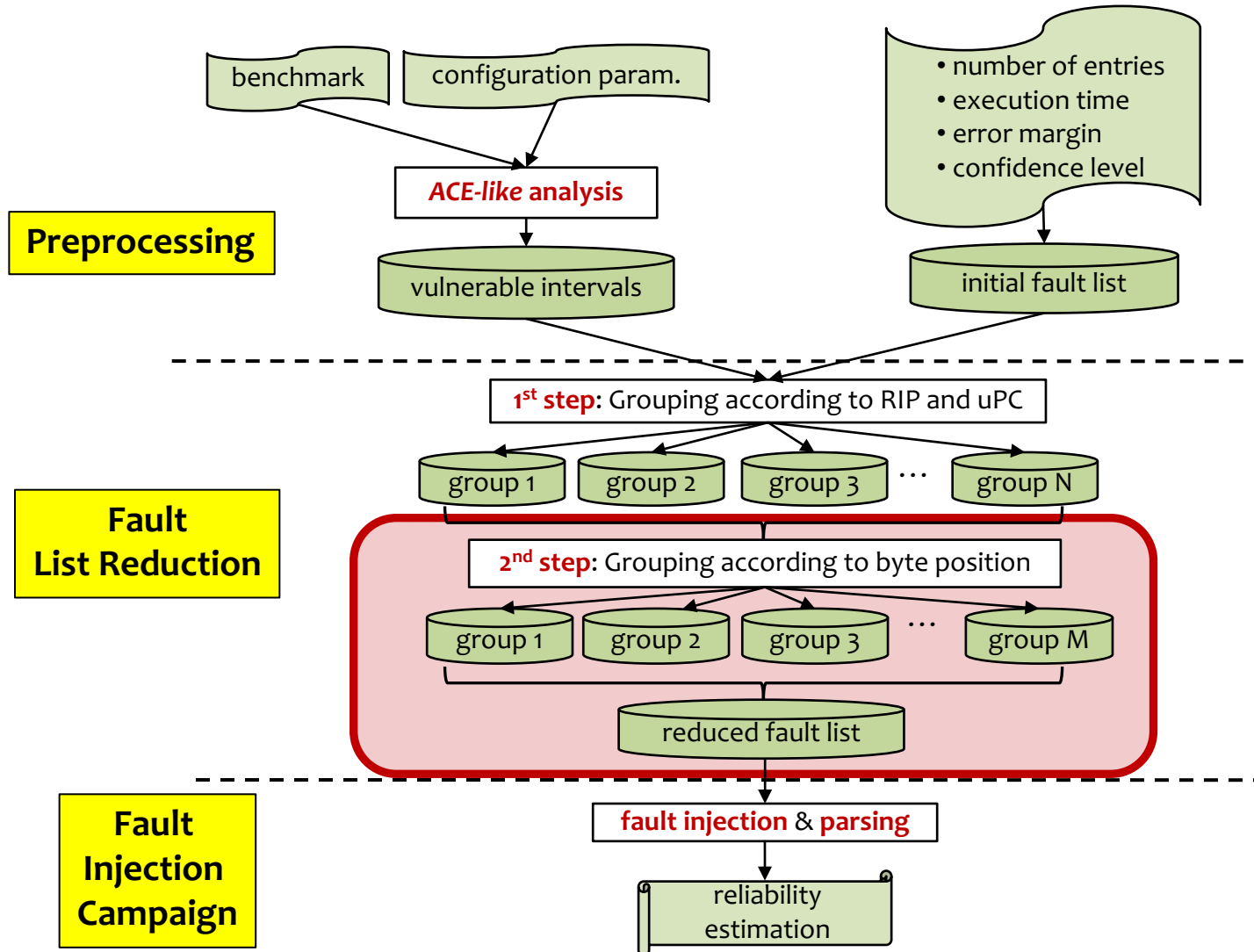
MeRLiN - Fault List Reduction (2)



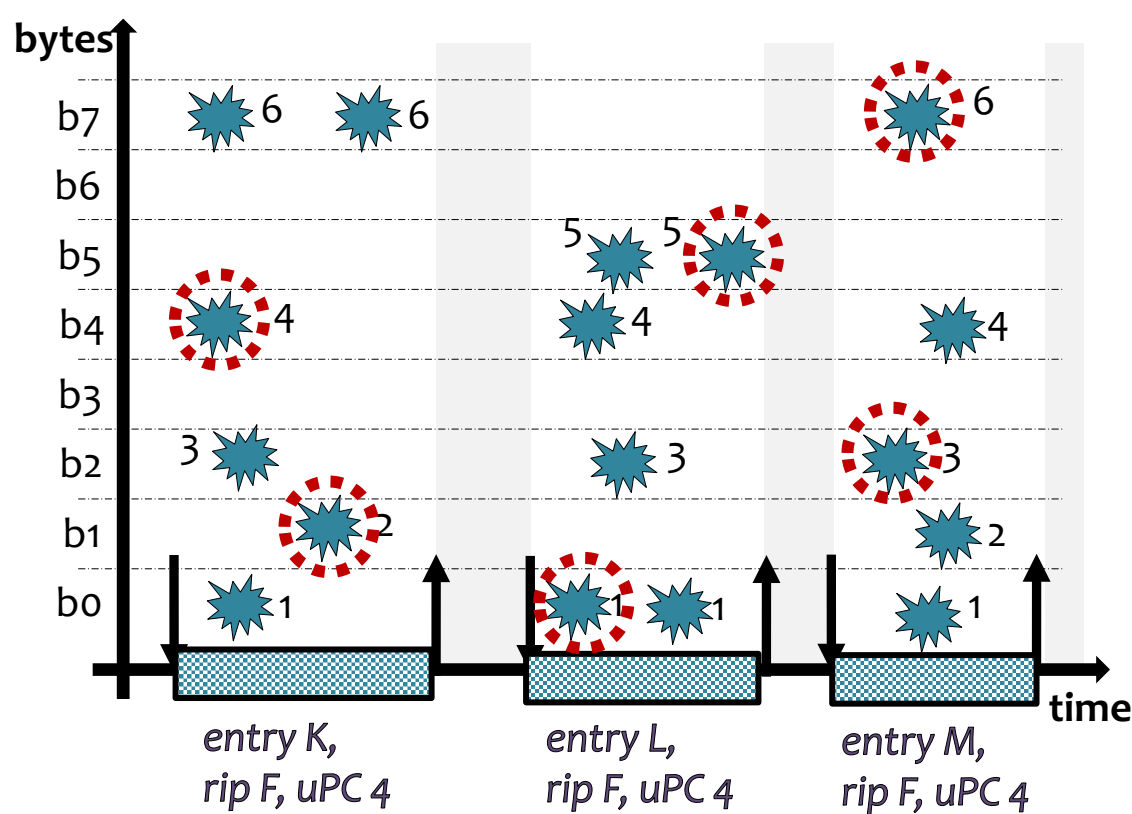
Group creation algorithm (**1st step**)

- Faults of non-vulnerable intervals are considered as **Masked**
- Group the rest faults according to the **RIP** and **uPC**

MeRLiN - Fault List Reduction (3)



