ACR: Enabling Computation Reuse through Approximate Computing

Xin He, Guihai Yan, Yinhe Han and Xiaowei LI

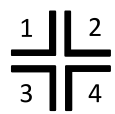
State Key Laboratory of Computer Architecture,
Institute of Computing Technology, Chinese Academy of Sciences
(**ICT, CAS**)

Power efficiency

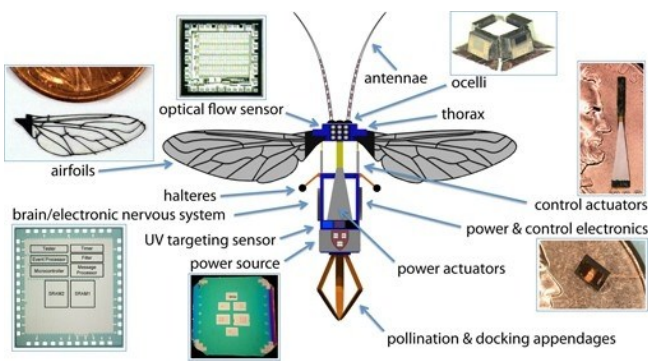
- High energy efficiency is required

Video gaming

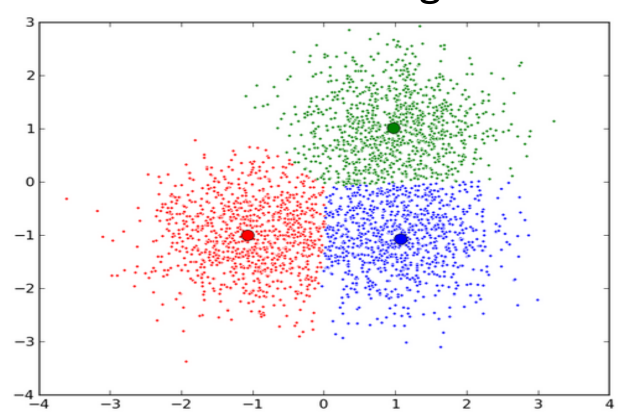
Image processing



Robotics



Machine learning



Approximate computing

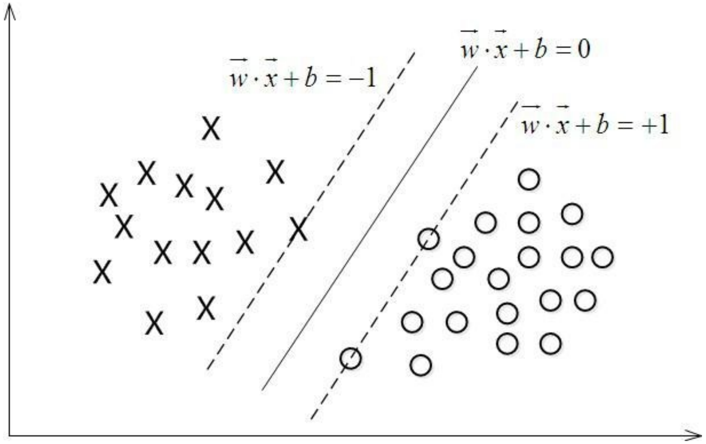
- Exploiting **tradeoffs** between quality and complexity

