Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Software Time Reliability in the Presence of Cache Memories

Suzana Milutinovic, Jaume Abella, Irune Agirre, Mikel Azkarate-Askasua, Enrico Mezzetti, Tullio Vardanega and Francisco Cazorla

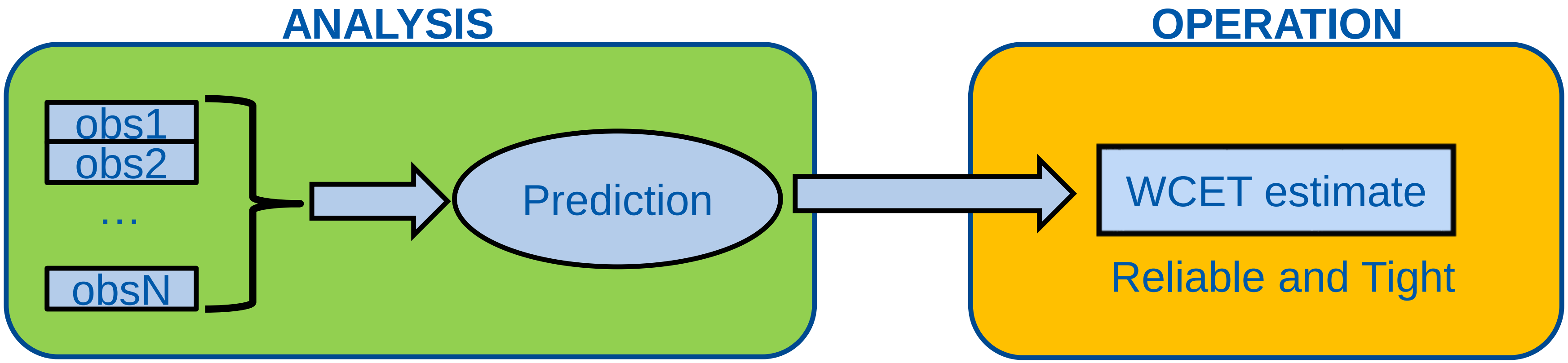
ADA-EUROPE 2017, Vienna, Austria

Real-Time Embedded Systems

Functional correctness	**Timing correctness**
Software performs its task	Software fits its assigned time budget

- Timing verification
  - Estimating the **W**orst-**C**ase **E**xecution **T**ime (WCET) of tasks
  - Finding the valid schedule of tasks

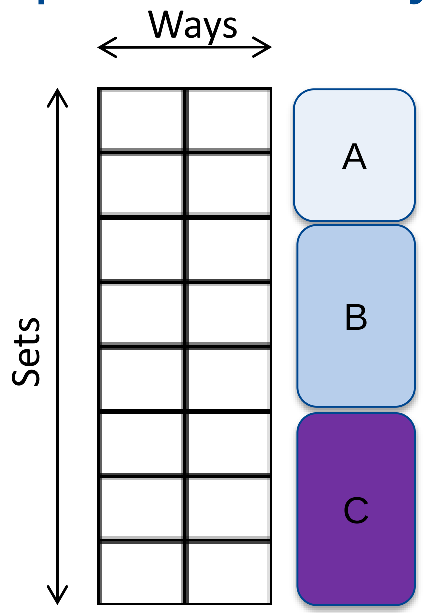
Measurement-Based Timing Analysis (MBTA)



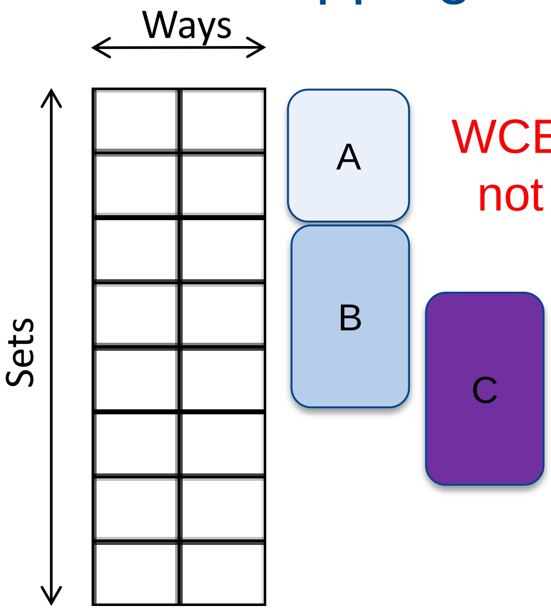
- Quality of WCET estimates depends on analysis measurements **representativeness**
  - User needs to capture worst conditions that can arise at operation
    - The worst-case behaviour of each resource with variable timing behaviour
    - Combined impact for all such resources

MBTA representativeness challenge

- Complex systems challenge achieving the required level of control to trigger worst conditions
  - Bus occupancy, data/code mapping in cache, etc.
  - Lower the confidence on WCET estimates
- Example of memory mapping → cache mapping