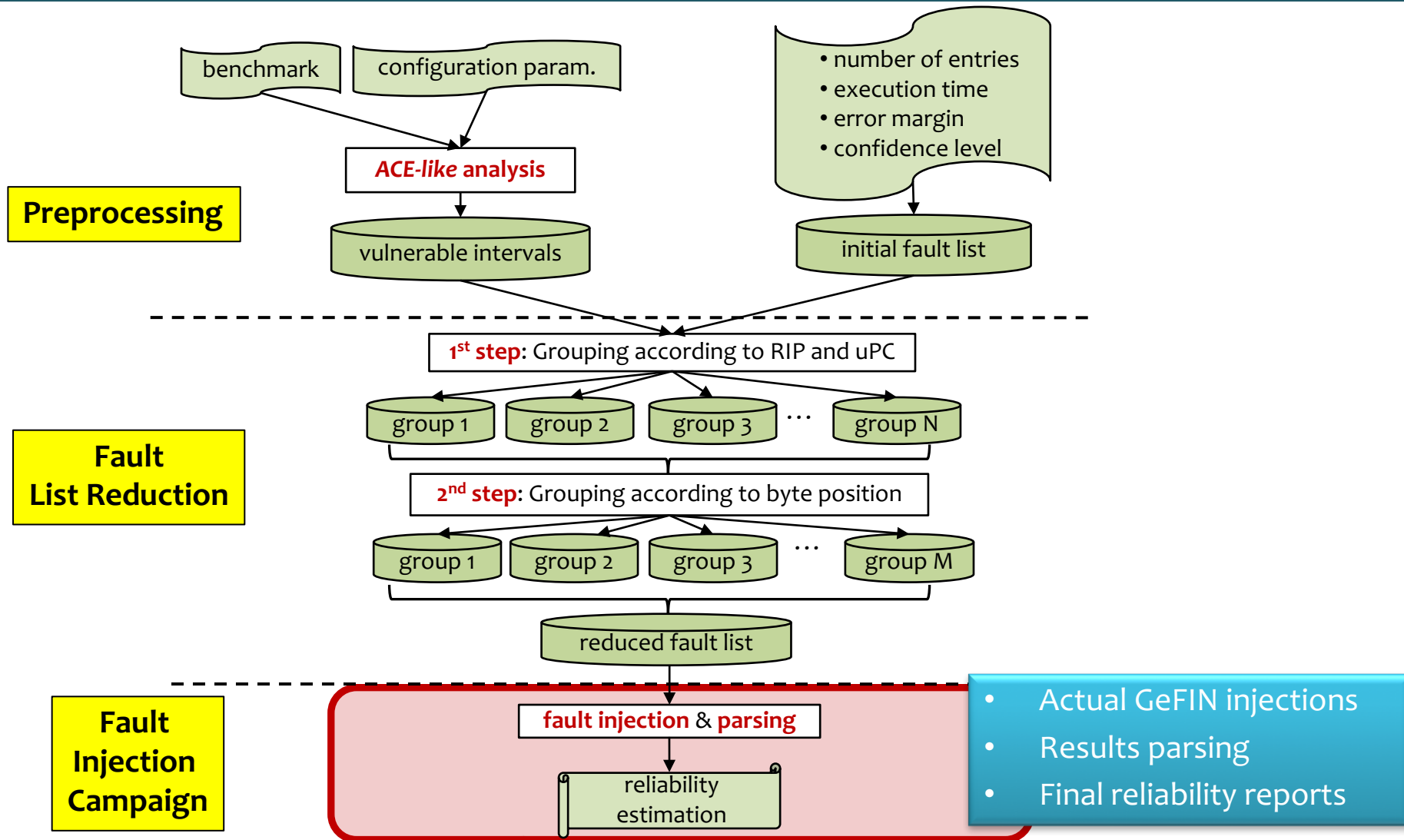
MeRLiN - Fault List Reduction (4)



Group creation algorithm (**2nd** step)

- To ensure accuracy for large groups, selection of faults from **different**:
 - **byte position**
 - **dynamic instances** of the same instruction
 - **hardware entries**
- **Smaller groups** that ensure **the final accuracy**

MeRLiN - Fault Injection Campaign



Experimental Setup and Benchmarks

Parameter	X86-64 microprocessor model configuration
Pipeline	OoO
Physical Int. Register File (RF)	256/128/64 registers
Physical FP. Register File	192 registers
Issue Queue entries	32
Store Queue (SQ)	64/32/16 store entries
Load Queue	64/32/16 load entries
ROB entries	100
Functional units	6 int ALUs; 2 complex int ALUs; 4 FP ALUs, 2 FP mul/div, 4 SIMD
L1 Instruction Cache	32KB, 64B line, 128 sets, 4-way, write back
L1 Data Cache (L1D)	16KB/32KB/64KB, 64B line, 64/128/256 sets, 4-ways, write back
L2 Cache	1MB, 64B line, 1024 sets, 16-way, write back
Branch Predictor	Tournament predictor
Branch Target Buffer	conditional and unconditional branches BTB (direct-mapped, 4K entries)

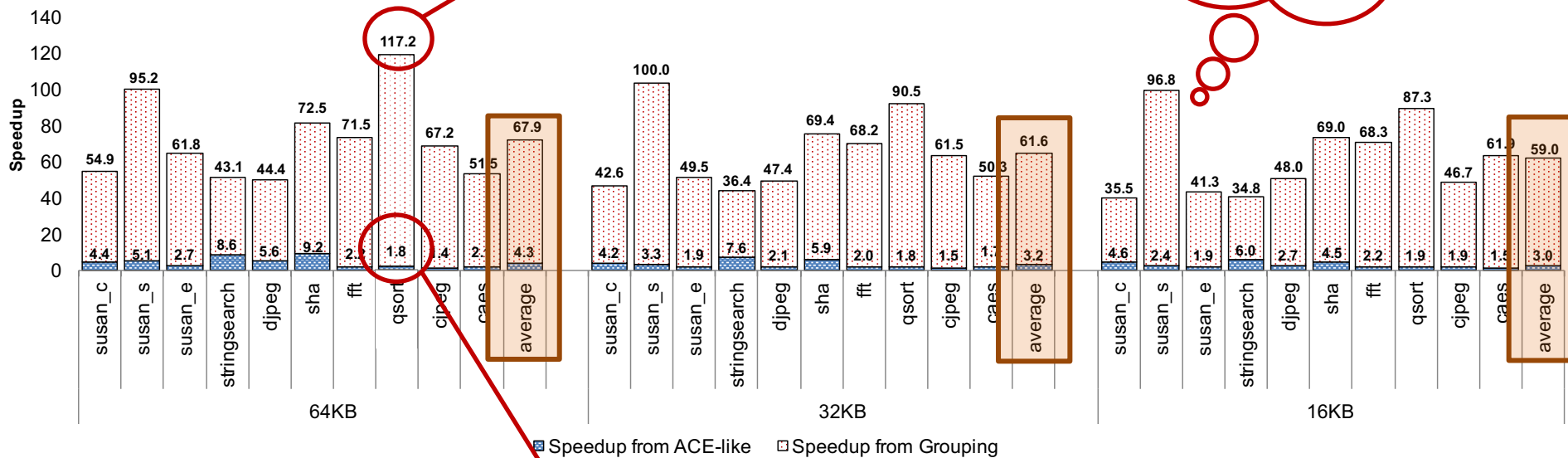
Target 3 structures:
RF (256/128/64 int. phys. registers)
SQ (64/32/16 entries)
L1D (16KB/32KB/64KB)

- **Speedup** and **accuracy** evaluation using:
 - ✓ **10 benchmarks** from **MiBench** suite (that **run to the end**)
 - ✓ **10 Simpoint samples of 100M** committed instructions from **SPEC CPU2006 suite**

Speedup (L1 Data Cache - MiBench)

$$\text{Speedup}_{\text{MeRLin}} = \frac{\text{Initial Faults (60K)}}{\text{Faults after MeRLin applied}}$$

Up to 2 orders of magnitude speedup for the **L1D** in some cases (e.g. qsort)