DCT Image Processing

Original method

Approximate method

Machine Learning

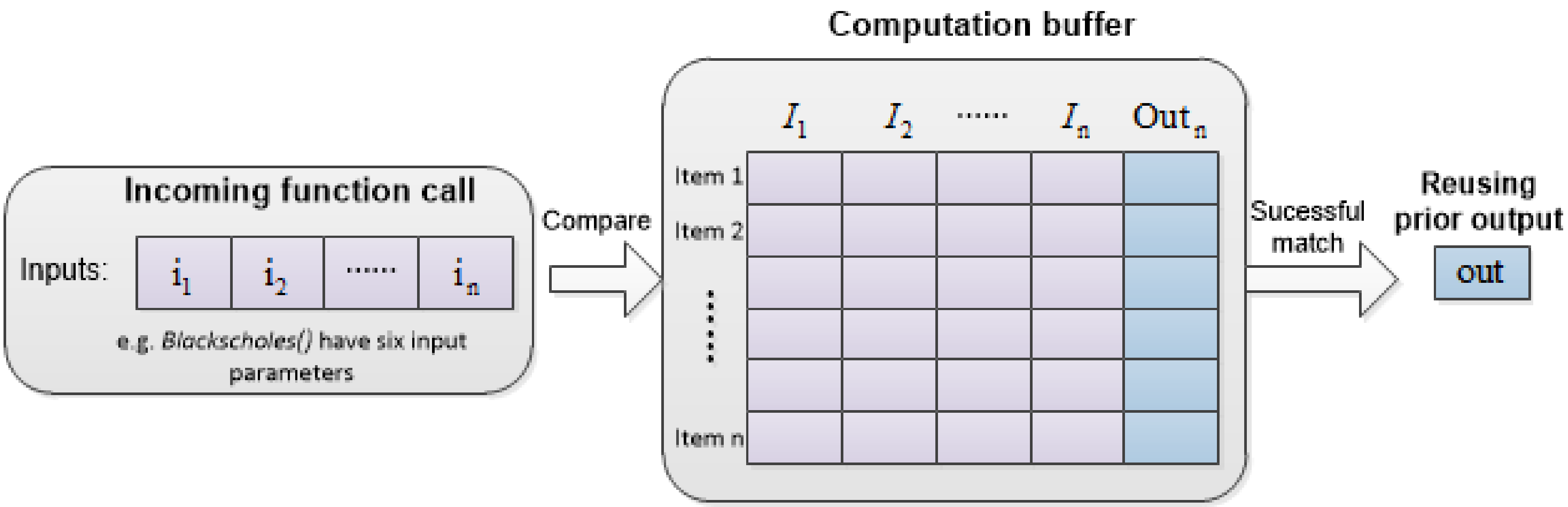


Quality: **2.82%** Reduction
Performance: **31.44%** Improvement

Quality: **1.3%** Reduction
Performance: **4.97X** Improvement

Conventional Computation Reuse scheme

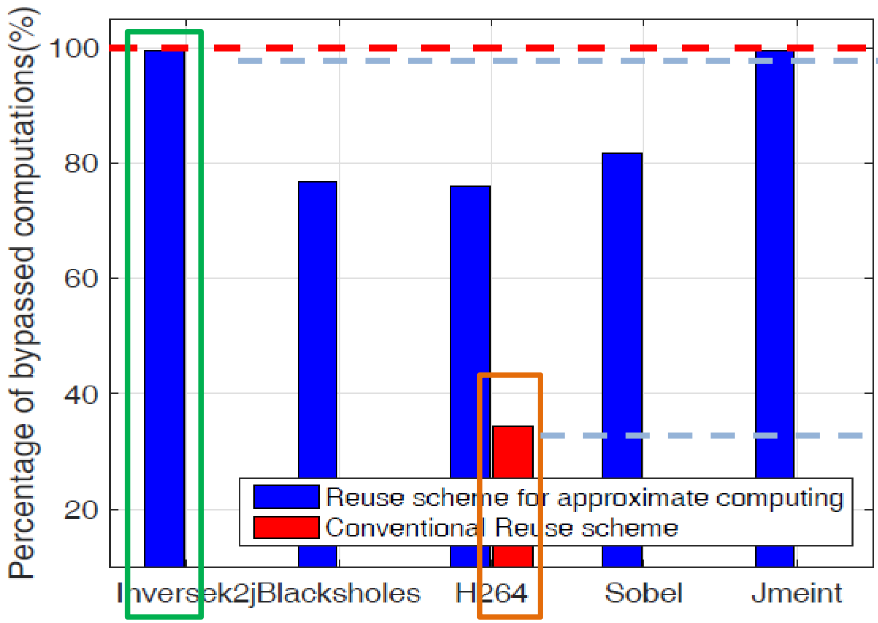
- Brief workflow of traditional computation reuse scheme