Cache placement @ Analysis

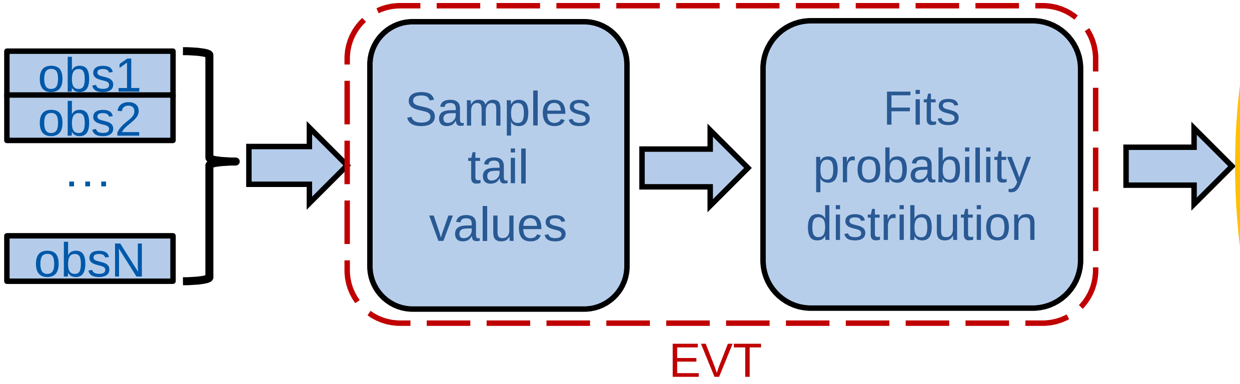


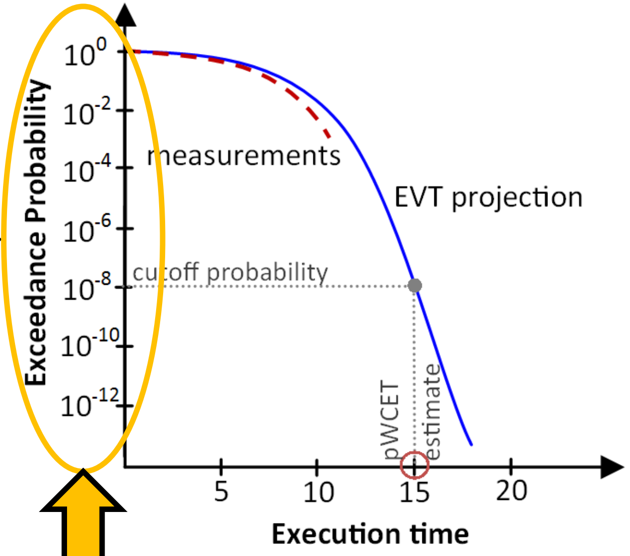
WCET estimates are not valid anymore!

Cache placement @ Operation

Measurement-Based Probabilistic Timing Analysis (MBPTA)

- Applies **Extreme Value Theory (EVT)** to the timing analysis





Quantitative confidence to pWCET estimates

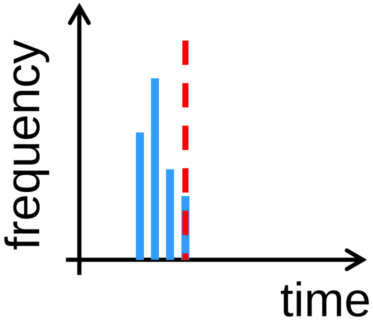
- Assuring measurements observations representativeness
  - User needs to capture worst conditions that can arise at operation
    - The worst-case behaviour of each resource with variable timing behaviour
    - ~~Combined impact for all such resources~~

Handled by EVT

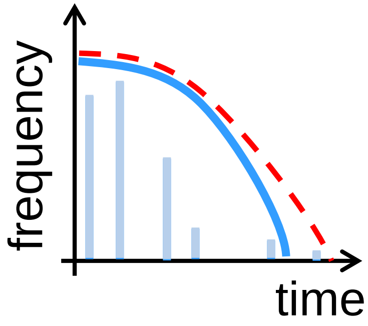
MBPTA and Representativeness

- Identify all resources with variable timing behaviour
  - Variable-latency FPU operation, cache behaviour, contention effects in multicores, …
- Bound their analysis-time behaviour

Exhibits **low** latency variation



Exhibits **high** latency variation



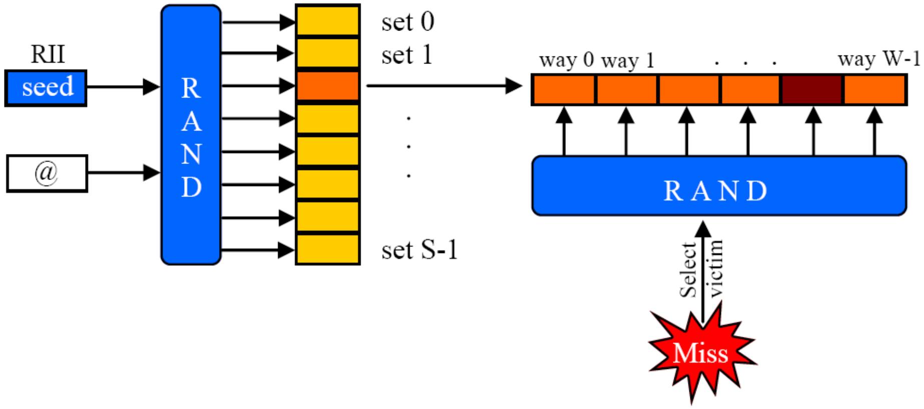
- Deterministic Upper-bounding
  - Force the resource to work in its worst latency
  - Relevant events captured in single run
- Probabilistic Upper-bounding
  - Time-randomization
  - Each event occurs with a probability Peoi
  - More runs -> more representativeness (not had with MBTA)

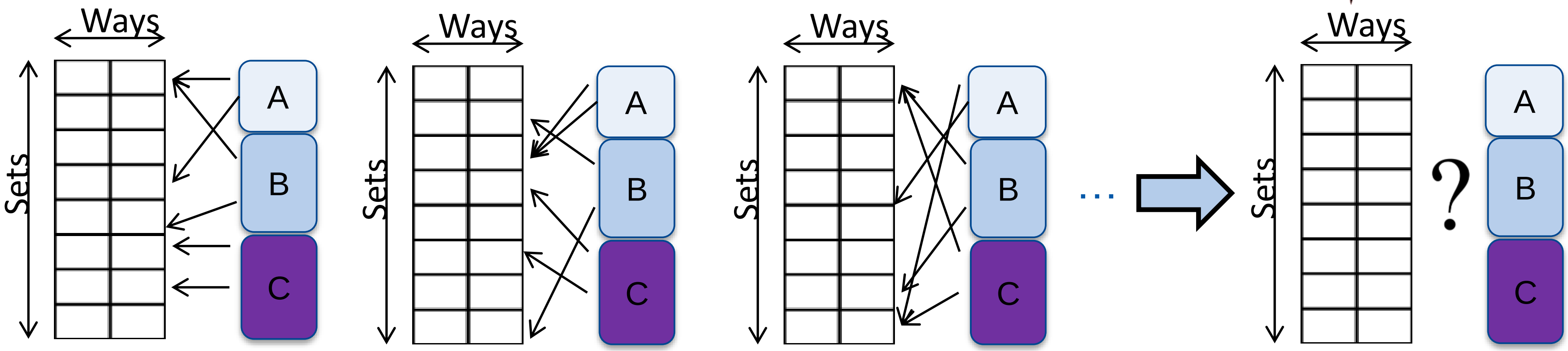
━ system operation

╌ system analysis

Representativeness: Cache memories

- Time-randomized (TR) caches
  - random placement and random replacement
  - user is not required to control memory layouts
  - favor incremental software integration