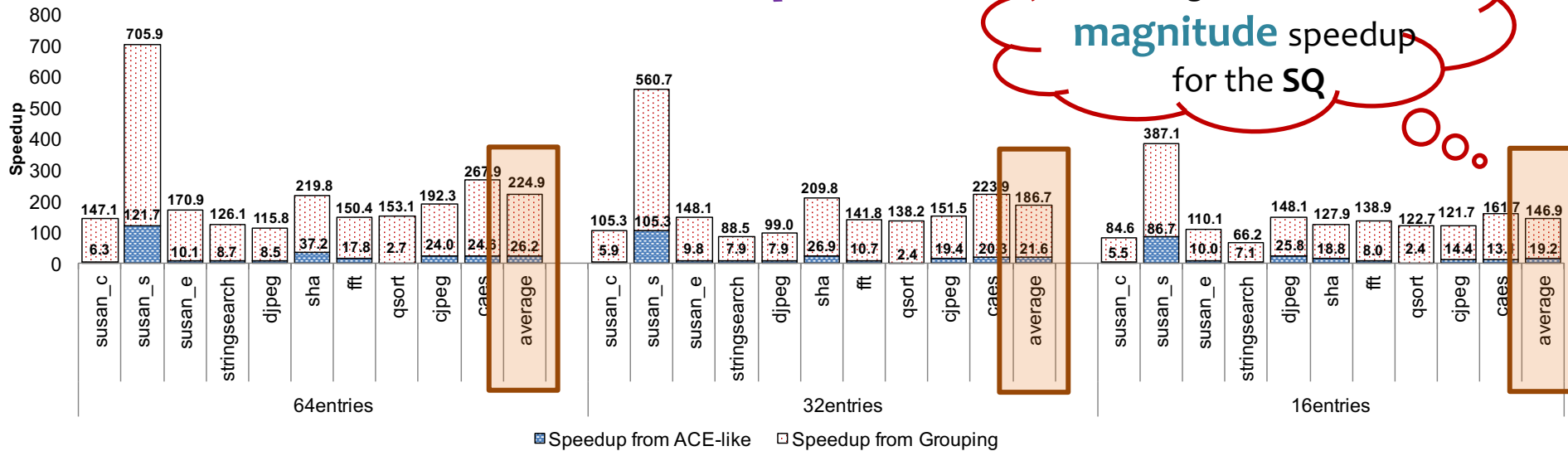
$$\text{Speedup}_{\text{ACE-like}} = \frac{\text{Initial Faults (60K)}}{\text{Faults after ACE-like step}}$$



Speedup (SQ - MiBench)

Store Queue

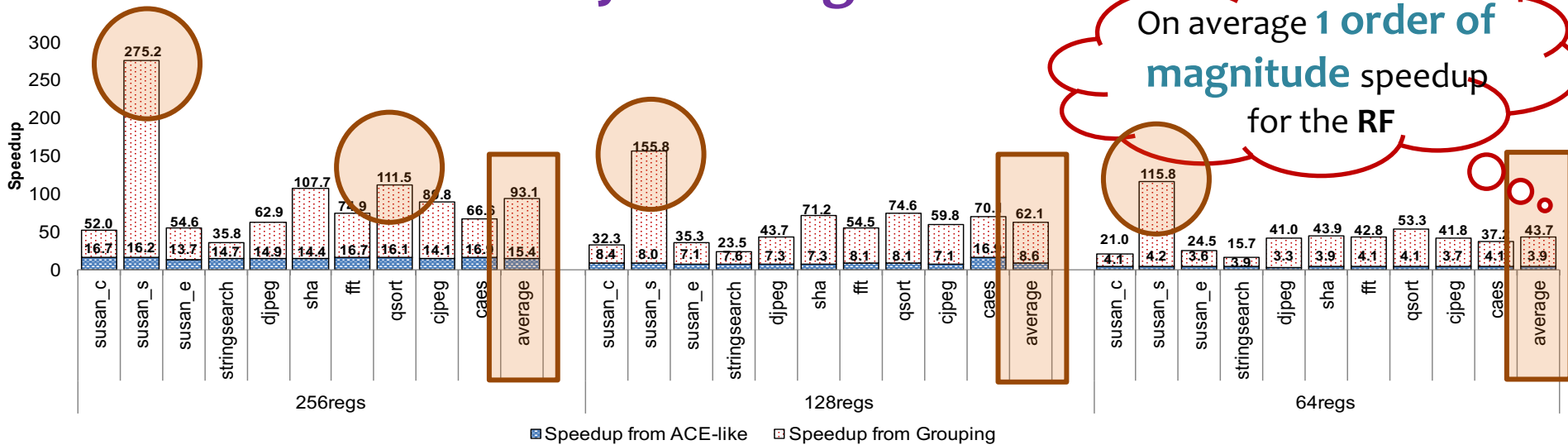
On average **2 orders of magnitude** speedup for the **SQ**



Speedup (RF - MiBench)

Physical Register File

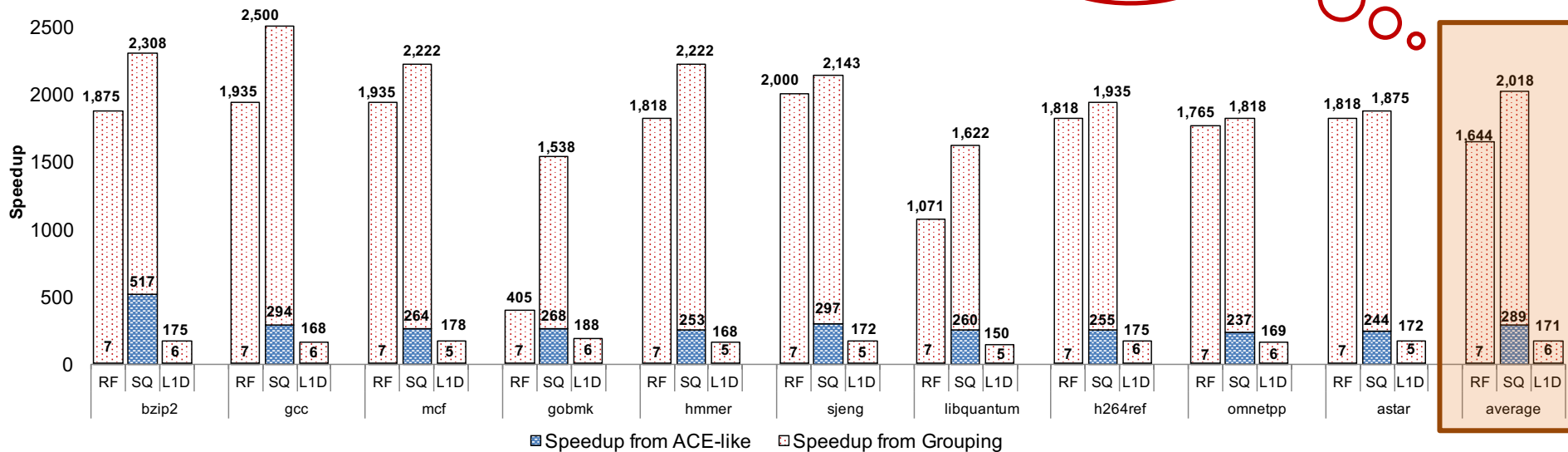
On average **1 order of magnitude** speedup for the RF



Speedup (RF, SQ & L1D – SPEC CPU 2006)

