Enhancing Speculative Execution with Selective Approximate Computing

Bernard Nongpoh[1], Moumita Das[2], Rajarshi Ray [1], Ansuman Banerjee[2]

[1] Department of Computer Science & Engineering, National Institute of Technology Meghalaya, Shillong, India.

[2] Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India.

Abstract

Speculative execution is an optimization technique in modern processors. ***Branch prediction*** and ***load value speculation*** are examples of speculative execution used in modern pipelined processors to avoid an execution stall. However, speculative executions incur a performance penalty as an execution roll-back, when there is a misprediction. In this work, we propose to aid speculative execution with approximate computing by relaxing the penalty associated with a misprediction. We propose a sensitivity analysis of load and branch instructions in order to identify the ones which can execute without any execution roll-back in the pipeline and yet can assert a certain user specified quality of service of the application with a probabilistic guarantee.

Problem Statement

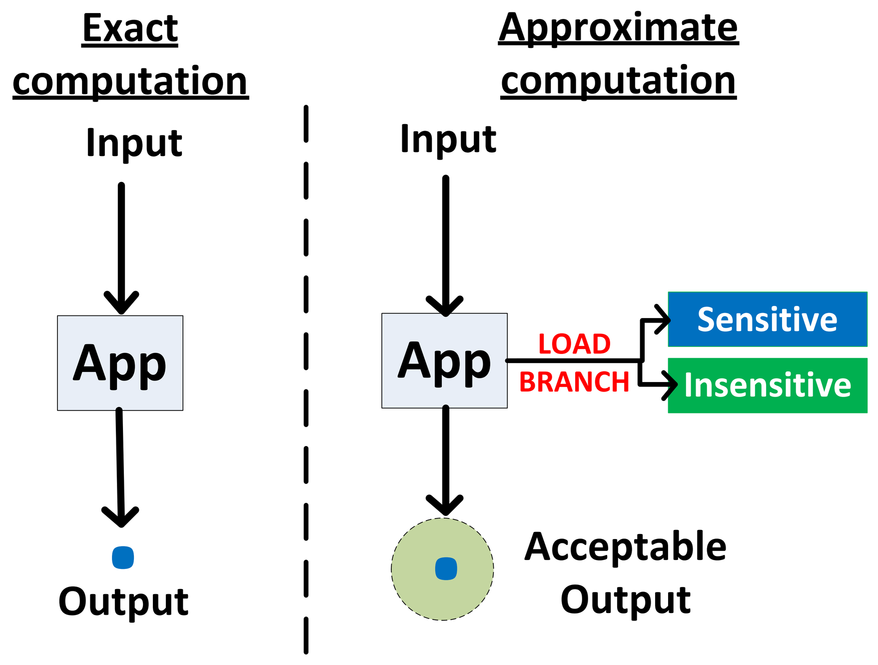


Figure 1: Instruction classification in approximate computing

Sensitivity Analysis: A Motivating Example

```
void SOR_execute(...){
  ...
//@cache-miss-heavy
Gip1 = G[i+1];
//@end
for(j=1;j<Nm1;j++){
//@cache-miss-heavy
 Gi[j] = omega_over_four*
 (Gim1[j]+Gip1[j]+Gi[j-1]
 +Gi[j+1] + one_minus_omega
  * Gi[j]);
   //@end
  }
  ...
}
```

Inst. Addr.	Assembly Code	D1 Misses
402670	movsd (%rax), %xmm0	249999
402609	mov (%rax), %rax	250
402658	movsd (%rax), %xmm1	250
4026ac	movsd (%rax), %xmm1	250
402651	mov -0x10(%rbp), %rax	244
402678	mov -0x34(%rbp), %eax	244
402685	mov -0x18(%rbp), %rax	243

D1 Cache Miss Rate : 2.7%

- **22** LOAD Instructions
- **7** LOAD data into FP registers
- **15** LOAD data into INT registers
- **3** Approximable LOADs with Confidence of at least 0.5

Sensitive / Insensitive

Contributions

A systematic method for **instruction** classification with quantitative confidence guarantee. The contributions are:

- A Dynamic analysis to automatically classify Instruction as sensitive or insensitive.
- Experimental results to demonstrate the gain in selective approximation.
- Propose a roll-back free execution for load/branch mis-prediction.