Analysis (1 run -> 1 **cache placement**)

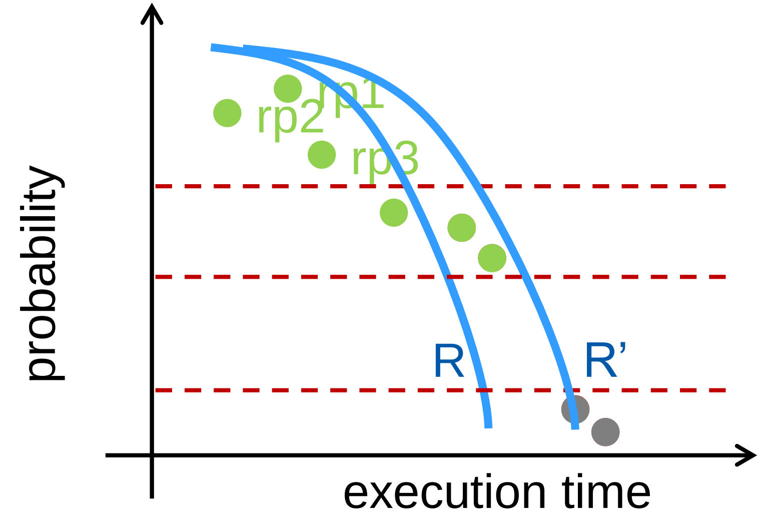
Operation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Cache-related representativeness issue

- Problem:
  - Certain address placements cause an abrupt change in miss counts (i.e. certain addresses if mapped to the same set cause an increase in miss count)
  - Occur with a probability
    - Low enough so that they are unlikely to be observed at analysis
    - High enough to be relevant

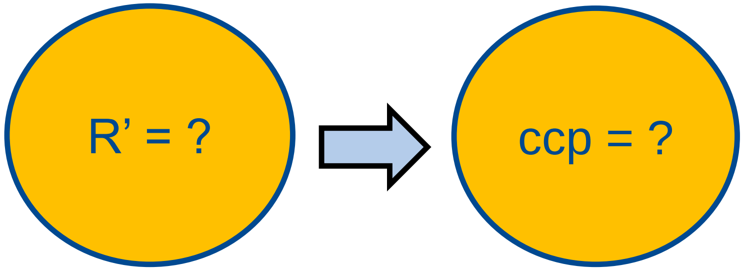
→ conflictive cache placements (ccp)



Pobs=0.021 for R=1000

R↑: Pobs ↓

Prel=10^{-9}
(dictated by safety standard)

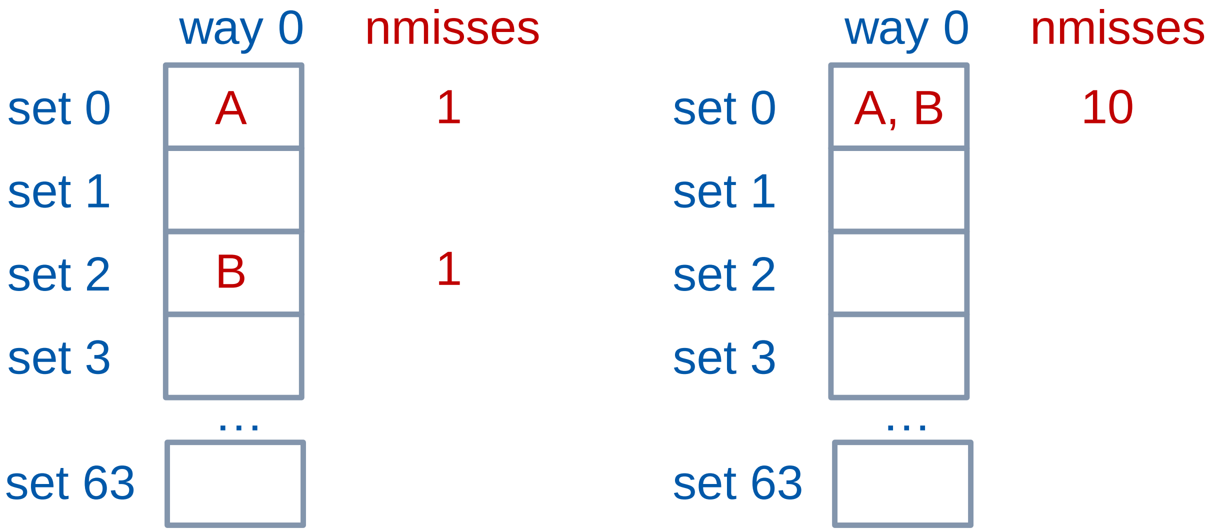

R' = ?

ccp = ?