- 128 physical integer registers
- 16 store entries
- 32KB L1D cache

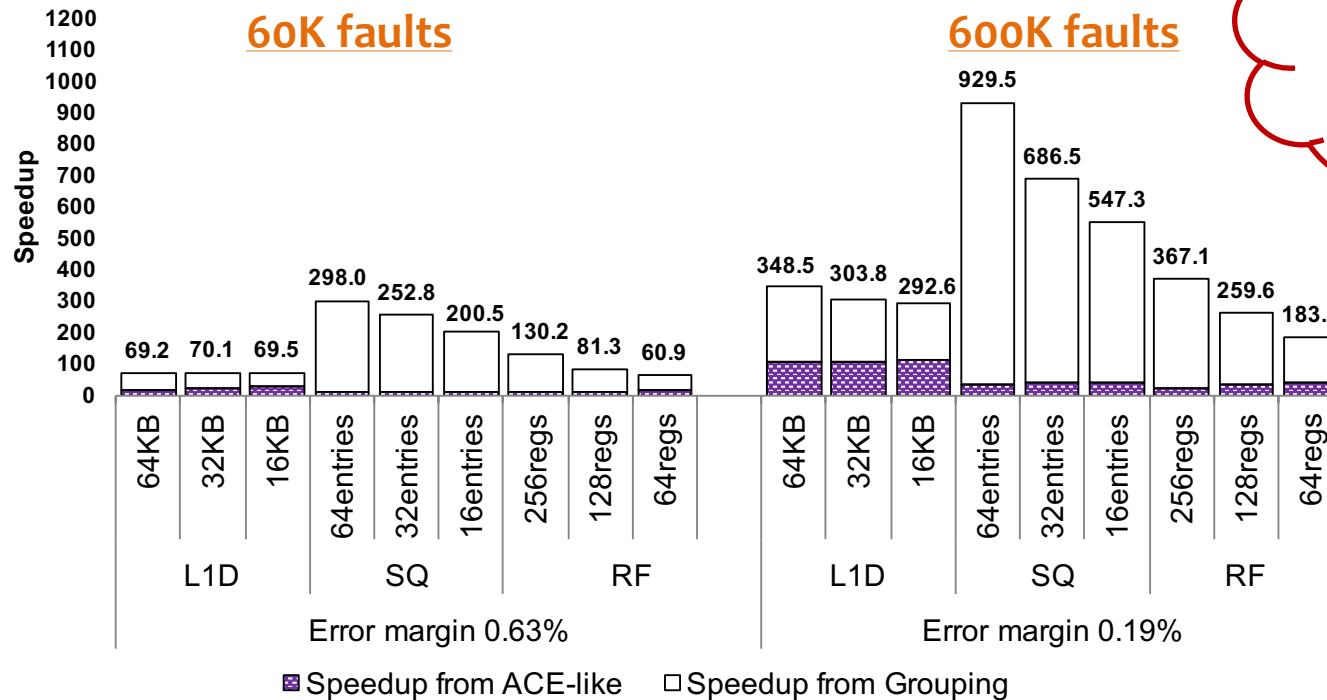
Up to **3 orders of magnitude** speedup using SPEC



Scaling of MeRLiN

- Using 10 MiBench and:

- **60K faults** (99.8% confidence level, ~0.63% error margin)
- **600K faults** (99.8% confidence level, ~0.19% error margin)

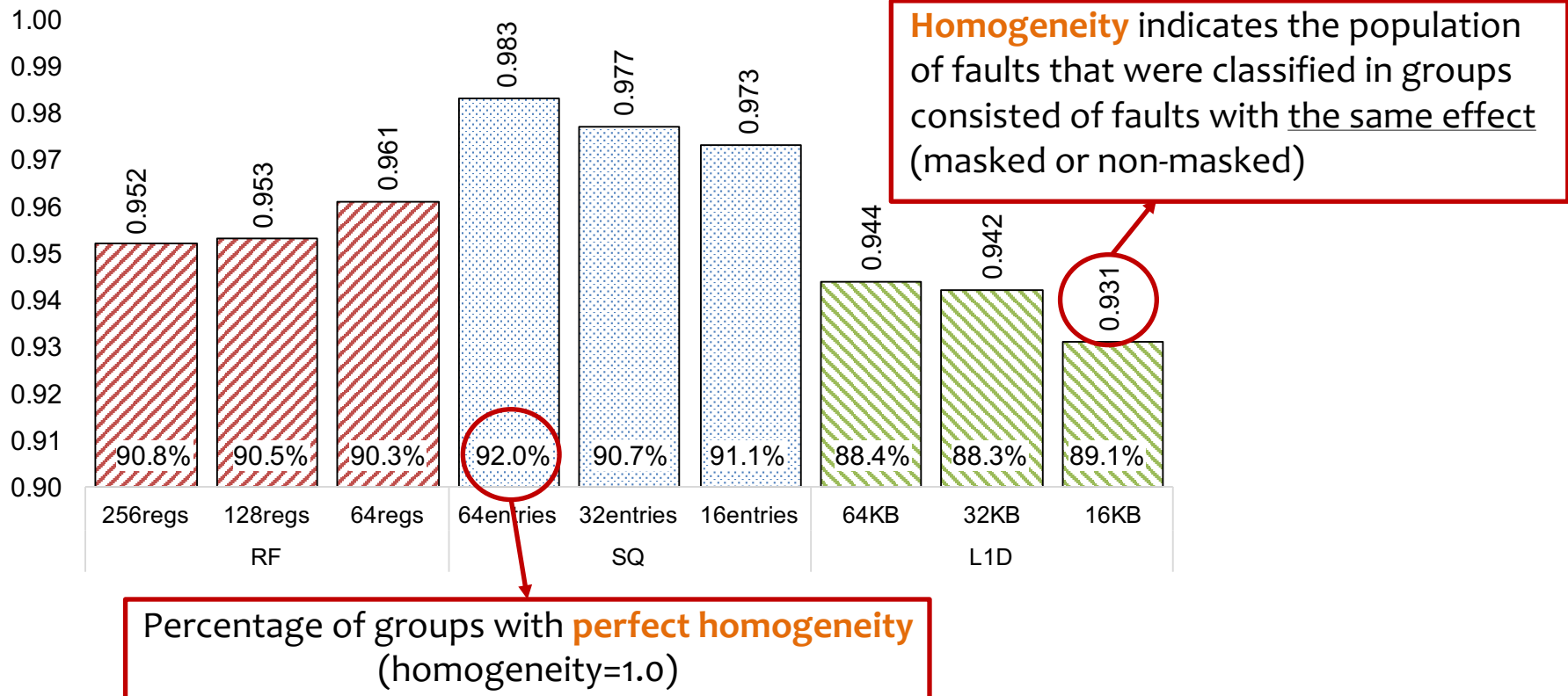


For **10 times increase** of the initial fault list, MeRLiN applies **only 2.89 times** more faults