Definition : Sensitive Instruction

Approximable Load: Given a confidence of inference θ and an application's QoS distortion limit α, **a load instruction** $\mathcal{I}$ is *approximable* if and only if it is asserted with a probability at least θ that an execution with an in-exact load value into the respective load register does not distort the application output beyond the limit α.

Approximable Branch: Given a confidence of inference θ and an application QoS distortion limit α, **a branch** $\mathcal{B}$ is *approximable* if and only if it is asserted with a probability at least θ that a wrong path execution along an incorrect branch of $\mathcal{B}$ does not distort the application output beyond α.

Sensitivity Analysis Using Hypothesis Testing

For every $i \in \mathcal{I}$, we propose a hypothesis that $\forall e \in E, \forall \ell \in \ell_i^e, (i_e, \ell) \rightarrow (i_{approx}, \ell) \implies \mathcal{R} \in QoS$, where E, ℓ_i^e, (i_e, ℓ) and (i_{approx}, ℓ). Let us denote such an hypothesis by K. Test the following null and contrary hypothesis:

$$H : Pr(K) < \theta$$
$$H' : Pr(K) \geq \theta \quad (1)$$

where $Pr(K)$ is the probability that the hypothesis K is true.

Sequential Probability Ratio Test

- SPRT is to decide whether additional experiments need to be performed to accept or reject a hypothesis on the basis of the previously observed outcomes.