Identifying conflictive cache placements 1/2

- J. Abella et al. "Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA" In ECRTS, 2014
  - The number of addresses mapped in a same set is the critical parameter affecting significantly execution time
  - **Conflictive cache placements are those in which more than number of ways (W) addresses are mapped in the same set**

Sequence:
ABABABABAB



	way 0	nmisses
set 0	A	1
set 1		
set 2	B	1
set 3		
	…	
set 63		

	way 0	nmisses
set 0	A, B	10
set 1		
set 2		
set 3		
	…	
set 63		

AB ccp ⇨ $Peoi = \left(\frac{1}{S}\right) \approx 0.016$ ⇨ $Pnobs(1\ run) = 1 - Peoi \approx 98\%$

$$Pnobs(R\ runs) = (1 - Peoi)^R$$

$$R' = ?:\ Pnobs(R'\ runs) \leq\ 10^{-9}$$

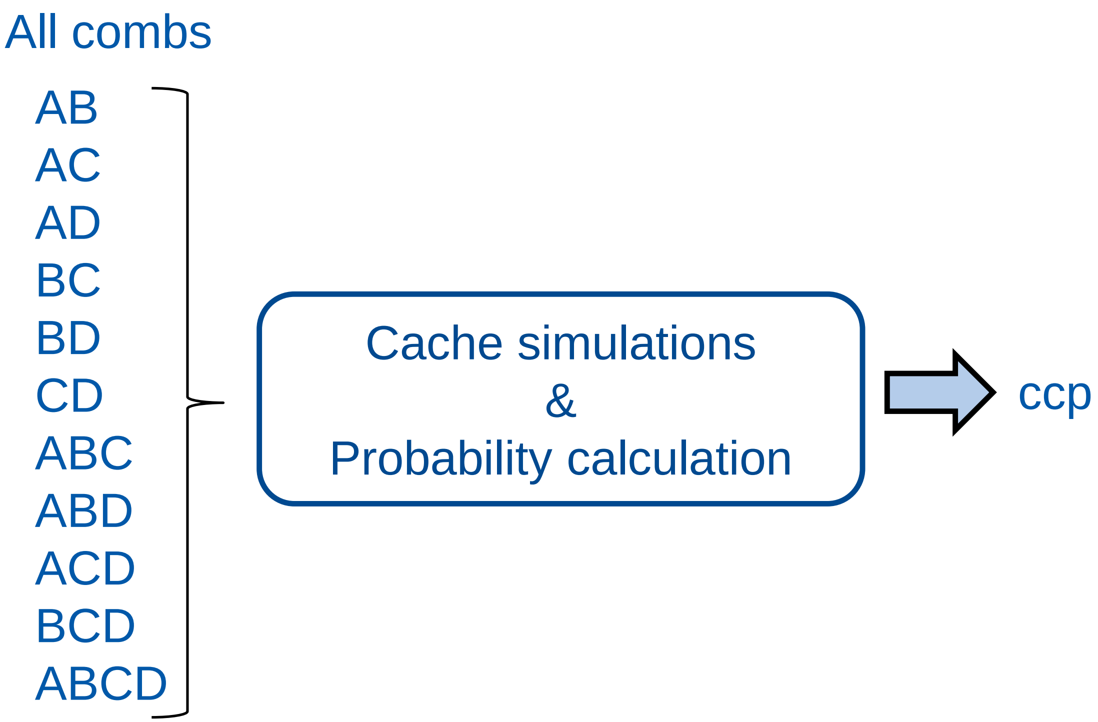
$$\boxed{R' = 1316}$$

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Identifying conflictive cache placements 2/2

Number of possible placements explodes
with increase of number of addresses in a program
→ Non-scalable solution

Sequence:
ABABABABABCD



All combs

AB
AC
AD
BC
BD
CD
ABC
ABD
ACD
BCD
ABCD

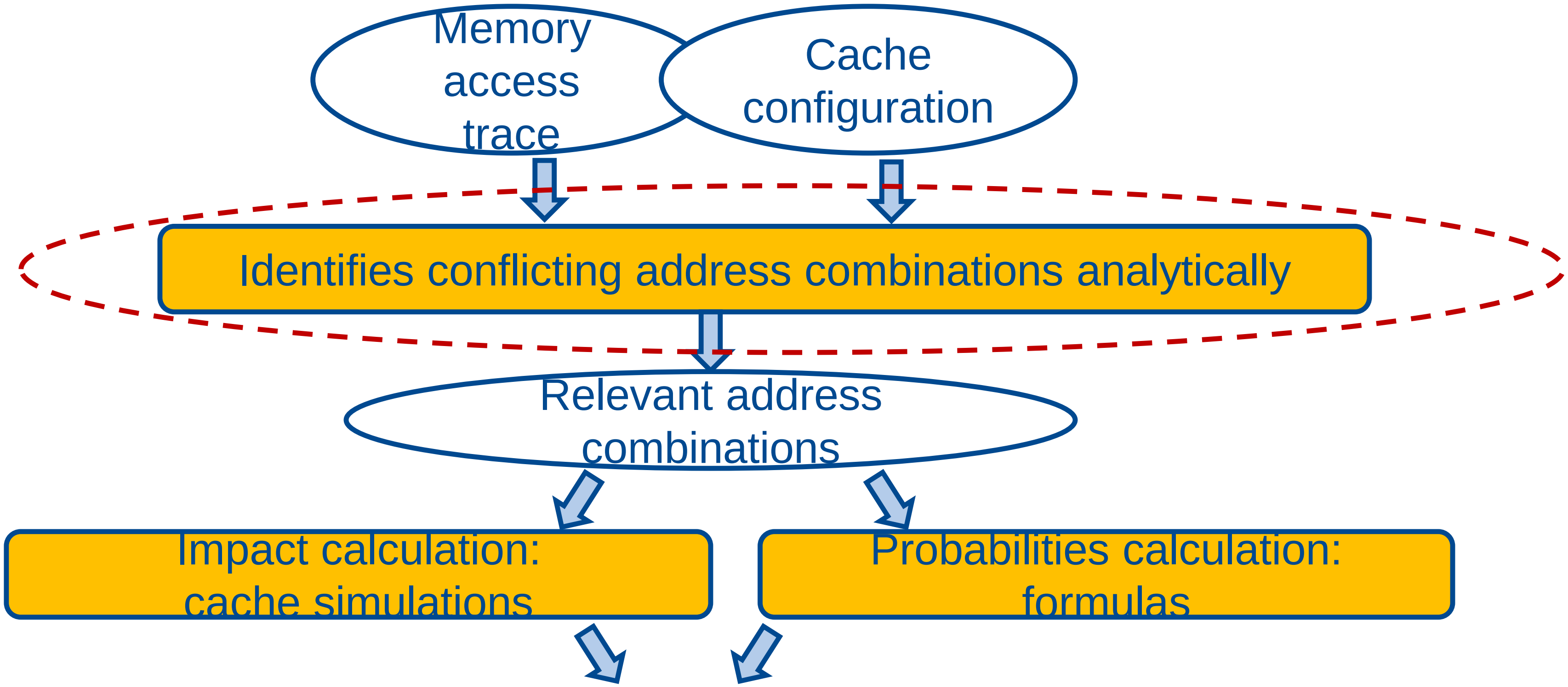
Cache simulations
&
Probability calculation

ccp

Our goal: Representativeness in presence of TR caches

- Time-aware Address Conflict (TAC) approach
  - Identifies conflictive cache placements and assesses whether they are captured at analysis time
  - If not, derives the needed number of measurements R’
  - Valid for arbitrary cache access patterns
  - Highly scalable solution