Reuse requirement If and only if input values of incoming calls **equal** to the ones in history

Efficacy of Conventional Reuse Scheme

➢ Proportion of reuse opportunities exploited



Reuse potential:
As much as 99%

Conventional reuse:
As low as 27%

In approximate application, there’s great reuse potential

Tight reuse requirement prevents CONVENTIONAL CRU from achieving high speedup

Requirements for approximate computing

- To enable computation reuse for approximate computing, we need

Similarity Quantification

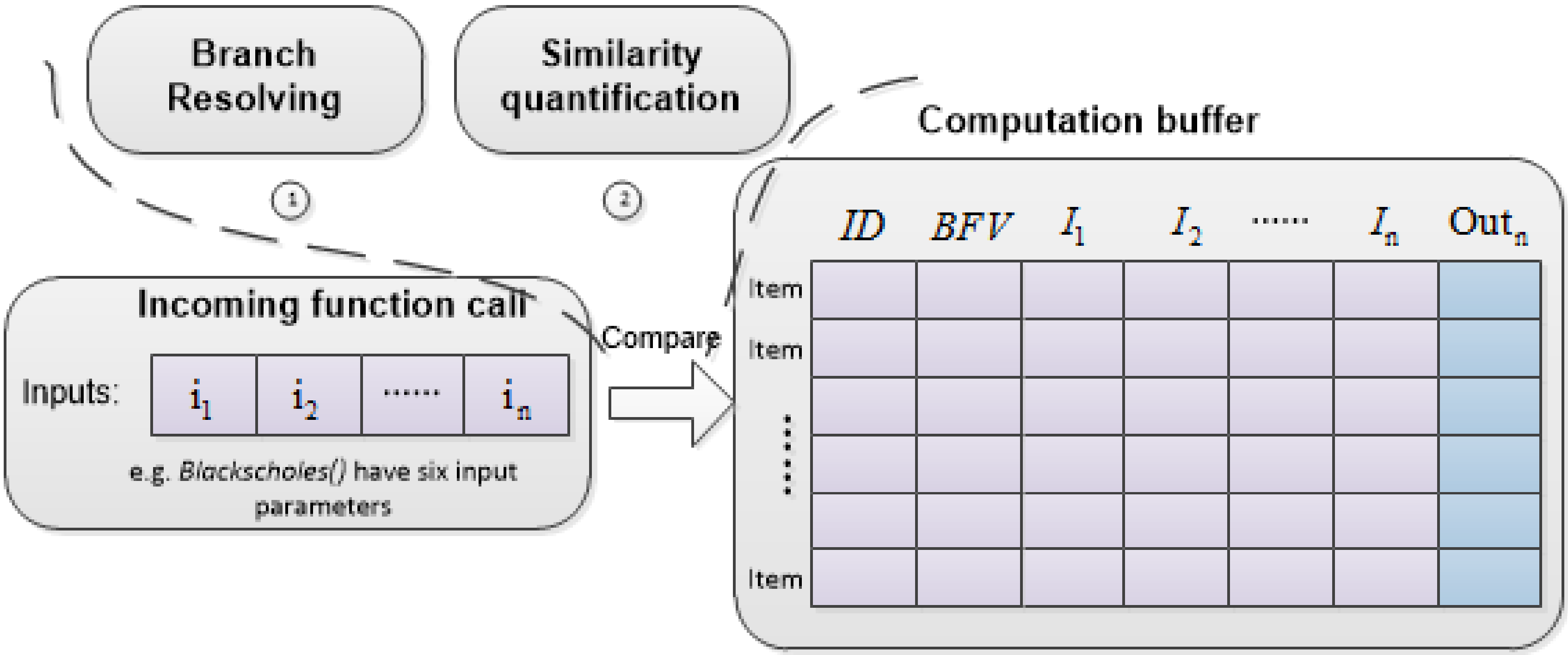
A criterion to measure the similarity between different