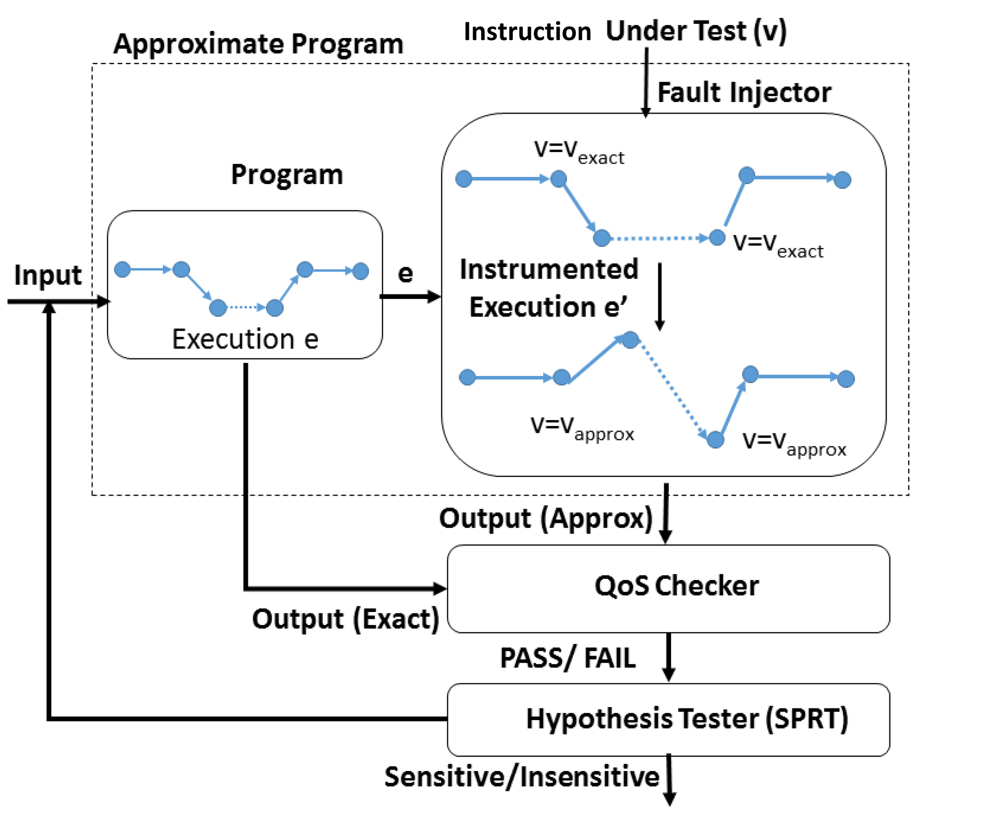


Figure 2: Framework of Dynamic Sensitivity Analysis with Hypothesis Testing

Workflow

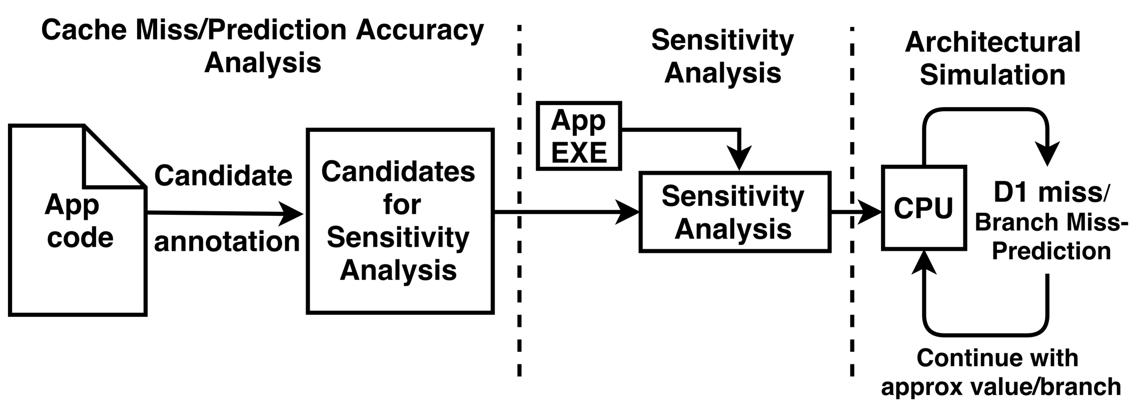
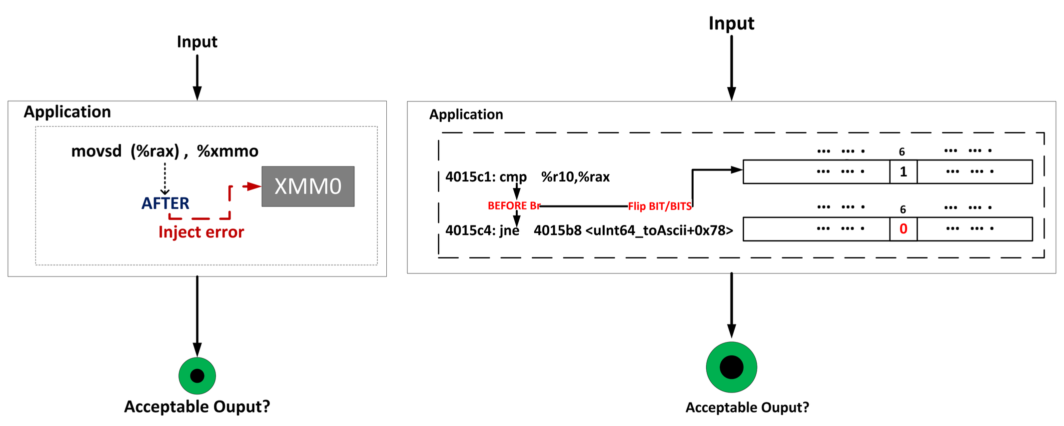


Figure 3: Schematic experimentation workflow of our proposed work

Fault injection in load/branch instruction



(a) LOAD instruction (b) BRANCH instruction

Figure 4: Fault injection using Dynamic Binary Instrumentation[1]