TAC method: Overview



Memory access trace

Cache configuration

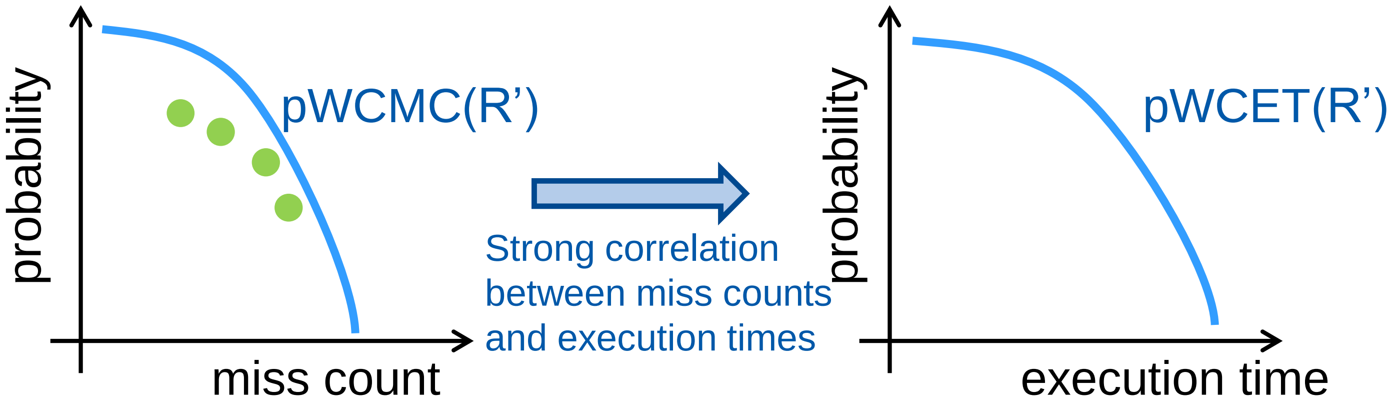
Identifies conflicting address combinations analytically

Relevant address combinations

Impact calculation: cache simulations

Probabilities calculation: formulas

- <miss count, probability> pairs : describe the combined impact of multiple combinations



Strong correlation between miss counts and execution times

Seeking conflictive address combinations: Mutual impact of addresses

- Memory access trace:

A, B, C, D, A, B, C, E, D, A

X_i

Notation:
W: number of cache ways
X_i: number of mem. accesses between two accesses to A

- Probability of access miss:

$$Pmiss = 1 - \left(\frac{W-1}{W}\right)^{\sum Pmiss(X_i)}$$

ABC?
ACD?
ADE?

- $\tilde{P}guilty$ estimator:

$$\tilde{P}guilty = 1 - \left(\frac{W-1}{W}\right)^{\exp} \qquad \exp = \begin{cases} 0 & : \quad q < W \\ q & : \quad W \le q \le K \\ K-1 & : \quad otherwise \end{cases}$$

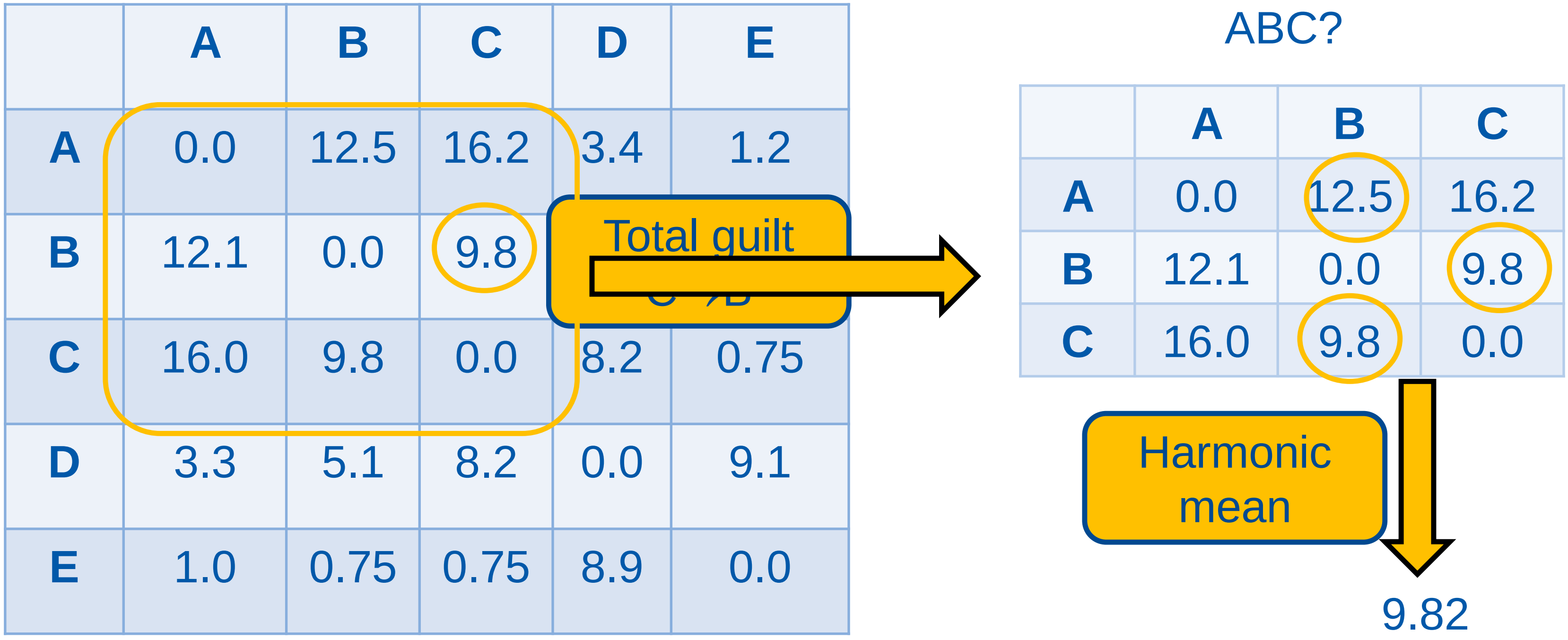
- *guilt* estimator:

$$guilt = \begin{cases} \dfrac{\tilde{P}guilty}{\exp} & : \quad \exp > 0 \\ 0 & : \quad otherwise \end{cases}$$

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Seeking conflictive address combinations: Address combination impact

- Address guilt matrix (adgm)
  - Different for each K value (e.g. 3 to 13, depending on *Prel*)



	A	B	C	D	E
A	0.0	12.5	16.2	3.4	1.2
B	12.1	0.0	9.8	[illegible]	[illegible]
C	16.0	9.8	0.0	8.2	0.75
D	3.3	5.1	8.2	0.0	9.1
E	1.0	0.75	0.75	8.9	0.0

Total guilt
C→B

ABC?

	A	B	C
A	0.0	12.5	16.2
B	12.1	0.0	9.8
C	16.0	9.8	0.0

Harmonic mean

9.82

Seeking conflictive address combinations: Smart search over adgm

	$\tilde{P}$***guilty***	**Total guilt**	**A**	**C**	**D**	**B**	**E**
A	135.75	152.0	0.0	55.5	55.5	41.0	~~0.0~~
C	124.2	132.0	55.3	0.0	46.2	30.5	~~0.0~~
D	124.2	132.0	55.3	46.2	0.0	30.5	~~0.0~~
B	56.5	71.2	10.2	30.5	30.5	0.0	~~0.0~~
~~**E**~~	~~1.0~~	~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~

	Bucket1	Bucket2
	C,D	**B**
A	55.5	41.0

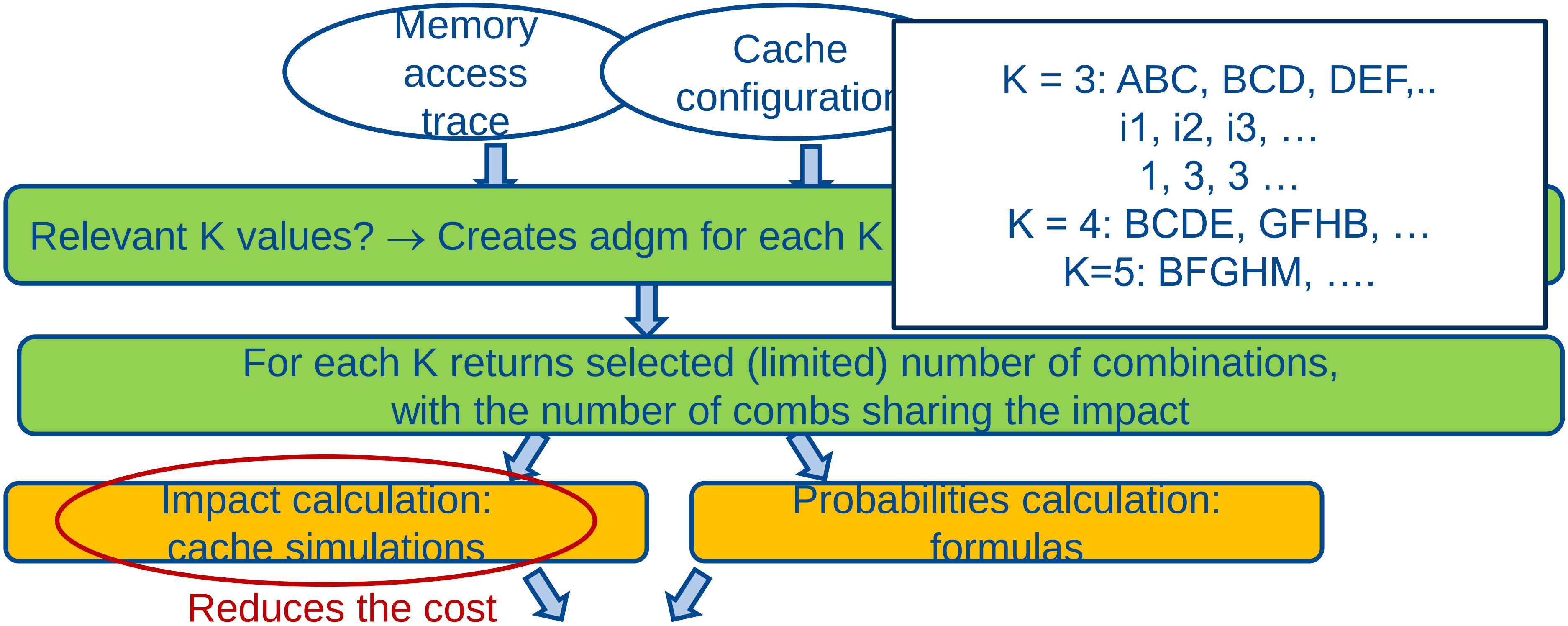
Is $\tilde{P}$*guilty* < 1% of the highest?

Is *guilt*→X < 1% of the total guilt?

Do addresses share guilt value?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

TAC method: Putting everything together



Memory access trace

Cache configuration

K = 3: ABC, BCD, DEF,..
i1, i2, i3, …
1, 3, 3 …
K = 4: BCDE, GFHB, …
K=5: BFGHM, ….

