



PROXIMA

Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA

Jaume Abella¹, Eduardo Quiñones¹, Franck Wartel²,
Tullio Vardanega³, Francisco J. Cazorla^{1,4}



This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu

Context

- ❑ Critical Real-Time Embedded Systems (CRTES) need functional *and* timing correctness
 - Derivation of WCET estimates needed for critical tasks

- ❑ Several timing analysis methods respond to this need, but ***all of them*** have some sources of uncertainty
 - Uncertainty for sound static timing analysis (STA)
 - Does documentation accurately describe HW timing?
 - E.g. partial or full errata
 - Is the implementation accurate?
 - Can we trust flow facts as provided by the user?
 - Uncertainty for measurement-based timing analysis (MBTA)
 - Do the test vectors really engage the worst-case conditions?
 - Is the data (execution times) collection method trustworthy?

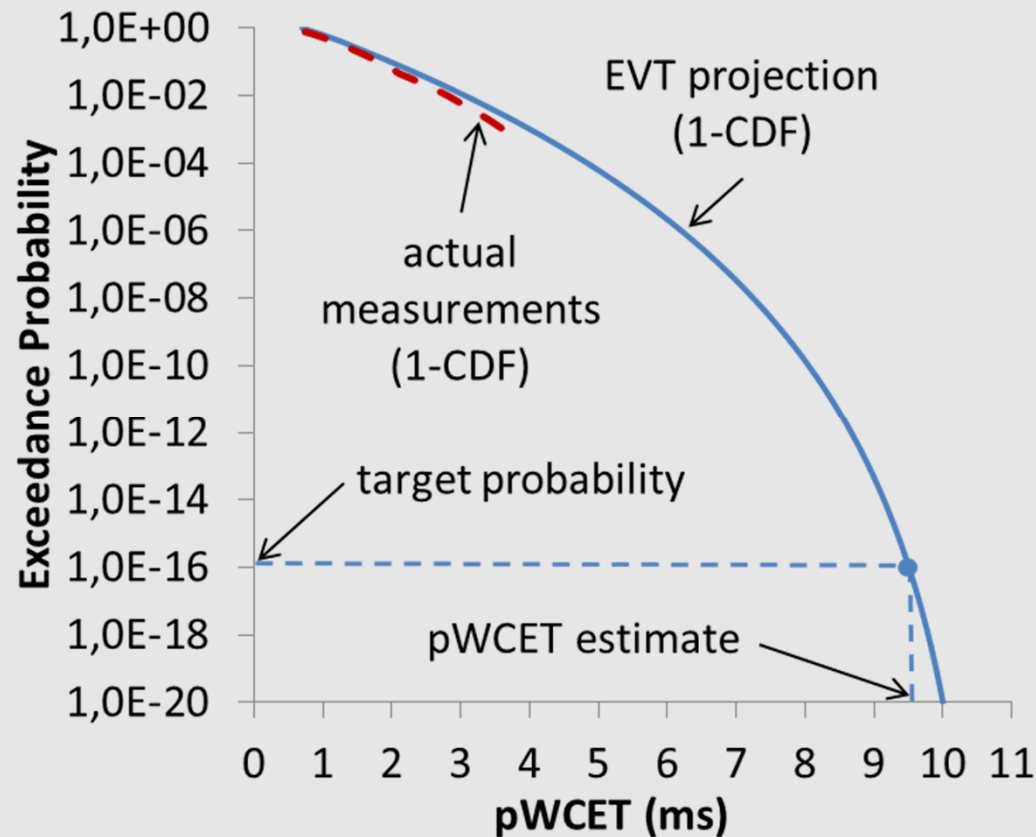
MBPTA (measurement-based probabilistic timing analysis)

- MBPTA is emerging as a viable alternative
 - Provides trustworthy upper bounds on tasks execution time
 - Industrially-friendly as it is based on measurements
 - **Reduces the burden on the user**
 - As shown before, PUB further reduces the amount of inputs from the user

- But MBPTA also has its own sources of uncertainty ...
 - They emanate from the particular way software may make use of some hardware resources
 - Still, the **probabilistic nature of the system timing** allows quantifying them and providing countermeasures