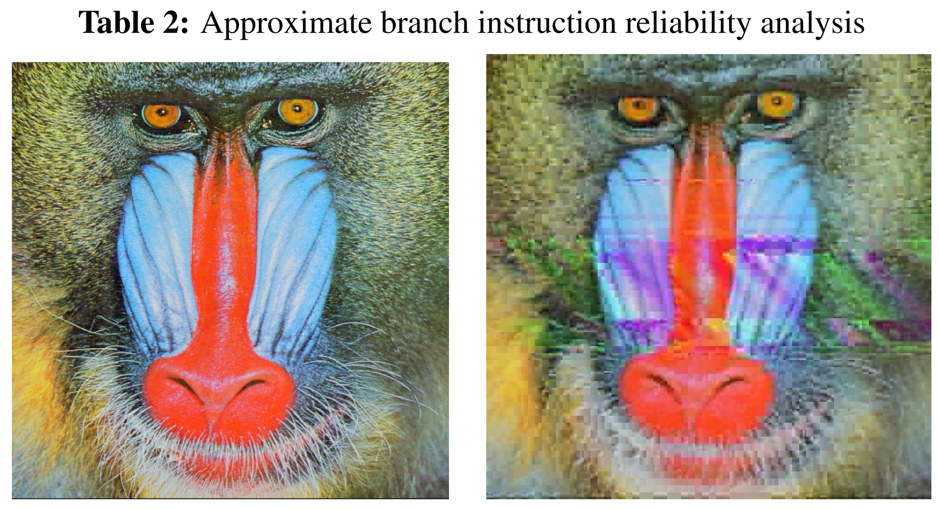
Reliability of Sensitivity Analysis

Application	Error-metric	Acc. Err.	Load Sensitivity			
			Cache miss %	Analyzed	Approx.	QoS Loss
SOR	NME	≤ 0.5	0.26	22	3	0.10
LU	NME	≤ 0.5	1.13	47	1	0.48
Soplex	% Error	≤ 0.5	2.85	1136	10	0.01
GemsFDTD	NME	≤ 0.5	9.87	110	10	0.03
JPEGEncoder	PSNR	≥ 10.5	0.52	40	3	23.38
StreamCluster	MDE	≤ 0.5	0.43	169	10	0.16
Bzip2 (v1.0.6)	PSNR	≥ 10.5	2.51	301	10	NIL

Table 1: Approximate load instruction reliability analysis

Application	Error-metric	Acc. Err.	Branch Sensitivity				
			Mispred. %	B. Intr.	Analyzed	Approx.	QoS Loss
SOR	Normalized mean error	≤ 0.5	9.47	25	8	5	0.27
LU	Normalized mean error	≤ 0.5	0.35	38	15	1	0.00
Soplex	Percent error	≤ 0.5	4.61	2489	100	5	0.06
GemsFDTD	Normalized mean error	≤ 0.5	3.71	3834	100	16	0.00
JPEGEncoder	Peak Signal to Noise Ratio	≥ 10.5	11.27	1481	100	13	19.06
StreamCluster	Mean distance error	≤ 0.5	12.20	867	100	16	0.39
Bzip2 (v1.0.6)	Peak Signal to Noise Ratio	≥ 10.5	2.31	867	100	6	NIL

Table 2: Approximate branch instruction reliability analysis



(a) Compressed image with exact computation

(b) Compressed image with Branch Approximation

Figure 5: QoS comparison with penalty free execution scheme

Architectural Simulation

System Configuration

Architecture	x86 with clock frequency of 3.4GHz, uni-cores
Type of Pipeline	In-order pipeline of width 1
Branch Predictor	Bimodal, penalty: 8 cycles
Private L1 Cache	32KB, 8-way, 64 byte blocks, 3-cycles latency
Shared L2 Cache	256KB, 8-way, 64 byte blocks, 32-cycles latency
Main Memory	A miss in L2 Cache is considered as a hit in the Main memory with a miss penalty of 200-cycles
Power Model	McPAT

Table 3: System configuration in an architectural simulator[2]

Module modified in architectural simulator[2]