Relevant K values? → Creates adgm for each K

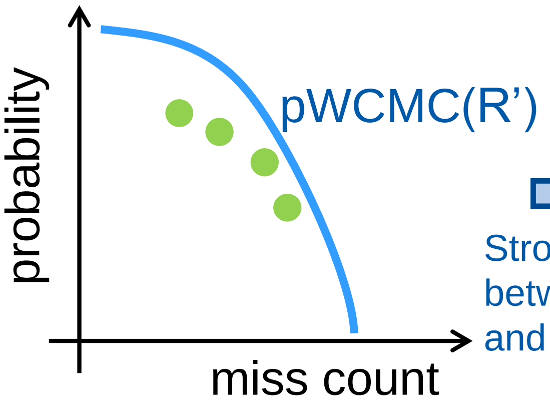
For each K returns selected (limited) number of combinations, with the number of combs sharing the impact

Impact calculation: cache simulations

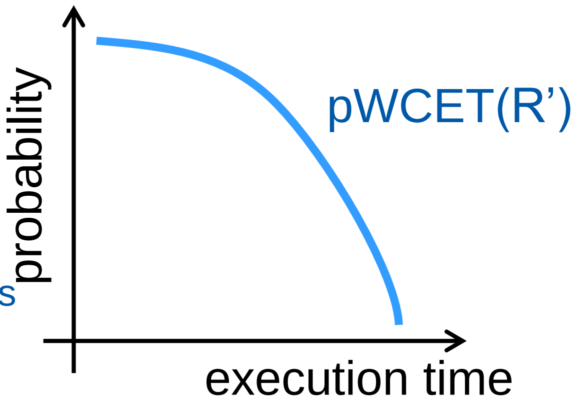
Probabilities calculation: formulas

Reduces the cost

- <miss count, probability> pairs : describe the combined impact of multiple combinations



Strong correlation between miss counts and execution times



Evaluation

- Benchmarks
  - EEMBC Autobench benchmark suite
  - Railway Case Study
- EEMBC Autobench experimental conditions
  - Cache setup
    - 4KB 64-set 2-way-associative (separated) data and instruction L1 caches
    - Cache line size 32B
    - Random placement and random replacement policies
  - Latencies
    - IL1/DL1 access latency: 1 cycle for hits, 4 cycles for misses
    - Main memory latency: 16 cycles
    - Fixed latency for non-memory operations
- Railway Case Study experimental conditions
  - LEON3-based FPGA board
  - Cache setup
    - 16KB 128-set 4-way-associative, 32B-line instruction L1 cache
    - 16KB 256-set 4-way-associative, 16B-line data L1 cache
    - Random placement and random replacement policies
- Default number of R used by MBPTA is 300

Controlled scenario: Evaluating TAC precision

- ReVS exhaustively explores all cache placements
  - **Guarantees exact results**
  - Only possible to apply to simple benchmarks
  - U = 15 most accessed addresses from EEMBC Automotive Bench

	ReVS IL1	TAC IL1
a2time	58,360	58,360
aifftr	6,840	6,840
aifirf	21,390	21,390
aiifft	8,920	8,920
basefp	82,080	82,080
bitmnp	4,640	4,640
cacheb	18,610	18,610
idctrn	65,770	65,770
iirflt	18,310	18,310

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

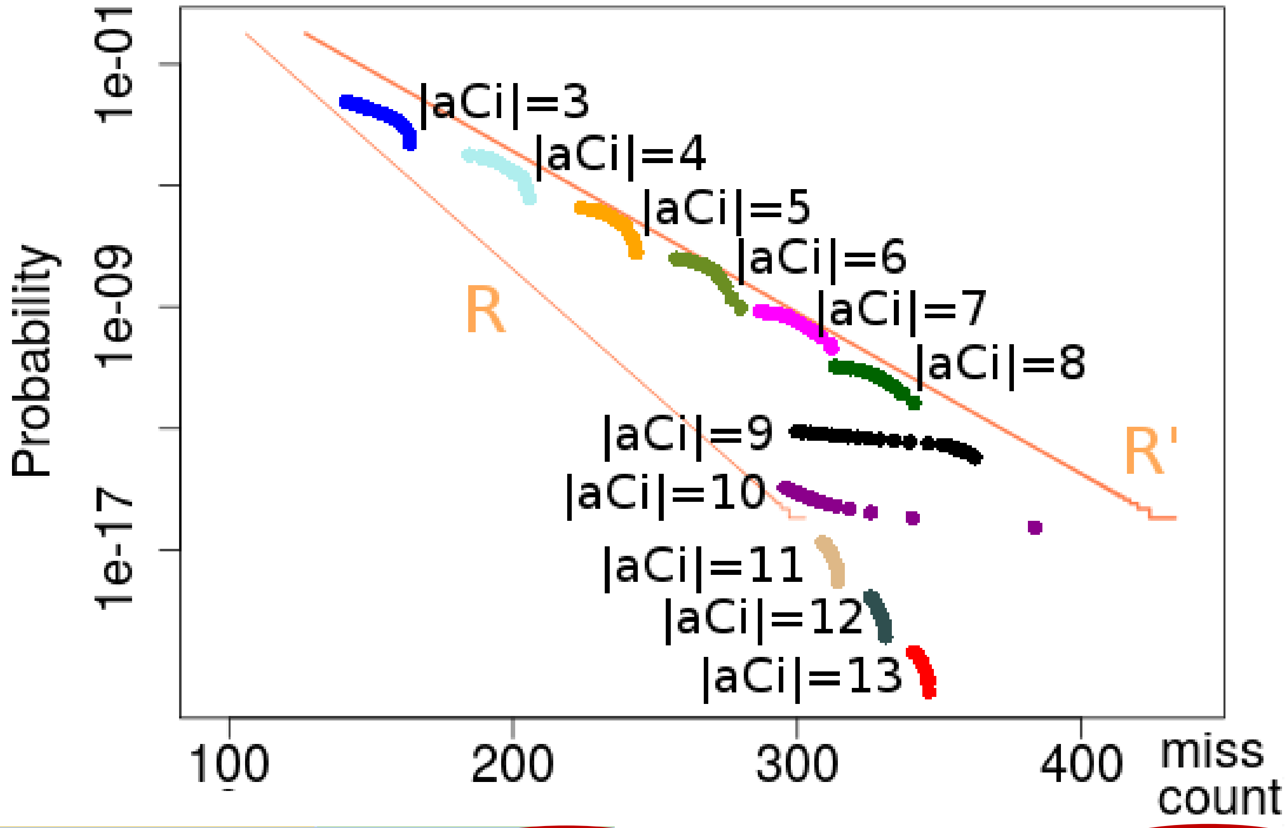
TAC evaluation on full EEMBC Autobench benchmarks