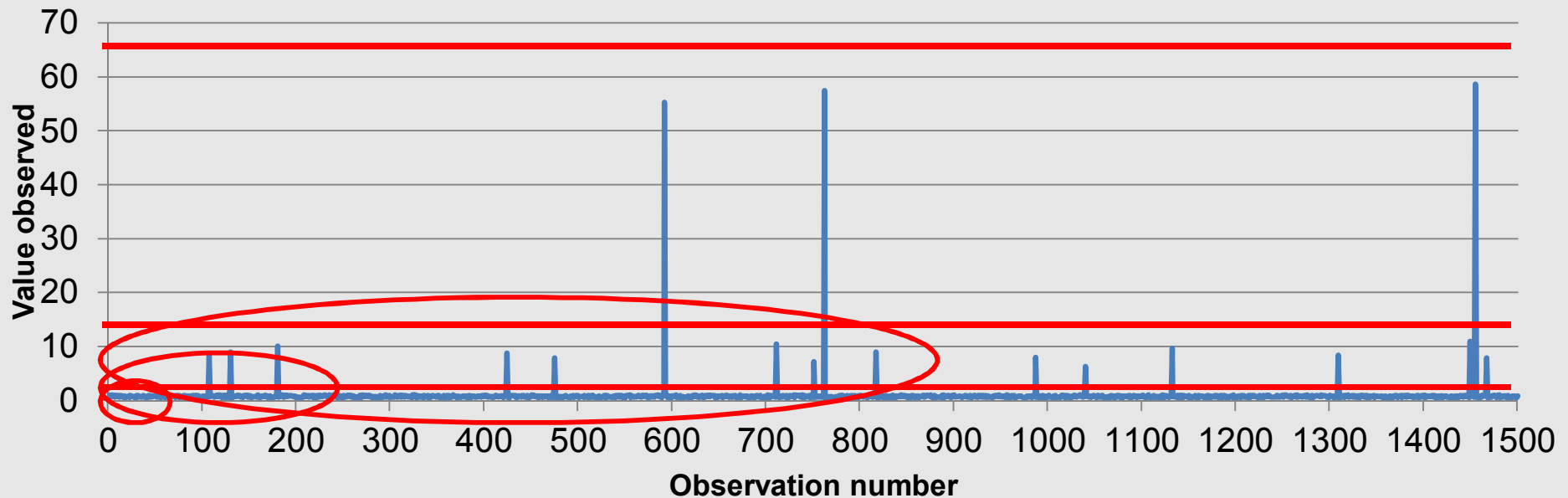
MBPTA

- MBPTA is the most powerful and **industrial-friendly** PTA technique proposed so far
 - Key challenge: EVT projection upperbounds true ET distribution



Extreme Value Theory (EVT)

- MBPTA relies on EVT to derive pWCET estimates
 - EVT provides the expected value for a given exceedance threshold
 - EVT applied to high execution time observations provides pWCET values
- What can EVT do and what cannot? An example

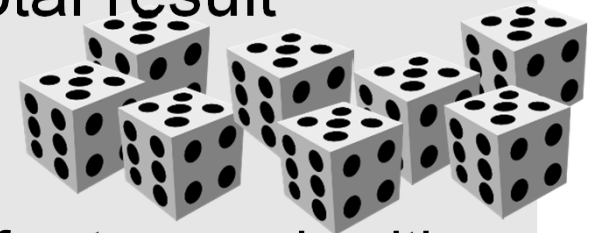


Extreme Value Theory (EVT)

- ❑ Do we need to observe the worst combination ever?
 - **NO!!! EVT predicts bad combinations**