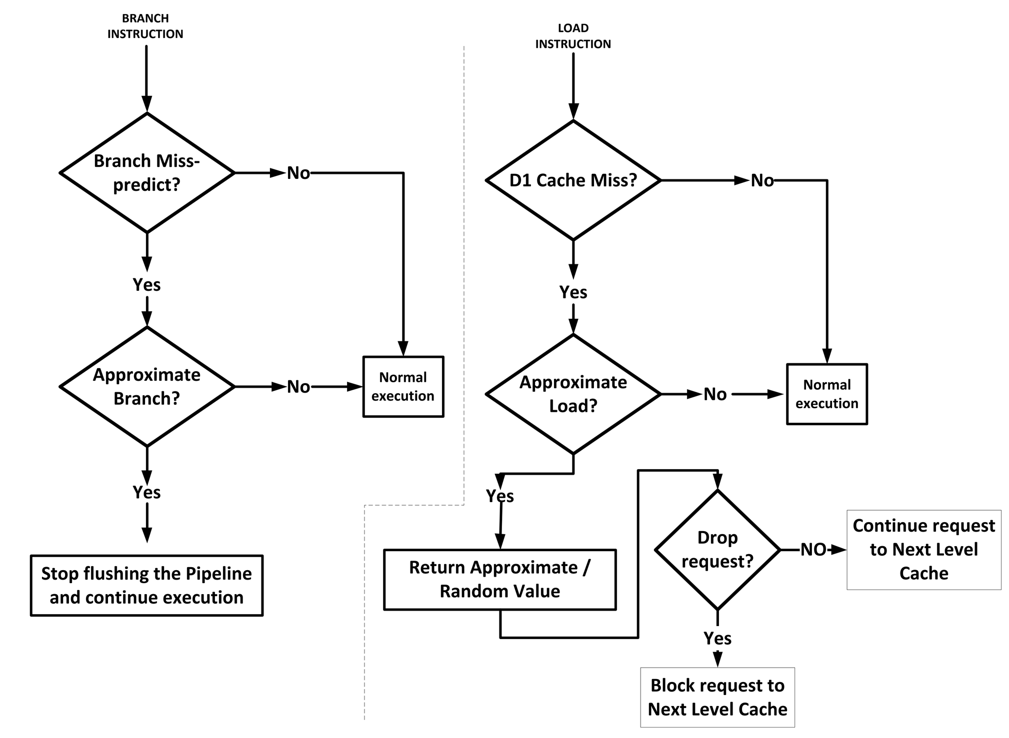


Figure 6: Roll-back free execution in approximate branch and load instruction

Evaluation of Dynamic Sensitivity Analysis

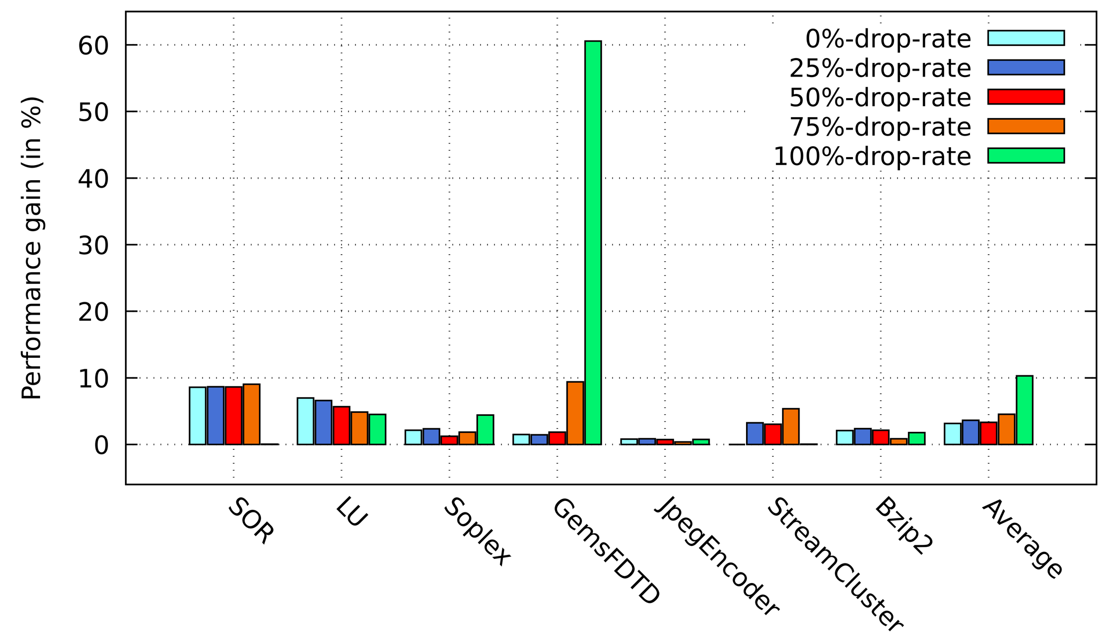


Figure 7: Performance gain in load approximation

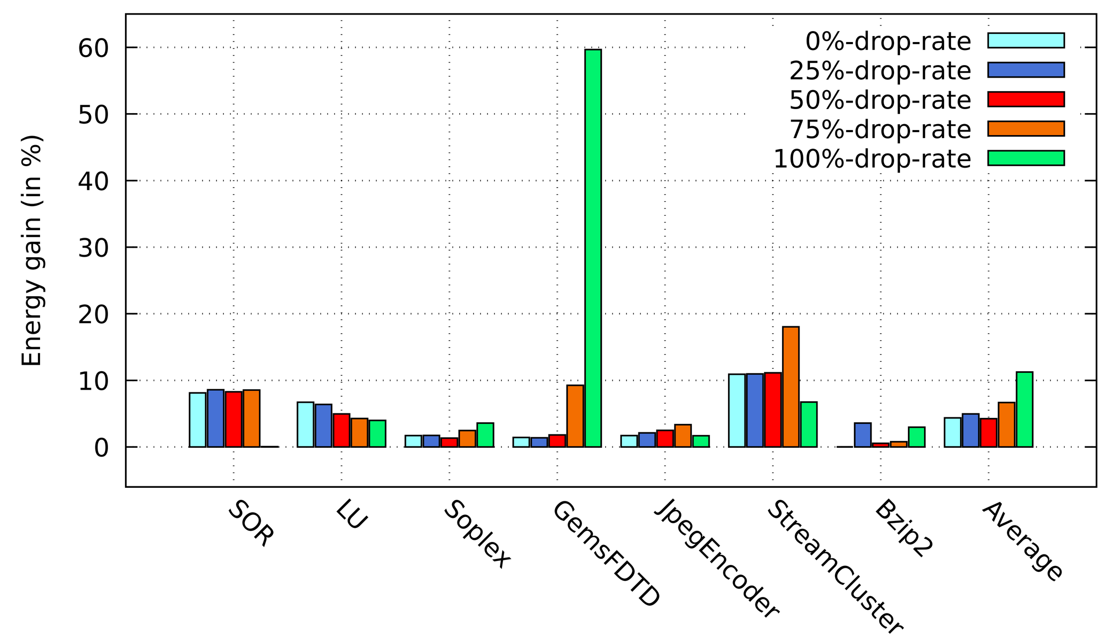


Figure 8: Energy gain in load approximation

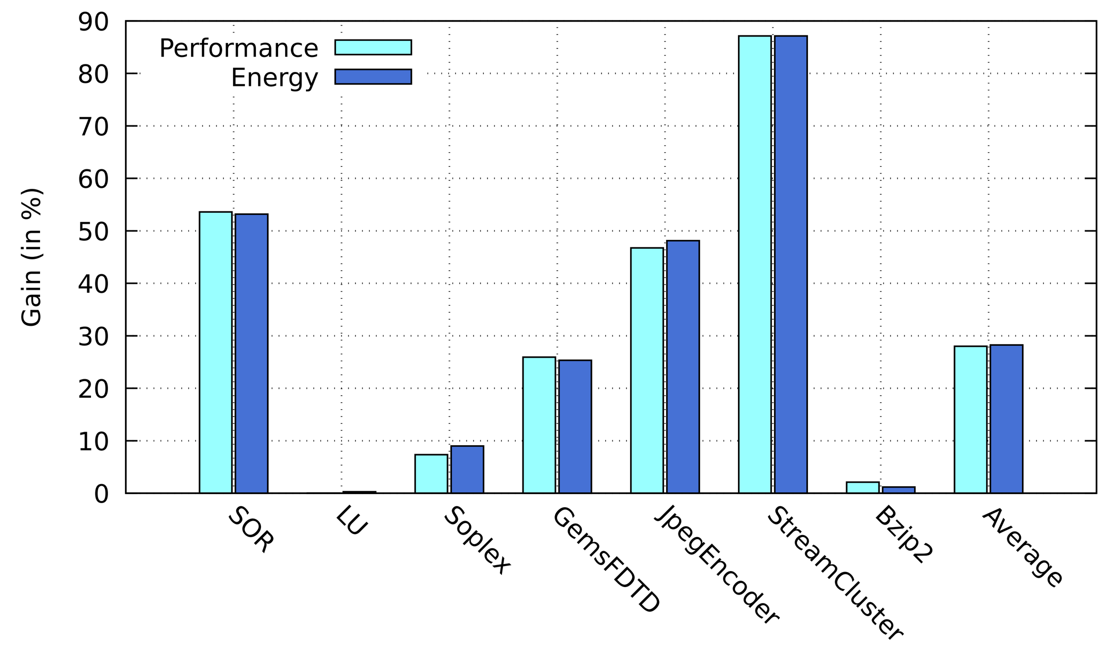


Figure 9: Performance and energy gain in branch approximation

Conclusion

- Present a statistical analysis of load and branch instructions in an application to classify into sensitive and insensitive one.
- The inference comes with a probabilistic guarantee.
- Observed that the approximable loads, branches can tolerate mispredictions in a speculative execution to produce an output with acceptable QoS.
- We present a misprediction penalty free execution framework for approximable loads and branches, and show promising performance and energy benefits by architectural simulation.

References

[1] Luk et al. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05.

[2] Smruti R. Sarangi, Rajshekar Kalayappan, Prathmesh Kallurkar, Seep Goel, and Eldhose Peter. Tejas: A java based versatile micro-architectural simulator. In *International Workshop on Power And Timing Modeling, Optimization and Simulation*, 2015.