Branches Resolving

Reduce the latency of packets according to traffic heterogeneity

Workflow of Proposed ACR

- Two extra steps should be included into approximate computing process



Deciding the computation pattern → Calculating the similarity

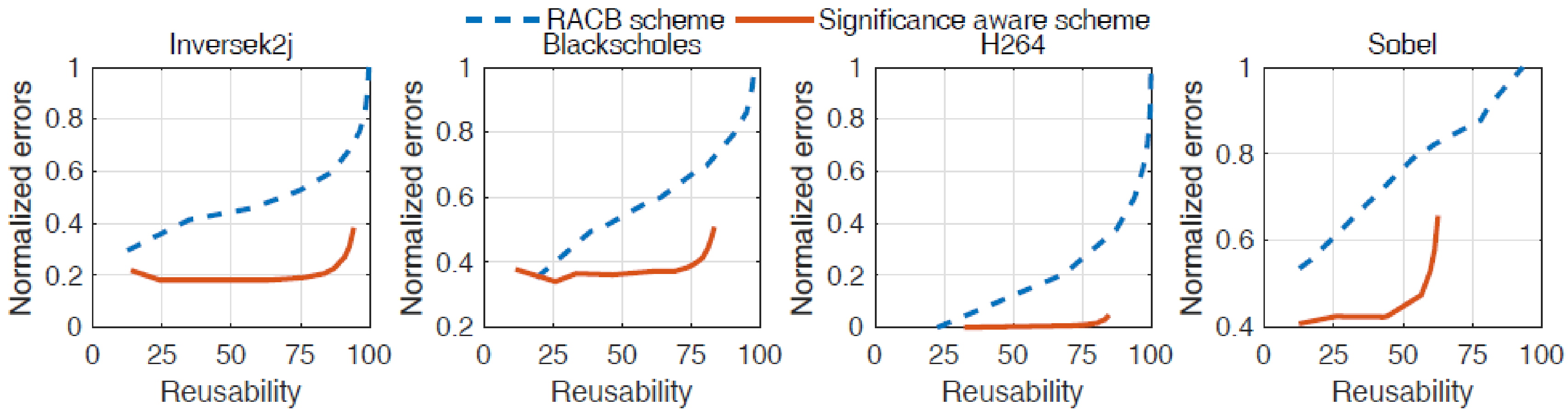
Similarity Quantification

- Prior approximate reuse scheme (i.e. RACB scheme from islped 2005)

Main idea: masking LSBs of values of inputs, and requiring the MSBs to be equal for approximate reuse

Pros:
1. Easy to implement
2. Effective for specific applications

Cons:
1. Too arbitrary
2. Limited speedup



significance aware approximate reuse

- Capture the relation between inputs and outputs and estimate the significance of inputs