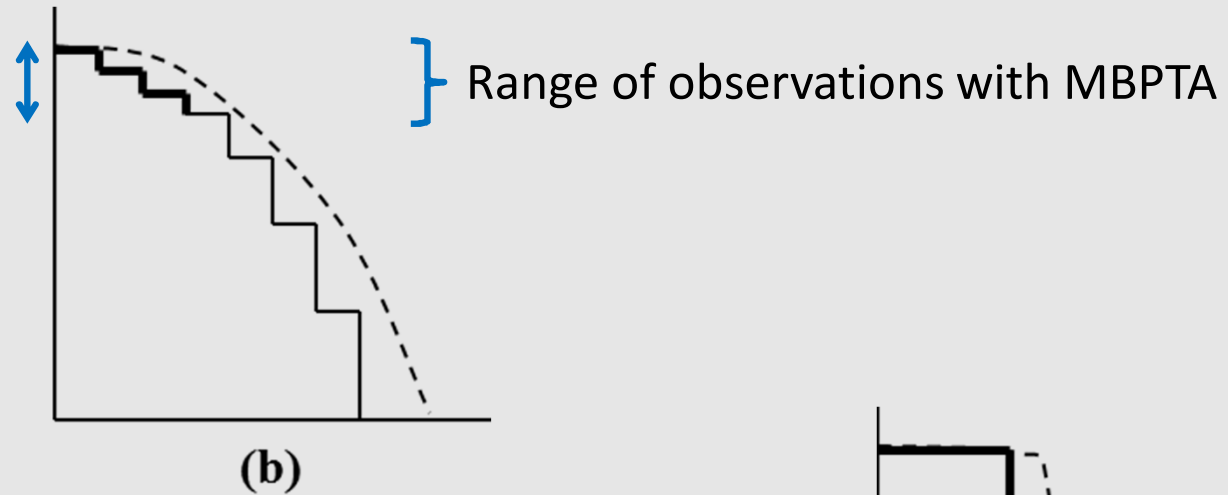
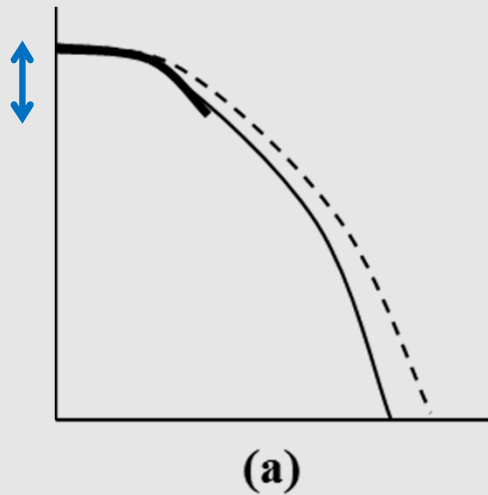
- ❑ Example: **throw 100 dices** and sum the total result
 - Sum is in the [100, 600] range
 - Most observations are in the [300, 400] range
 - EVT accurately upper bounds the probability of outcomes in either extremes (<300, >400)


- ❑ What if 1 die can give 10,000 with probability 0.0000001?
 - Sum now evaluates in the [100, **10,600**) range
 - Most observations continue to fall in the [300, 400] range
 - With no 10,000 value being observed
 - **EVT unlikely to upper bound the probability of values >600**

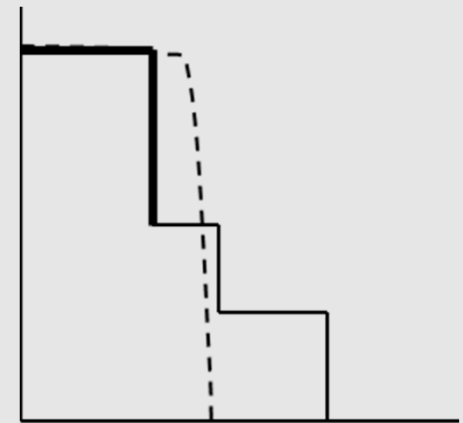


Execution Times and EVT

- Execution times are discrete
 - Smallest time scale: 1 cycle
 - Furthermore, some particular execution times cannot occur
 - Thus, CCDF looks like a step function



Can this  happen? YES, but...
1) Here we identify why and when,
2) And provide means to detect it