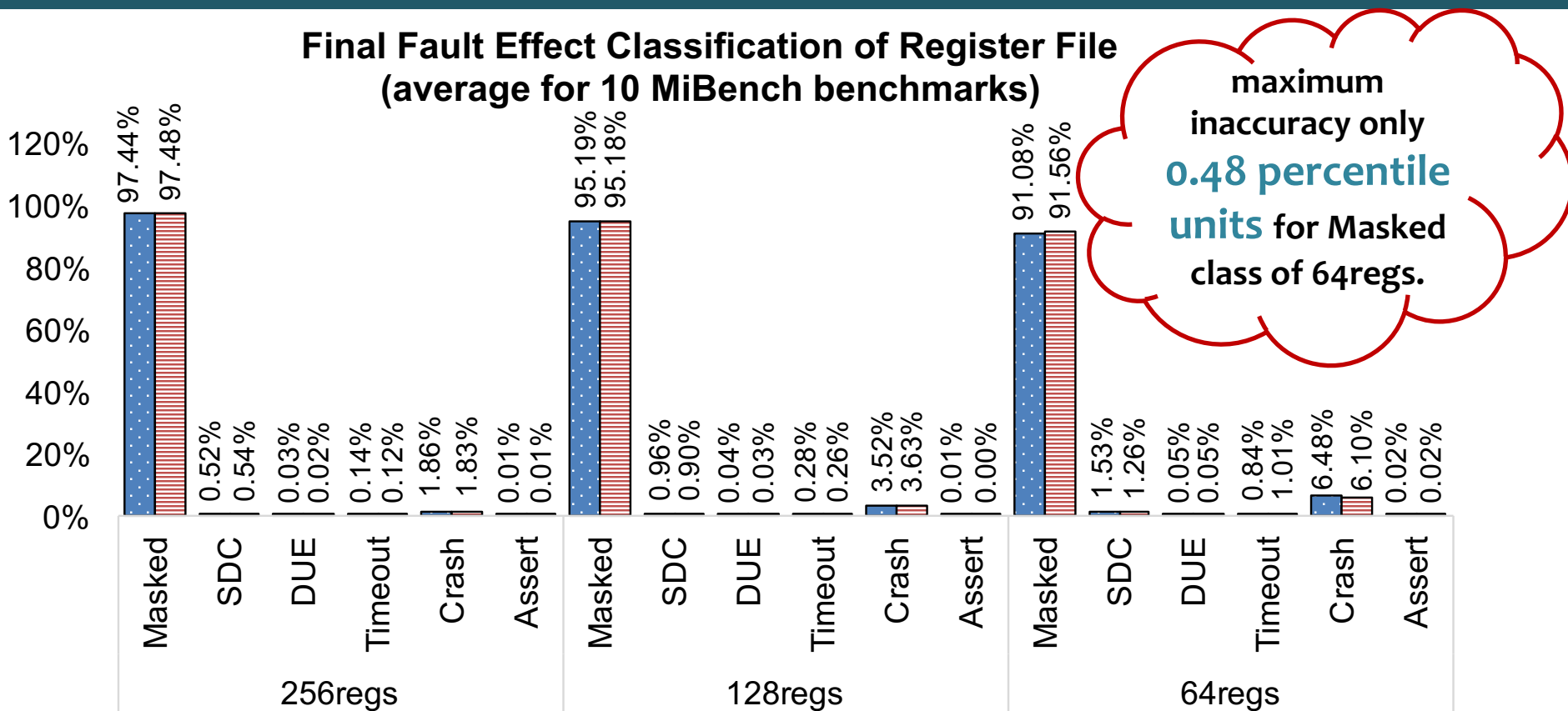
Homogeneity of the groups

- Using only two categories (**masked** & **non-masked**) instead of six
- Inject all the **remaining faults** after the ACE-like step

Homogeneity using only masked and non-masked categories



Reliability Estimation Accuracy - Final (RF - MiBench)



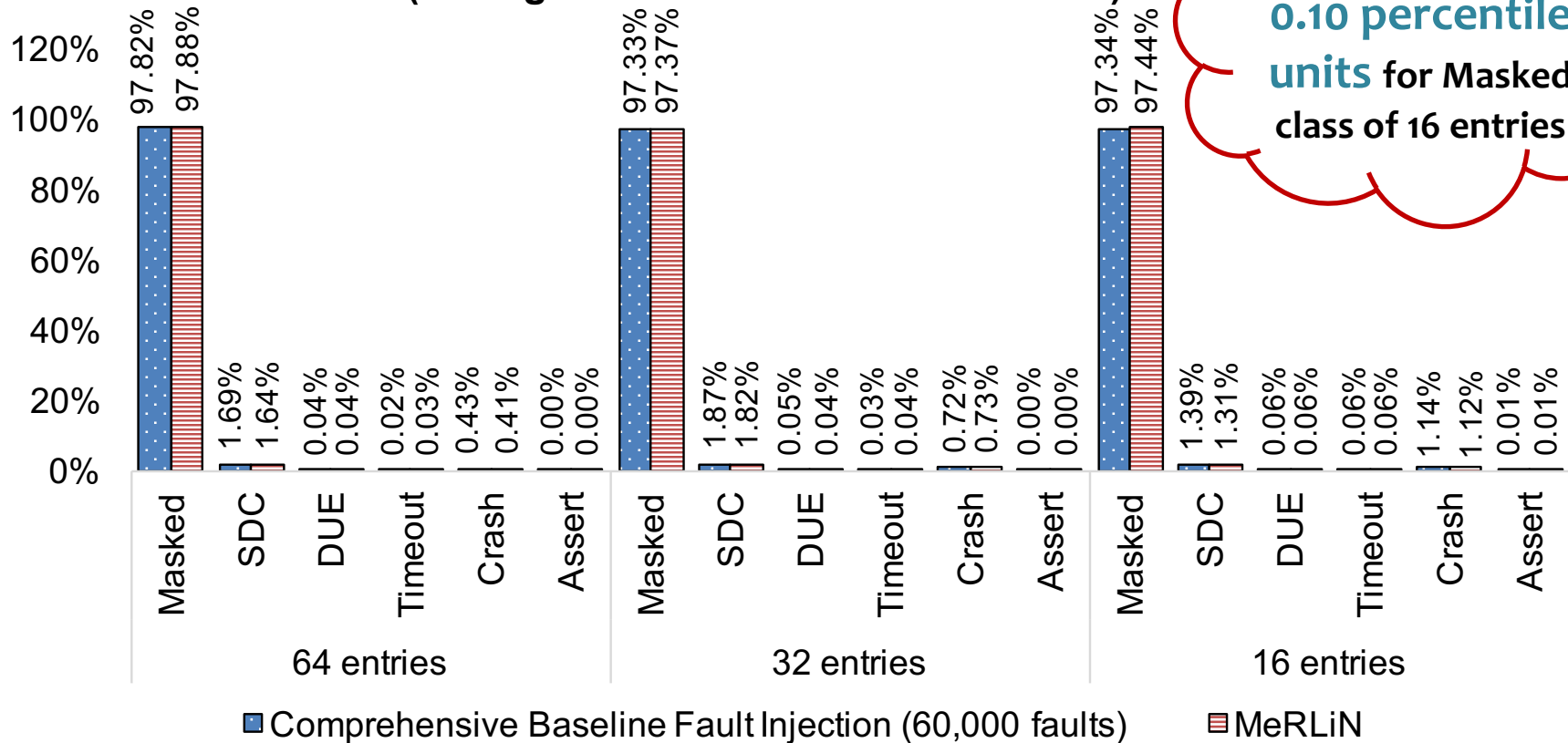
■ Comprehensive Baseline Fault Injection (60,000 faults) ■ MeRLiN

- **Blue bars:** accuracy when we inject **all the 60,000 faults**
- **Red bars:** accuracy **after applying MeRLiN** using the 60,000 faults



Reliability Estimation Accuracy - Final (SQ - MiBench)

Final Fault Effect Classification of Store Queue
(average for 10 MiBench benchmarks)