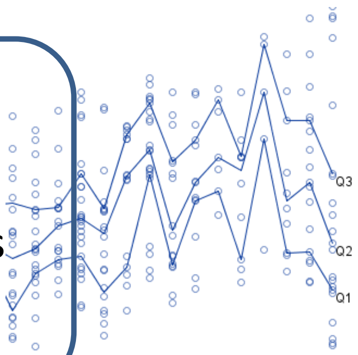
Code region or function at runtime

blackbox

Perturbation analysis cannot be applied

Use executed computation to model the code region or function

We have:
1)Values of inputs
2)Values of outputs
3)Branch decisions



from compiler explicit markers

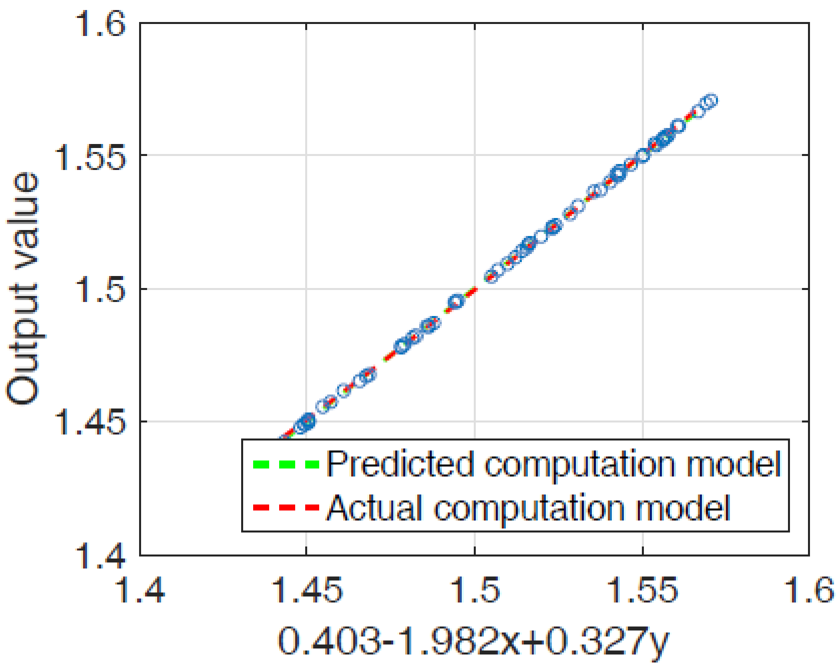
Proposed significance aware approximate reuse

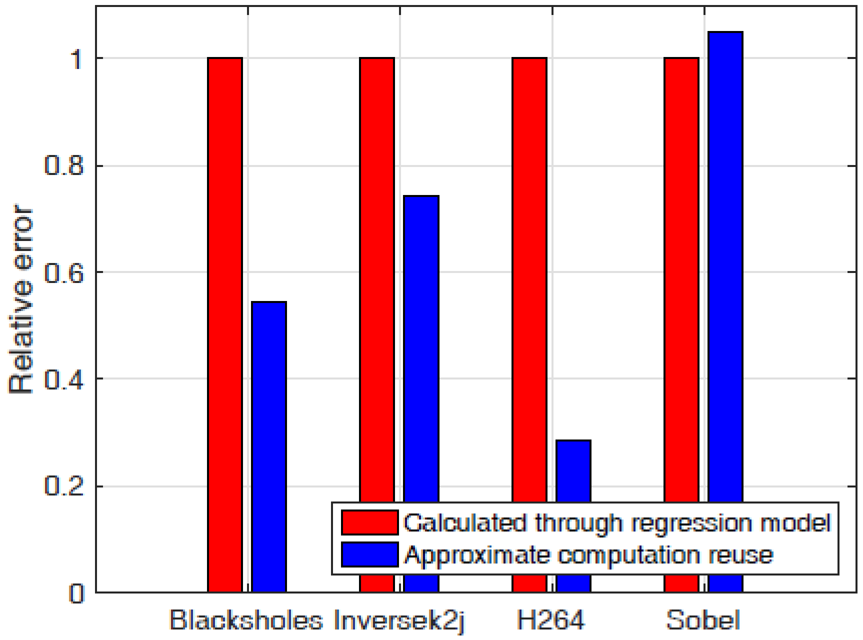
- Linear regression based statistical technique to obtain significance of inputs

E.g. inversek2j() benchmark:

$$z = \text{acos}\left((x^2 + y^2 - 0.5)/0.5\right) \quad (1)$$

$$Output = asin\left(\left(y \times \left(0.5 + 0.5 \times cos(z)\right) - x \times 0.5 \times \sin(z)\right)/(x^2 + y^2)\right) \quad (2)$$





Resolving conditional branches

- Logistic regression based statistical technique to modeling conditional branches in function
- Then the model is used to obtain branch decision of incoming

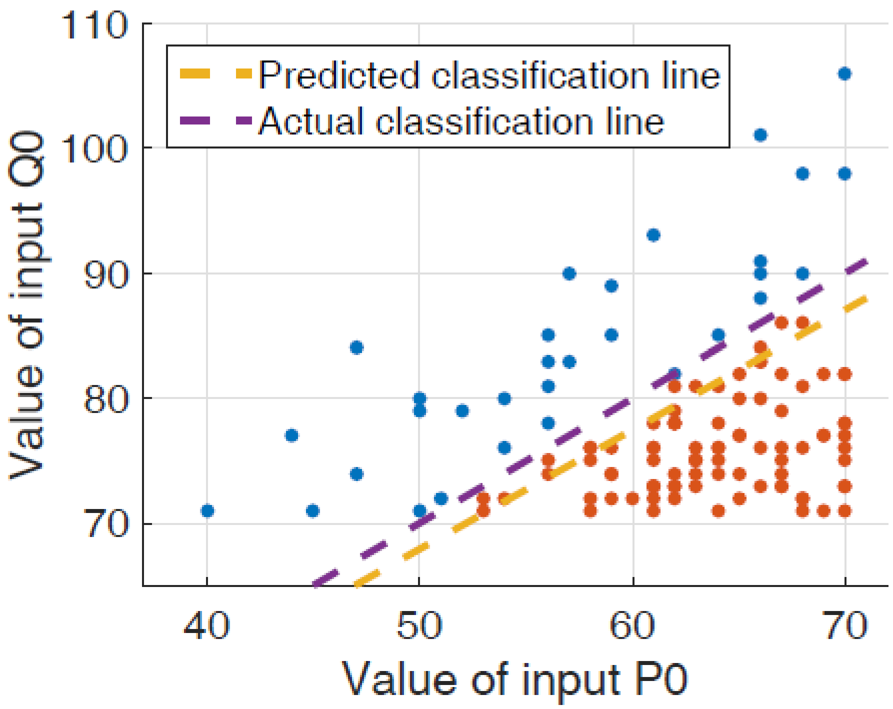
Speculative branch prediction

Actual conditional branch:

$$if(qo - p0 < 20)$$

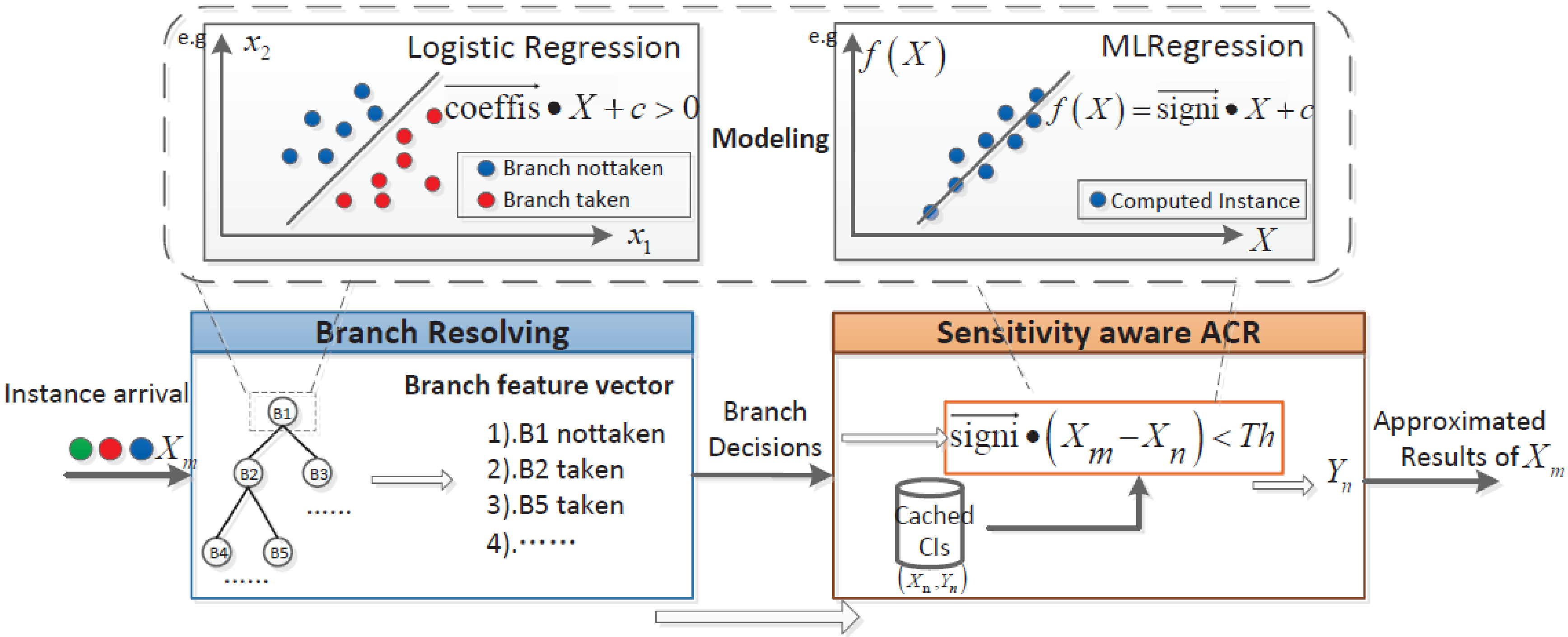
Predicted conditional branch:

$$-0.96p0 + 0.08p1 - 0.06p2$$
$$+q0 - 0.07q1 + 0.03q2 = 20.09$$



Overview of proposed ACR scheme

1. Trained with computations executed in history



Workflow of Speculative Approximate Computation Reuse

2.

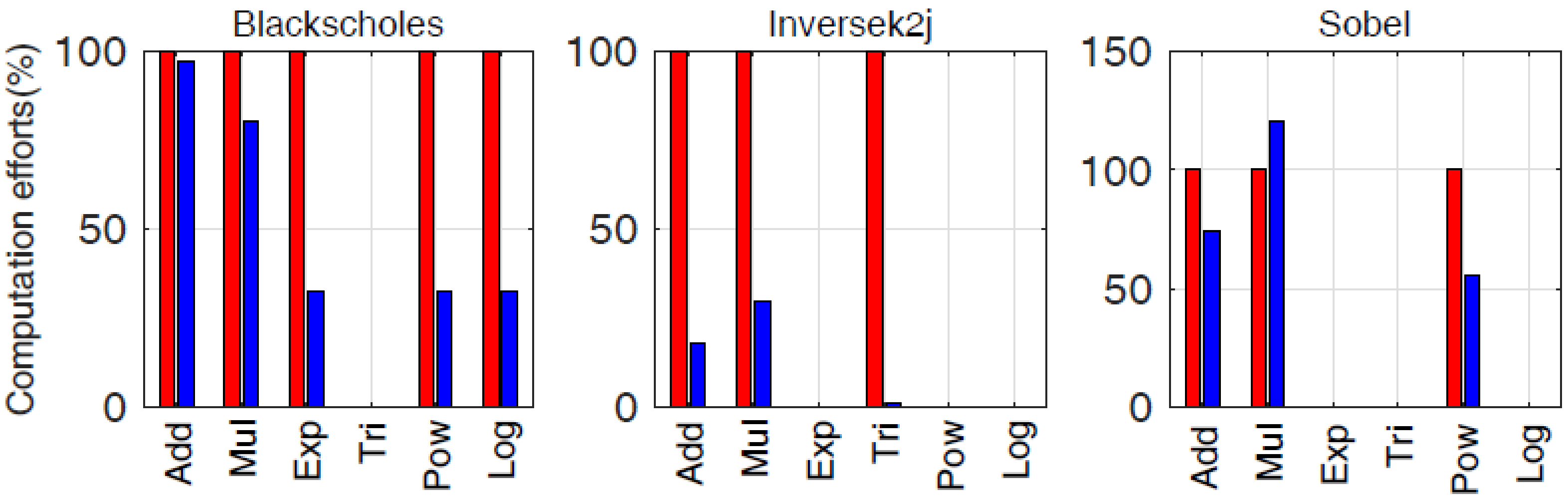
For an incoming Computation, | Its computation pattern is specified first | Then, | Its similarity with history computation is obtained

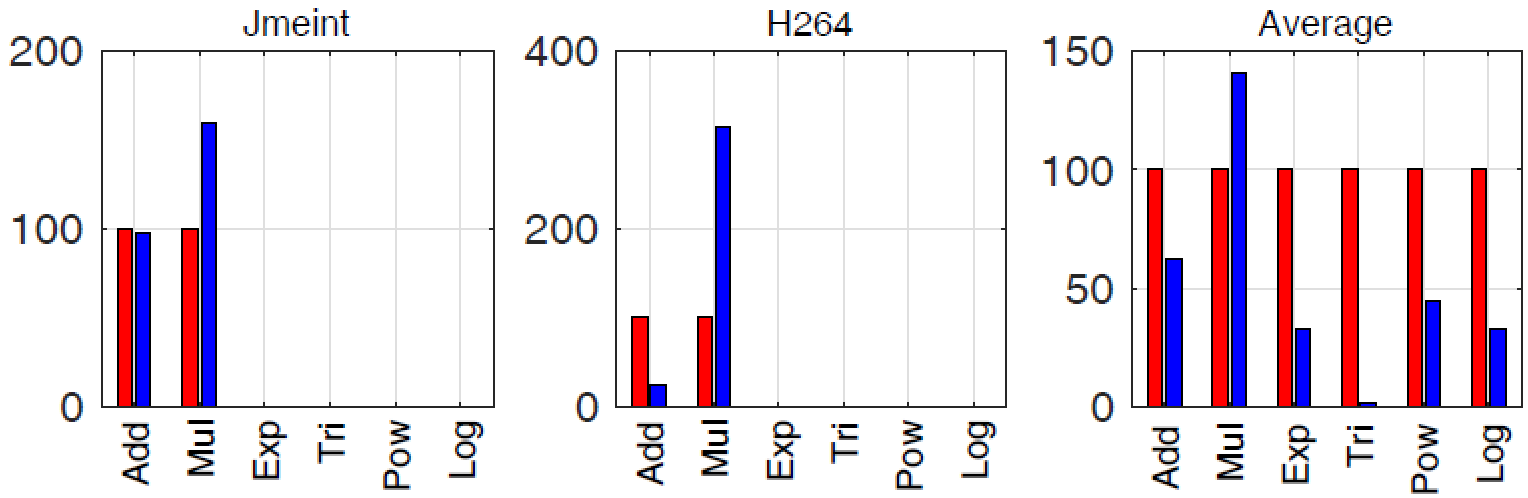
Benchmarks

- Axbench: a benchmark designed for approximate computing from Georgia tech

Benchmark	Description	N.in	N.br
Blackscholes	Pricing a portfolio of options with the Black-Scholes equation	6	5
Inversek2j	Robotic: Inverse kinematics for 2-joint arm	2	0
Sobel	Sobel edge detector in Image Processing	9	1
Jmeint	Triangle intersection detection in 3D gaming	18	20
H264	Loop filter in h264	6	5

The percentage of computation effort




Jmeint
H264
Average
200
100
0
400
200
0
150
100
50
0
Add
Mul
Exp
Tri
Pow
Log

Storage

- Area overhead:

Benchmark	Blackscholes	Inversek2j	Sobel	Jmeint	H264
Storage Cost(kB)	65.73	40.08	29.05	37.14	42.38

Conclusion

- Approximate computing can exploit the potential of computation reuse
- Not all error-tolerant application are suitable for approximate computation reuse
- Error-tolerant applications which benefit most from ACR would be:
  - Time consuming: contains complex function like sin(), cos(), exponential function
  - Contain small number of input parameters
  - Have a few number of conditional branches

Thanks for listening!

Question?