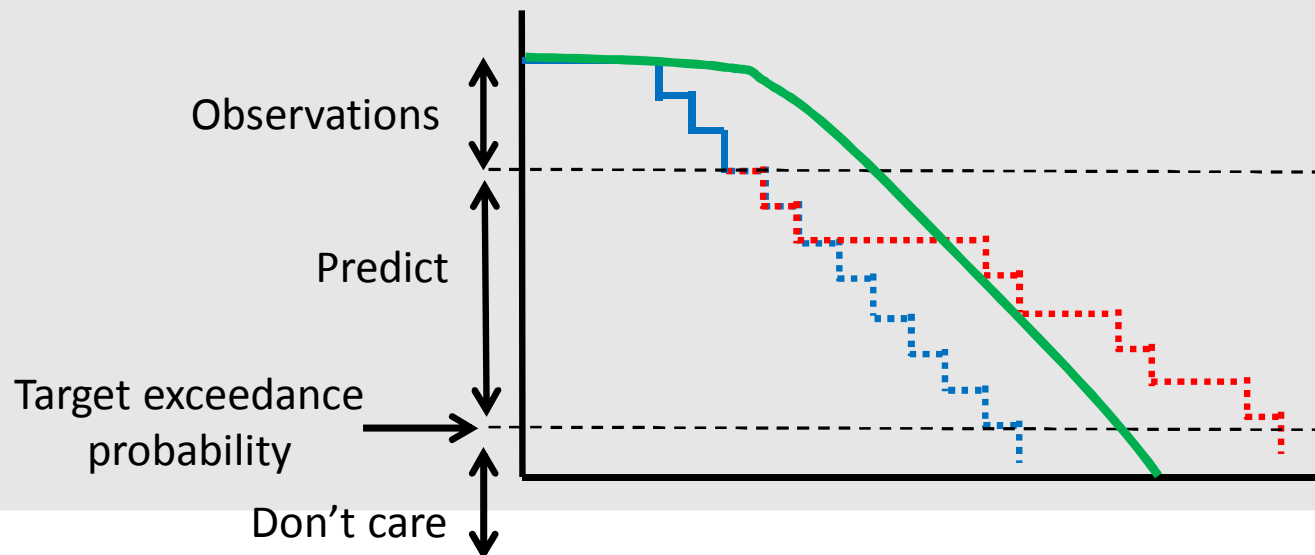
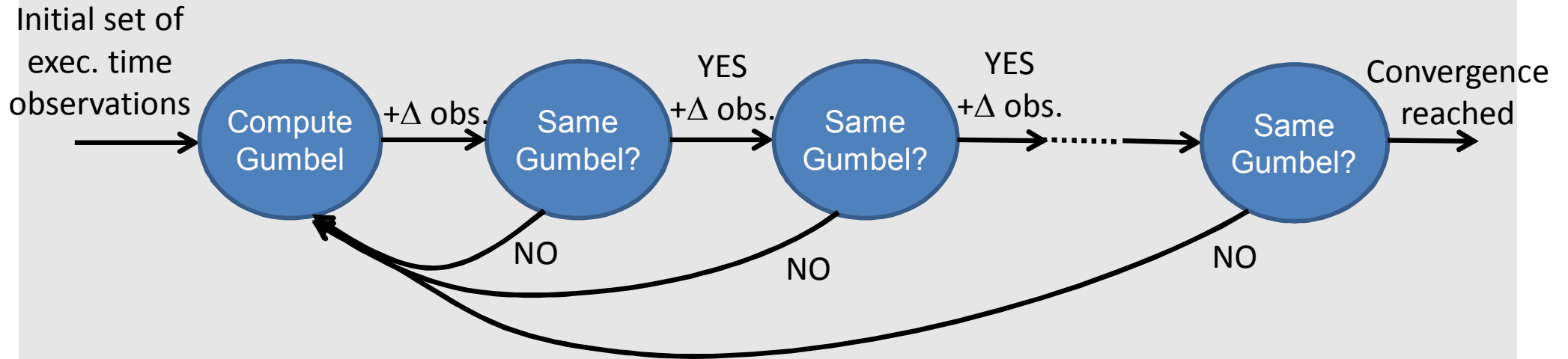


Outline of the rest of this talk

- ❑ Convergence in MBPTA
- ❑ Random events in MBPTA-friendly processors
 - Types
 - Challenging SW/HW interactions
- ❑ Heart-of-Gold (HoG)
 - Rationale
 - Technique
- ❑ Evaluation
- ❑ Conclusions

Convergence in MBPTA

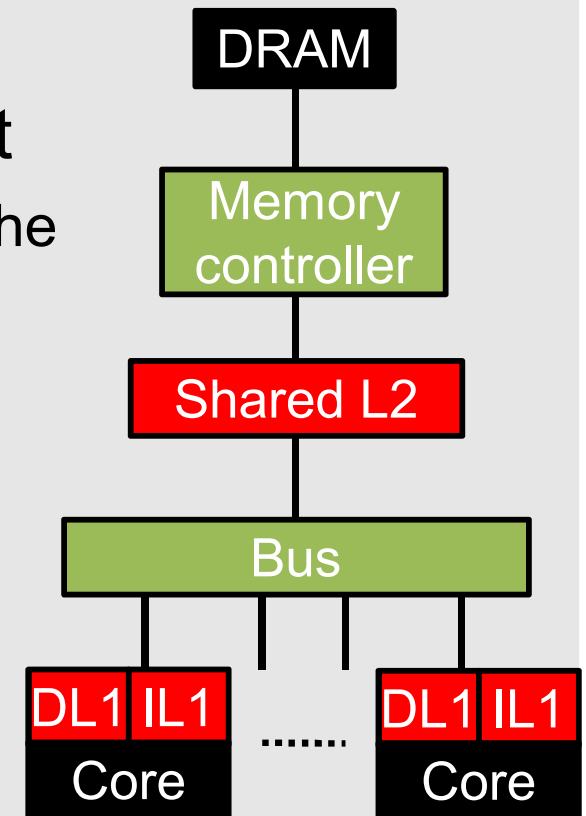
- Iteratively evaluate if the EVT projection is stable