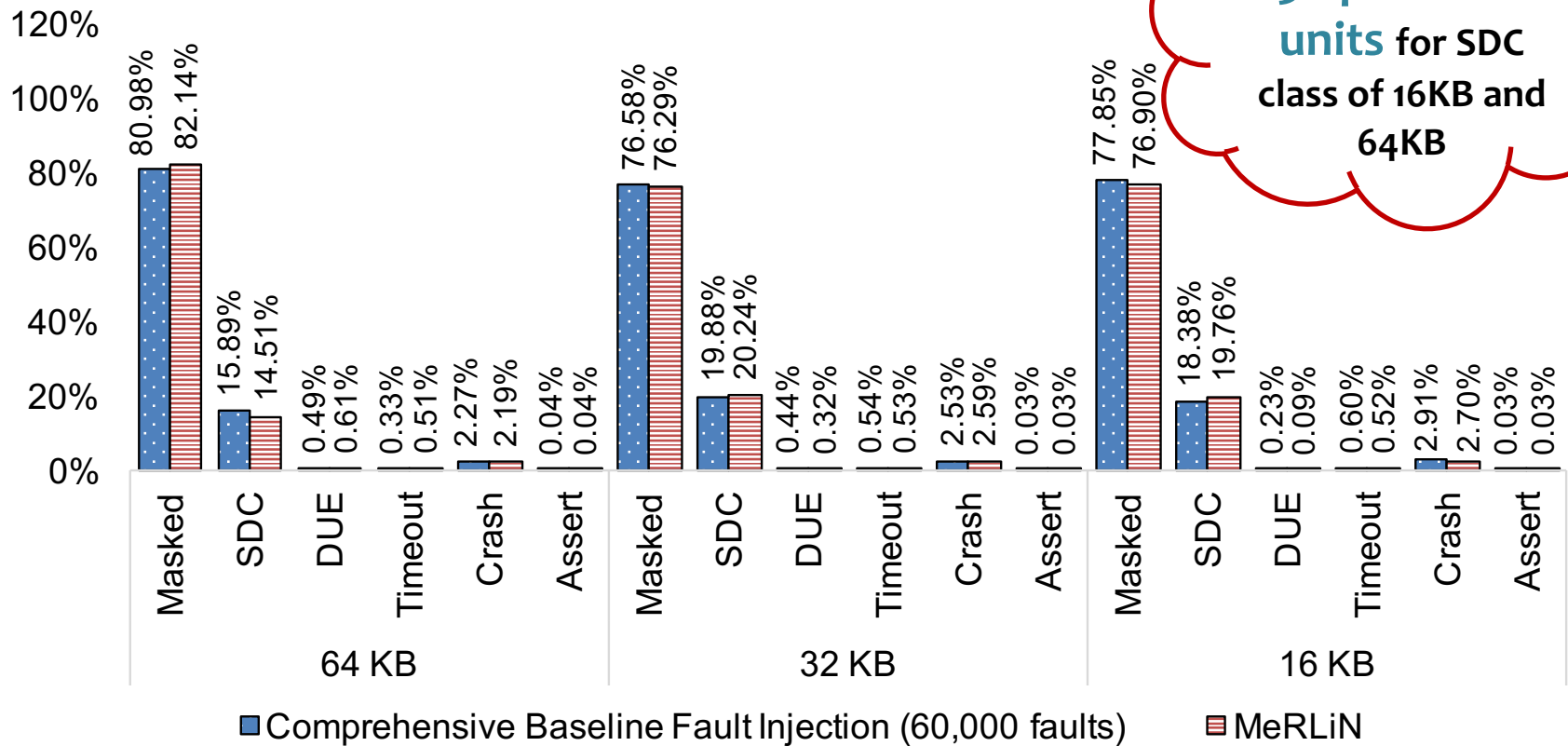


maximum
inaccuracy only
**0.10 percentile
units** for Masked
class of 16 entries



Reliability Estimation Accuracy - Final (L1D - MiBench)

Final Fault Effect Classification of L1 data cache
(average for 10 MiBench benchmarks)



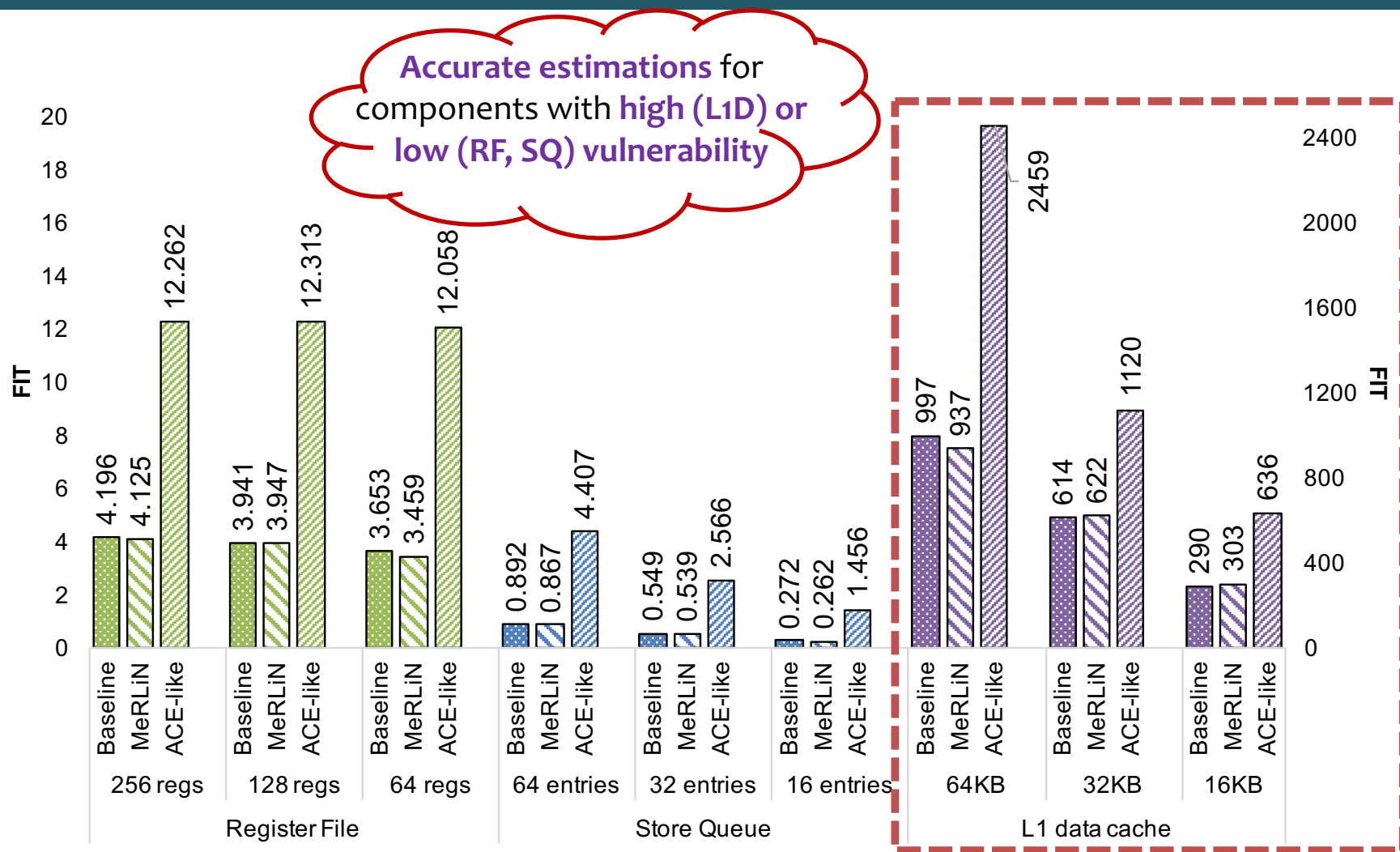
Final Reliability Assessment (in FIT) (1)

- **Final reliability in FIT (Failures In Time) :**
 - 1) **Baseline:** fault injection of all the 60K faults
 - 2) **MeRLiN:** same initial 60K fault list as in the Baseline
 - 3) **ACE-like:** from the *ACE-like* task of MeRLiN's flow

$$\text{FIT}_{\text{Component}} = \text{raw FIT} \times \text{Size (in bits)} \times \text{AVF},$$

where **raw FIT** = 0.01 FIT/bit
(any value can be used)

Final Reliability Assessment (in FIT) (2)





MICRO-50, Boston, October 2017

Next ...
Part 5

Tutorial:

Microarchitecture Level Reliability Assessment

Throughput and Accuracy
<http://micro50-tutorial.di.uoa.gr/>

Organizers/Presenters:

Athanasios Chatzidimitriou, Manolis Kaliorakis, Dimitris Gizopoulos

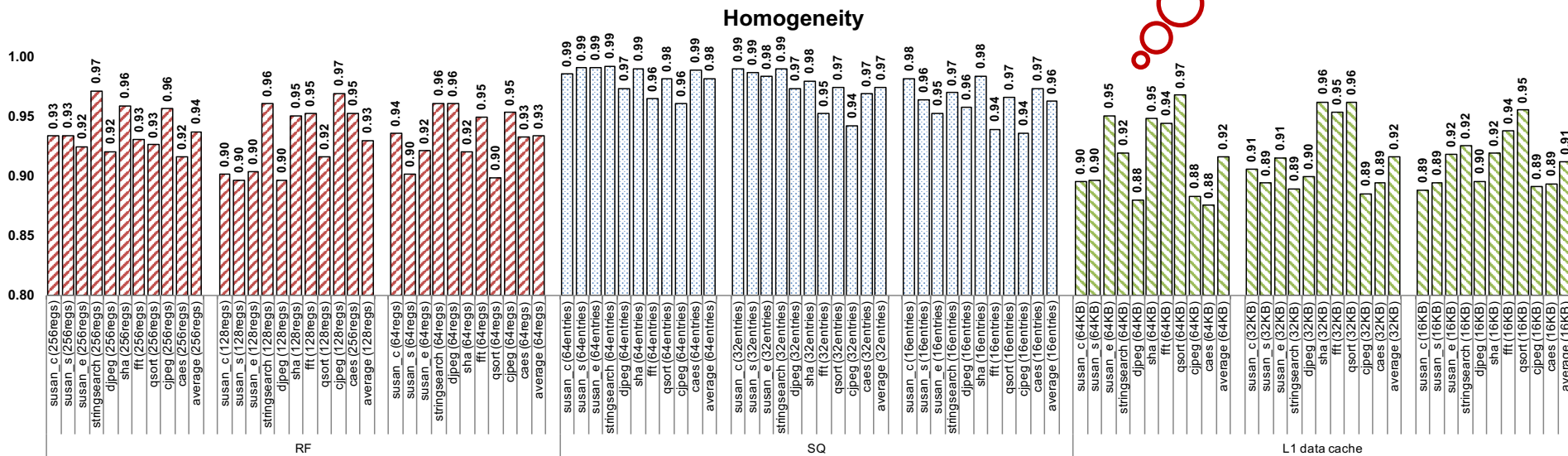
Backup

Homogeneity of the groups – Six classes

- To evaluate the accuracy of the grouping algorithm after the ACE-like step, we defined **homogeneity**:

$$\text{homogeneity} = \frac{\sum_{\text{group } l}^{\text{group } N} \# \text{faults} \times \text{dominant_class} \%}{\# \text{total_faults} \times 100 \%}$$

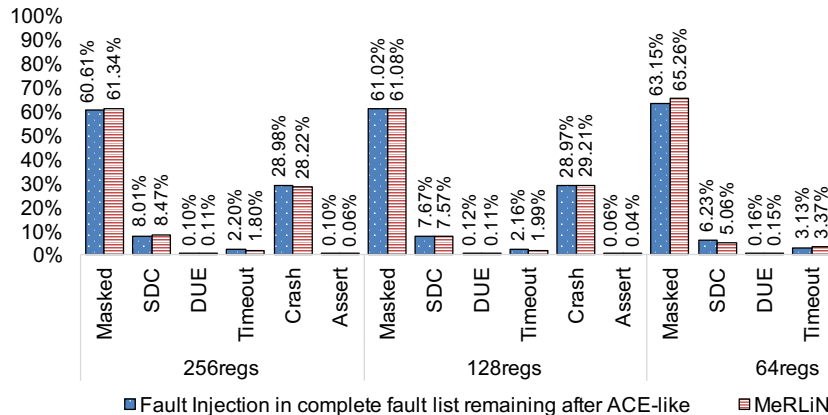
Using the **six fault effect categories** the homogeneity is very **high (>0.88)**



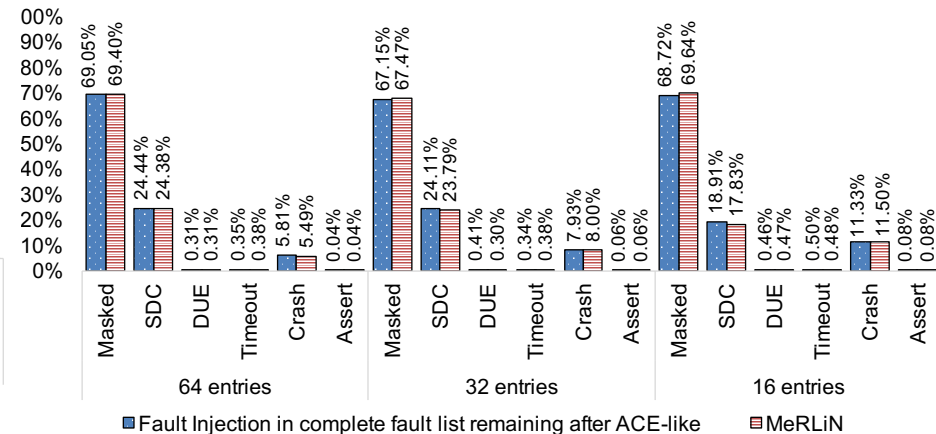
Reliability Estimation Accuracy – After ACE-like (RF, SQ & L1D - MiBench)

- Excluding all the faults that are detected by ACE-like as Masked

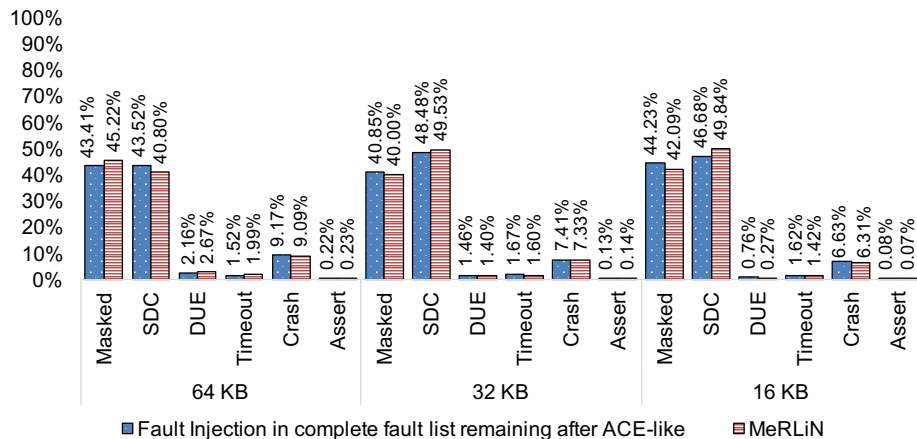
Fault Effect Classification after ACE-like of Register File
(average for 10 MiBench benchmarks)



Fault Effect Classification after ACE-like of Store Queue
(average for 10 MiBench benchmarks)



Fault Effect Classification after ACE-like of L1 data cache
(average for 10 MiBench benchmarks)



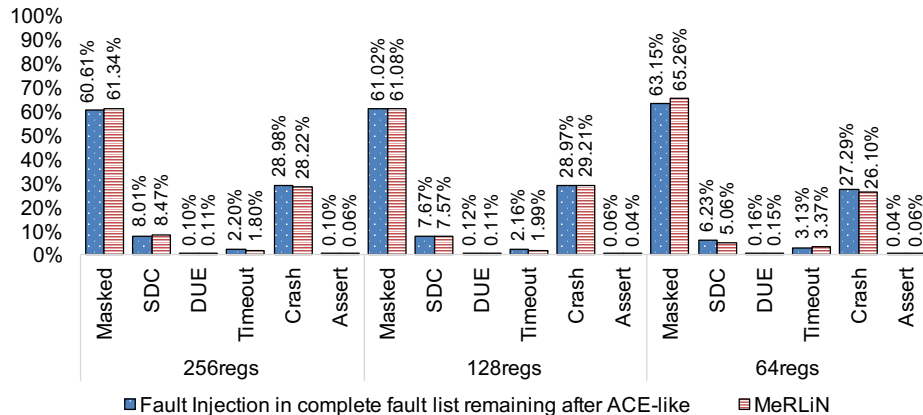
- Blue bars:** accuracy when we inject all the remaining faults after ACE-like step
- Red bars:** accuracy after MeRLiN's grouping algorithm in the same fault that (after ACE-like)



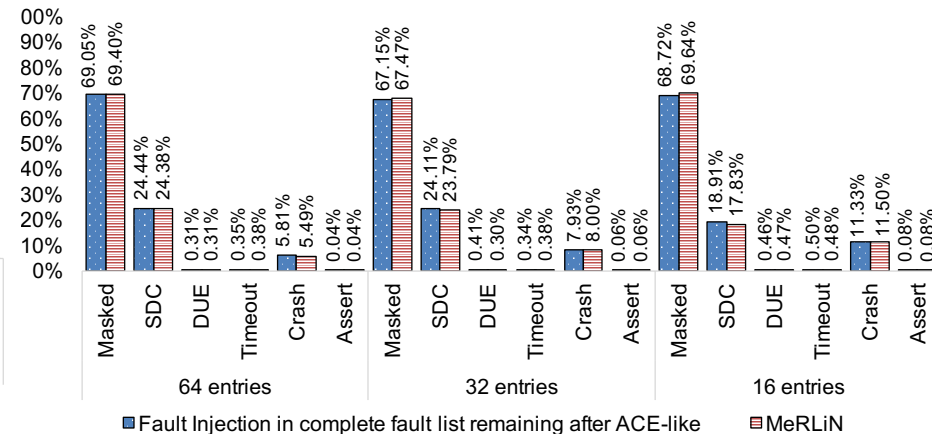
Reliability Estimation Accuracy – After ACE-like (RF, SQ & L1D - MiBench)

- Excluding all the faults that are detected by ACE-like as Masked

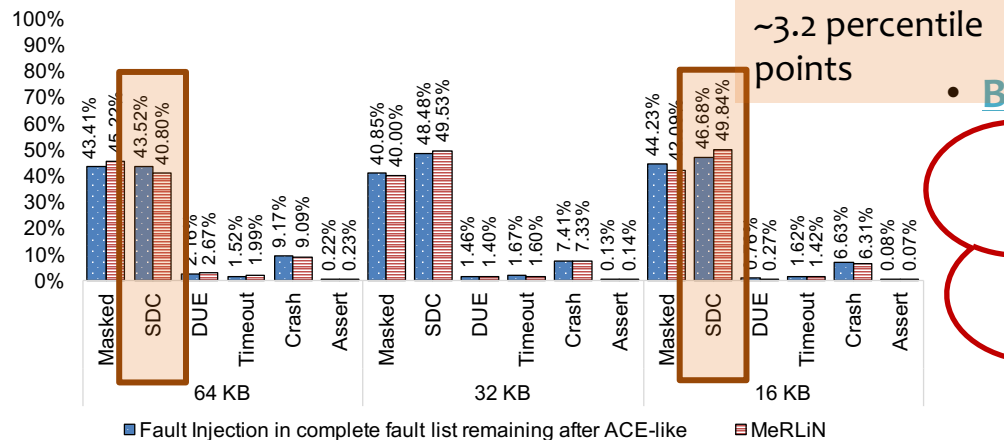
Fault Effect Classification after ACE-like of Register File
(average for 10 MiBench benchmarks)



Fault Effect Classification after ACE-like of Store Queue
(average for 10 MiBench benchmarks)



Fault Effect Classification after ACE-like of L1 data cache
(average for 10 MiBench benchmarks)



MeRLiN provides accurate results even for the remaining faults **after the ACE-like step**

Reliability Estimation Accuracy (RF – SPEC CPU 2006)

- Evaluation of MeRLiN accuracy running **bzip2** and **gcc Simpoint samples from SPEC CPU2006**
- For **RF with 128 registers**, **SQ with 16 store entries** and **32KB L1D**
- Modifications concerning the fault effect classification (**5 classes instead of 6**)

Category	Effect
Masked	A fault was over-written or hit in non-vulnerable interval
DUE	Extra ISA exception raised
Crash	Process, system or simulator crash
Assert	Termination due to assert instruction
Unknown	At the end of the Simpoint interval it is not known if the fault will eventually be classified in one of the previous classes. Includes SDCs and Timeouts.

Category	gcc (MeRLiN)	gcc (baseline 60K faults)	bzip2 (MeRLiN)	bzip2 (baseline 60K faults)
Masked	85.08%	85.08%	84.98%	84.98%
DUE	0.06%	0.07%	0.29%	0.81%
Crash	3.67%	3.13%	3.50%	4.10%
Assert	0.01%	0.01%	0.03%	0.02%
Unknown	11.18%	11.71%	11.20%	10.09%

Maximum inaccuracy = 1.1 percentile points (Unknown class of bzip2)



MICRO-50, Boston



Relyzer vs MeRLiN

Relyzer	MeRLiN
injects faults in the software level	injects faults in the microarchitecture level
identifies the SDCs and the Crashes	identifies DUE, SDC, Crash, Assert, Timeout
computes only the Program Vulnerability factor (PVF)	computes the Architectural Vulnerability factor (AVF) AVF = PVF + HVF (Hardware Vulnerability factor)

- Relyzer heuristics evaluation:

- Bounding addresses:**

- Costly to track addresses in caches
- MeRLiN provides finer grained effect classification for non-masking categories

- Def-Use & Store-equivalence:**

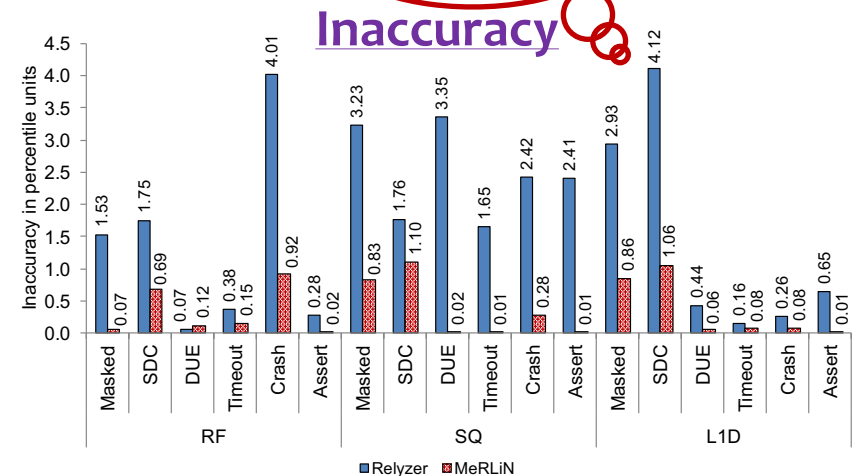
- Applicable only in the software level

- Control-equivalence** (10 MiBench):

- RF with 128 physical registers
- SQ with 16 entries
- L1D of 32KB

Speedup: very close but MeRLiN slightly prevails in L1D and RF

Relyzer selects only **one pilot** for large groups of the same static instruction, while MeRLiN selects **more pilots** from different bytes, dynamic instances and entries



MICRO-50, Boston



Statistical Analysis

- Compare the **mean** and the **variance** of MeRLiN with comprehensive fault injection (60,000 faults)

1st conclusion (mean of MeRLiN is equal to the mean of comprehensive injection)

$$E(k_{MeRLiN}) = E(k) = \frac{\sum_{i=1}^n s_i \cdot p_i}{F}$$

Diagram annotations for the mean formula:

- $\sum_{i=1}^n$: size of group
- s_i : population of final groups
- p_i : probability of selecting a fault from a group
- F : population of initial fault list (60K faults)

2nd conclusion (variances of MeRLiN and of comprehensive injection are from 6 to 8 orders of magnitude smaller than the mean)

$$\sigma^2(k) = \frac{\sum_{i=1}^n s_i \cdot p_i \cdot (1 - p_i)}{F^2}$$

$$\sigma^2(k_{MeRLiN}) = \frac{\sum_{i=1}^n s_i^2 \cdot p_i \cdot (1 - p_i)}{F^2}$$

Diagram annotations for the variance formulas:

- s_i (in $\sigma^2(k)$): very small (~100 faults per group)
- $p_i \cdot (1 - p_i)$ (in $\sigma^2(k)$): p_i or $1 - p_i$ are zero or small due to the high homogeneity of the groups
- F^2 (in $\sigma^2(k_{MeRLiN})$): large initial fault list (60K faults)