TAC

	R’ (IL1)	R’(DL1)	R’	likelihood(R’)
a2time	67,150	300	67,150	10^{-9}
aifftr	300	4,760	4,760	10^{-9}
aifirf	20,080	8,090	20,080	10^{-9}
aiifft	300	10,630	10,630	10^{-9}
basefp	78,220	300	78,220	10^{-9}
bitmnp	330	1,800	1,800	10^{-9}
cacheb	19,840	1,500	19,840	10^{-9}
idctrn	67,460	43,040	67,460	10^{-9}
iirflt	29,920	2,430	29.920	10^{-9}

Railway Case Study

- A safety function part of the European Vital Computer (EVC)
  - Travelling spe
  - The highest in



Different input sets

TEST0		
TEST1		
TEST2		
TEST3		
TEST4		
TEST5		
TEST6		
TEST7	300	300
TEST8	300	300
TEST9	300	1,740

Conclusions

- Assuring measurements observations representativeness

MBTA	MBPTA + ReVS	MBPTA + TAC
Qualitative assessment of coverage of relevant platform events impacting WCET (dependant on user expertise)	Quantitative assessment of coverage of relevant events	Quantitative assessment of coverage of relevant events
	User instructed to collect needed number of measurements R’	User instructed to collect needed number of measurements R’
	Only for very simple benchmarks	Scalable to real program sizes

- Future work
  - Provide solution when user lacks the control over input vectors
  - Generalize TAC for more complex cache hierarchies

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Software Time Reliability in the Presence of Cache Memories

Suzana Milutinovic, Jaume Abella, Irune Agirre, Mikel Azkarate-Askasua, Enrico Mezzetti, Tullio Vardanega and Francisco Cazorla

ADA-EUROPE 2017, Vienna, Austria

TAC execution time cost

- Numbers reported on average per benchmark, running 100 jobs in parallel (cache simulations are highly parallelizable)
- Controlled scenario (U = 15 for IL1/DL1)
  - ReVS: 27 hours for simulations
  - TAC: 2s to derive combs + 11 minutes for simulations (148x faster)
- Full EEMBC Autobench (U = 2,500 for IL1, U = 5,600 for DL1)
  - ReVS: unable to finish
  - TAC: 1min to derive combs + 38 minutes for simulations
- Railway case study (U = 2,994 for IL1, U = 597 for DL1)
  - ReVS: unable to finish
  - TAC: 1.3min to derive combs + 0.35 minutes for simulations

Correlating Execution Time and Miss Counts

- We compare normalized execution times and miss counts over 1000 runs

$$NormMiss_i = \frac{Miss_i - \left(MIN_{j=0}^{R} Miss_j\right)}{\left(MAX_{j=0}^{R} Miss_j\right) - \left(MIN_{j=0}^{R} Miss_j\right)}$$

- Pearson product-moment correlation coefficient and Spearman’s rank correlation coefficient

	a2time	aifftr	aifirf	aiifft	basefp	bitmnp	canrdr	idctrn
Pearson	0.997	0.918	0.960	0.923	0.999	0.998	0.974	0.950
Spearman	0.933	0.911	0.956	0.913	0.998	0.998	0.973	0.951

Seeking conflictive address combinations: Mutual impact of addresses [Corner case]

- Memory access trace:

A, B, A, C, A, B, A, C

m h m h

- Probability of access miss:

$$Pmiss = 1 - \left(\frac{W-1}{W}\right)^{\sum Pmiss(X_i)}$$

ABC?
ACD?
ADE?
ABCD?

- $\tilde{P}guilty$ estimator:

$$\tilde{P}guilty = 1 - \left(\frac{W-1}{W}\right)^{\exp} \qquad \exp = \begin{cases} \underline{0 \quad : \quad q < W} \\ q \quad : \quad W \le q \le K \\ K-1 : \quad otherwise \end{cases}$$

- *guilt* estimator:

$$guilt = \begin{cases} \dfrac{\tilde{P}guilty}{\exp} : & \exp > 0 \\ 0 & : \; otherwise \end{cases}$$

Seeking conflictive address combinations: Address combination impact [Harmonic mean]

- Address guilt matrix (adgm)
  - Different for each K value (e.g. 3 to 13, depending on *Prel*)

	A	B	C	D	E
A	0.0	12.5	16.2	3.4	1.2
B	12.1	0.0	9.8	5.3	0.75
C	16.0	9.8	0.0	8.2	0.75
D	3.3	5.1	8.2	0.0	9.1
E	1.0	0.75	0.75	8.9	0.0

ABC?

	A	B	C
A	0.0	12.5	16.2
B	12.1	0.0	9.8
C	16.0	9.8	0.0

Harmonic mean

9.82

ABC: 12.5, 9.8, 9.8

F: 1 cold miss

ABC?

Arithmetic mean = 10.70
Harmonic mean = 9.82

ABCF?

Arithmetic mean = 8.02
Harmonic mean = 0.00

Seeking conflictive address combinations: Smart search over adgm [Address buckets]

	$\tilde{P}$*guilty*	**Total guilt**	**A**	**C**	**D**	**B**	**E**
A	135.75	152.0	0.0	55.5	55.5	41.0	0.0
C	124.2	132.0	55.3	0.0	46.2	30.5	0.0
D	124.2	132.0	55.3	46.2	0.0	30.5	0.0
B	56.5	71.2	10.2	30.5	30.5	0.0	0.0
E	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Bucket1 Bucket2

	C,D	**B**
A	55.5	41.0

2 0 : 1

1 1 : 2

0 2 : 0

Is $\tilde{P}$*guilty* < 1% of the highest?

Is *guilt*→X < 1% of the total guilt?

Do addresses share guilt value?