Random Events

- Some of them strictly hardware dependent
 - We can study the shape of the distributions they can produce
 - Existing ones (random bus arbitration) **cannot create risky distributions**

- Others are hardware-software dependent
 - How SW uses HW determines the shape of the distribution
 - Thus, effects different for each program
 - Basically, **random-placement** and random-replacement caches
 - Our analysis shows that risky distributions typically happen under weird cache setups
 - Let us look at an example...

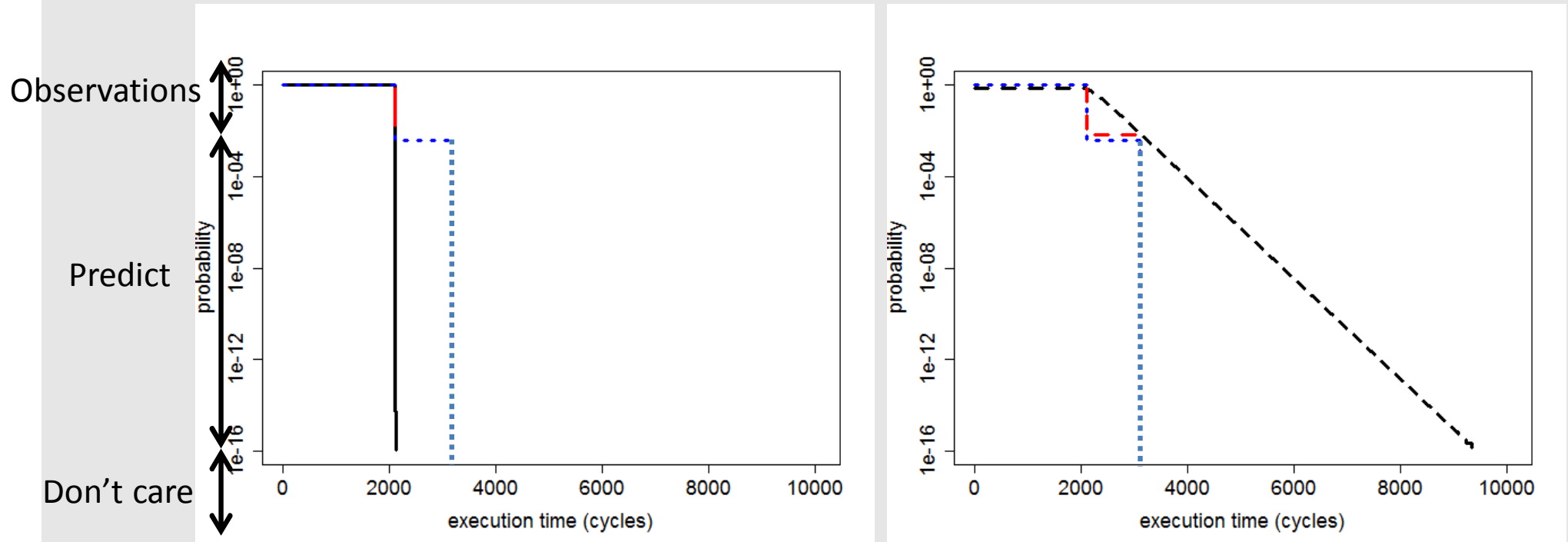


Corner Case

- ❑ Random placement cache, direct-mapped, 1024 sets
 - So, 1 line per set
- ❑ Randomly arbitrated bus across cores to access memory
- ❑ Thorny (corner-case) program
 - Loop (from 1 to N)
 - Some iterations, accessing only A and B
 - Conflict unlikely to occur (1 out 1024 runs) creates a large number of misses (2 x N misses)
 - Overall, runs behave as follows
 - 1023 out of 1024 have 2 misses and nearly-constant execution time (random variation due to random bus)
 - 1 out of 1024 has a huge number of misses

Corner Case

- Corner program
 - **Red**: observations
 - **Black**: EVT projection
 - **Blue**: real distribution



Heart-of-Gold (HoG)

□ Rationale

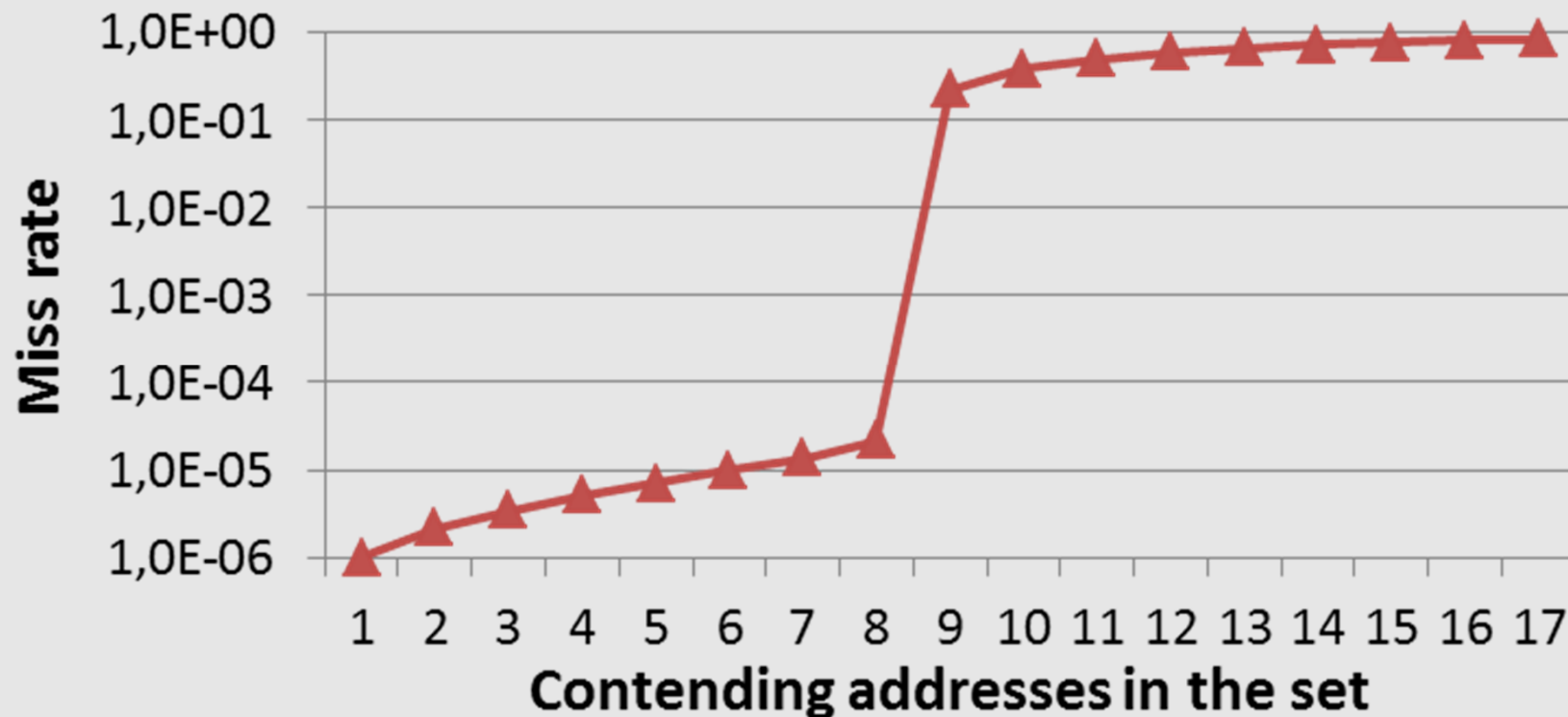
- Some rare events can occur with a probability in the conflictive probability range (e.g., between 10^{-3} and 10^{-16})
- Such **probability depends on the number of cache sets**
- Therefore, decreasing number of sets increases probability of rare events (“tsunamis”)
- HoG
 - **Halve cache size** until weird events cannot exist
 - If they exist, they will be observed (*)
 - Alternatively, **increase number of observations**: same effect

(*) They won't be observed with a probability lower than the probability of failure defined in safety standards

- Thus, it can be neglected from a certification perspective

HoG: Number of Addresses per Set

- How many addresses need to compete for the same set to create the “big step” in the exceedance function?
 - Big step occurs when more than W addresses in the set
 - E.g., $W=8$, big step at 9



HoG: Determining Conflictive Scenarios

- ❑ Enumerate **all possible cache address placements**
 - Approximated with the weak combinations theory
 - Omitted in the presentation
 - Enumerate them unrestrictedly

- ❑ Enumerate all possible cache address placements such that **at least $W+1$ cache lines collide** in one set
 - Also approximated with the weak combinations theory

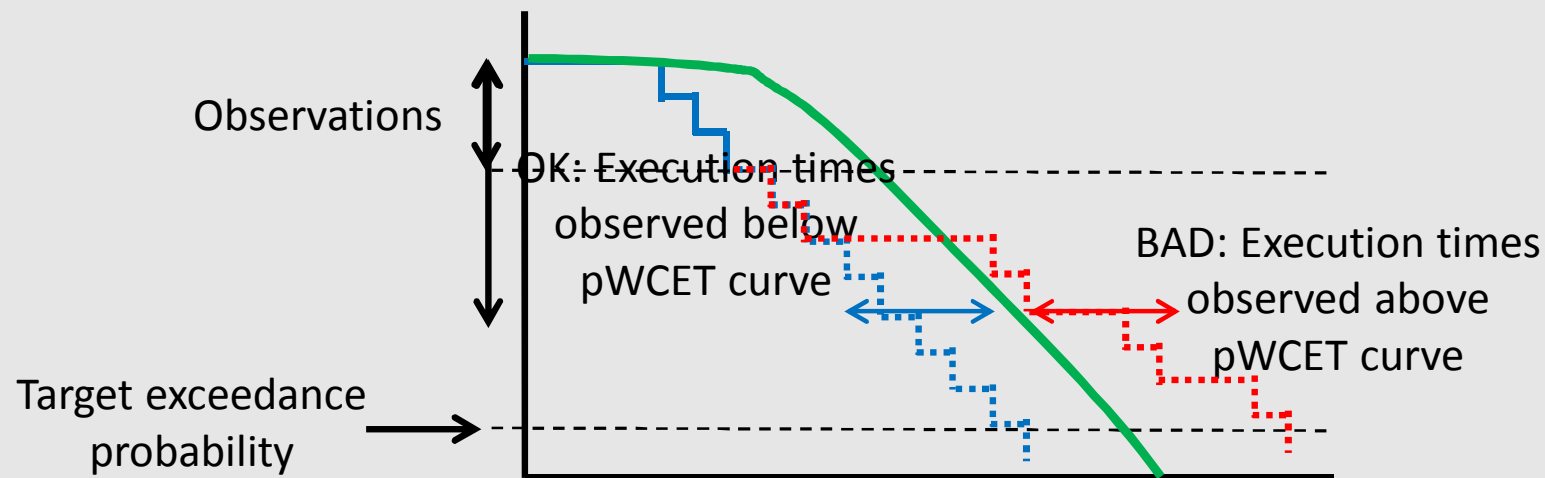
- ❑ Ratio: $P_{\text{extreme}}(U, W, S)$
 - Gives the probability of a given run to expose a conflictive scenario
 - U is number of cache lines, W associativity, S number of sets

HoG: Using P_{extreme}

- Probability of not observing conflictive scenario in one run
 - $1 - P_{\text{extreme}}$
- Probability of not observing it in R runs
 - $P_{\text{not seen}} = (1 - P_{\text{extreme}})^R$
- $P_{\text{not seen}}$
 - Lower than acceptable failure rate (e.g., 10^{-9}), we are done
 - Risk is negligible
 - Otherwise
 - Halve cache
- In practice, R is no less than 300, so $P_{\text{extreme}} > 0.07$ guarantees no risk
 - Typically, if addresses occupy $>15\%$ of the cache space, then $P_{\text{extreme}} > 0.07$

HoG: Halving the Number of Cache Sets

- ❑ S is decreased until $P_{\text{not seen}}$ below acceptable failure rate
 - Now execution times include conflictive scenarios
 - By decreasing cache size we observe lower probability events
 - Execution times must be below the pWCET curve



- ❑ The same effect is obtained by increasing the number of runs (R grows, $P_{\text{not seen}}$ decreases)

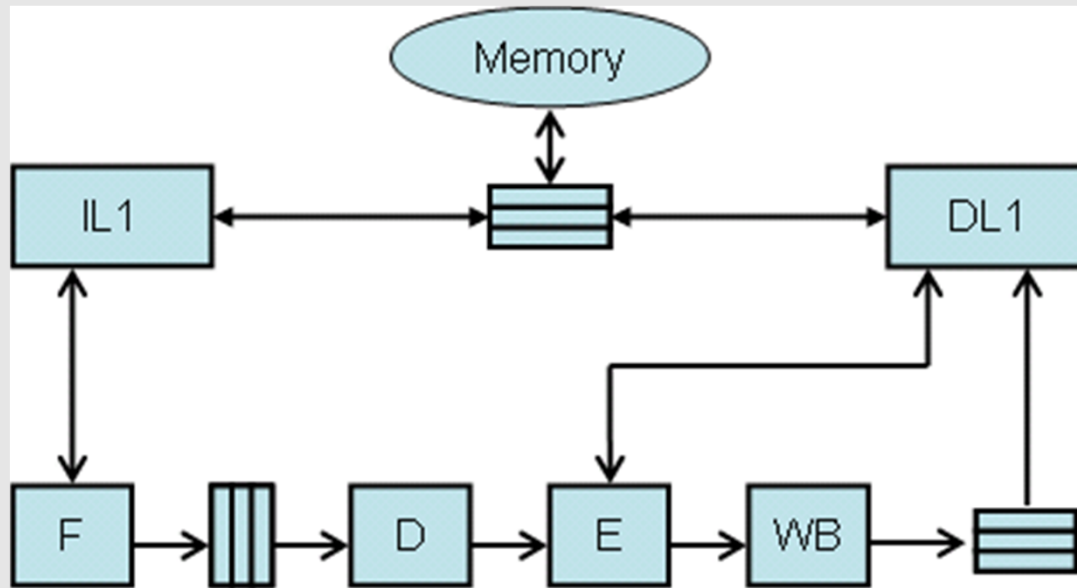
HoG: Determining U (cache lines that matter)

- How many cache lines need to be considered?
 - Open challenge: it depends on the structure of the program
 - Now working on this

- Programs considered so far
 - EEMBC Autobench: representative of the automotive domain
 - Main loop traversing all data a number of times
 - Thus, all cache lines accessed matter and have similar impact in terms of misses

Evaluation

- ❑ Simulator resembling embedded processor
 - IL1 and DL1 random placement and random replacement
 - EEMBC: 8KB 8-way 16B/line IL1 and DL1
 - Corner program: 32KB 1-way 16B/line DL1, perfect IL1



EEMBC Results

- P_{extreme} must be above 0.07 to guarantee that conflictive scenarios have been observed

Bench.	DL1		IL1	
	U	$P_{\text{extreme}}(U, S, W)$	U	$P_{\text{extreme}}(U, S, W)$
a2time	739	1.000	964	1.000
aifftr	11,706	1.000	1,436	1.000
aifirf	852	1.000	905	1.000
aiifft	11,702	1.000	1,240	1.000
cacheb	587	1.000	1,023	1.000
canrdr	416	1.000	2,166	1.000
iirflt	806	1.000	1,611	1.000
puwmod	167	0.993	791	1.000
rspeed	102	0.566	372	1.000
tblock	1,381	1.000	647	1.000
ttsprk	460	1.000	495	1.000

Thorny Program (corner-case) Results

- P_{extreme} must be above 0.07 to guarantee that conflictive scenarios have been observed
 - pWCET is 189,374

Parameter	Value		
	1	2	4
F			
$P_{\text{extreme}}(U, \frac{S}{F}, W)$	0.00098	0.00195	0.00390
μ	186,168	192,430	200,509

- What can we do?
 - Keep increasing input data for MBPTA until HoG regards the pWCET as trustable
 - Reason: MBPTA converged too early

P_{extreme} not yet reached,
but pWCET exceeded!!!

Indicates that pWCET
cannot be used

Conclusions

- ❑ All timing analyses have sources of uncertainty
 - In most cases **uncertainty cannot be quantified**
- ❑ MBPTA makes almost everything probabilistic
 - Therefore, **uncertainty is probabilistic and can be quantified**
- ❑ HoG relies on the probabilistic nature of the system to
 - **Identify** whether some risk exists
 - If exists, **decrease cache size** until conflictive scenarios are proven to be observed (or increase number of runs)
 - **Determine** whether pWCET estimates can be trusted
- ❑ Future work
 - Generalise the determination of the number of cache lines accessed to consider in the analysis



PROXIMA

Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA

Jaume Abella¹, Eduardo Quiñones¹, Franck Wartel²,
Tullio Vardanega³, Francisco J. Cazorla^{1,4}



This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu