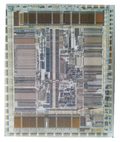
Instruction-level Timing Error Prediction Modelling

Northwestern University

Yuanbo Fan, Chao Yan { yuanbo, chaoyan2012 }@u.northwestern.edu
EECS 349 Machine Learning, advised by Prof. Bryan Pardo



Abstract

In this project, we build an instruction-level timing error prediction model for a 5-stage pipelined ALPHA processor which supports timing speculation. This model predicts that an instruction has timing error given the information of instruction sequence and data usage.

Introduction

Timing error is defined as a violation in circuit-level timing constraints during program execution. In traditional processors, timing error may cause catostrophic system failure. Processors which support timing speculation are augmented with timing-error detection and correction techniques so that they able to recover from timing errors.

However, the recovery cost is very high. Also, based on some existing work, for a specific processor, timing error is a strong function of programs. Therefore, an effective timing-error prediction model can be built based on the information of how the programs are executed on the processor. Then, the compiler can apply this model to generate codes that have lower probability of producing timing errors by using instruction scheduling and instruction selection. Moreover, if the model is implemented in the hardware, it can predict timing error at run-time. With some padding techniques, it can avoid the full recovery cost due to timing errors.

In this project, we use machine learning algorithms to build an instruction-level timing error prediction model for a 5-stage pipelined ALPHA processor which supports timing speculation. This model predicts that an instruction has timing error given the information of instruction sequence and data usage.

Learning Process

- **Dataset**

  We use ten instruction-level test programs to build ten test sets. In each set, we pick nine programs for trainning and the other one for testing.

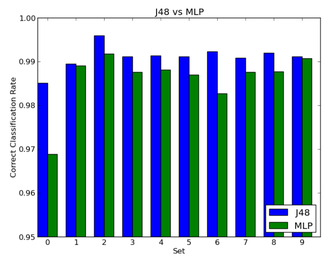
NO.	Name	Description
1	copy	copy memory contents of N elements
2	objsort	object sorting algorithm
3	parsort	parsort algorithm
4	fib	compute nth fibonacci numbers
5	fib_rec	compute nth fibonacci number recursively
6	evens	compute even numbers that are less than n
7	insertion	insertion sorting
8	parallel	compute first n multiples of m
9	sort	bubble sorting
10	saxpy	integer SAXPY

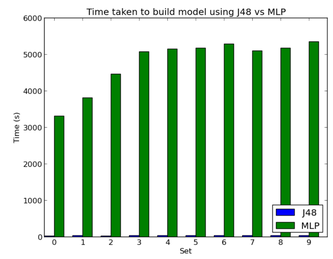
```
/*
   TEST PROGRAM #3: compute first 16 fibonacci numbers
                    with forwarding and stall conditions in the loop
*/

    long output[16];

    void
    main(void)
    {
      long i, fib;

      output[0] = 1;
      output[1] = 2;
      for (i=2; i < 16; i++)
        output[i] = output[i-1] + output[i-2];
    }
*/

    data = 0x1000
    lda     $r1,data
    lda     $r4,data+8
    lda     $r5,data+16
    lda     $r9,2
    lda     $r1,1
    stq     $r1,0($r3)
    addq    $r1,$r9,$r9
loop:   ldq     $r2,0($r3)
    ldq     $r2,0($r4)
    addq    $r2,$r1,$r2
    addq    $r3,0x8,$r3
    addq    $r4,0x8,$r4
    addq    $r9,0x1,$r9
    cmpult  $r9,0x10,$r18
    stq     $r2,0($r3)
    addq    $r5,0x8,$r5
    bne     $r18,loop
    call_pal        0x555
```

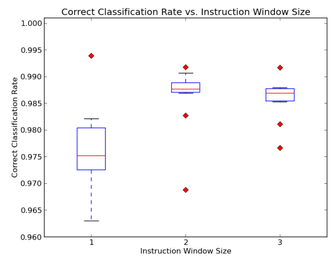
Each instance in the training and testing sets consists of several parts, including the current instruction and (N-1) preceding instructions. N is defined as the instruction window size which means the number of instruction affect that if the current instruction produces a timing error. For operand data used in each instruction, we use both number and bits representation.

- **Learning Algorithms**
  - J48 Decision Tree
  - Multilayer Perceptron
- **Sofware Packages**
  - Synopsys tools
  - WEKA software packages

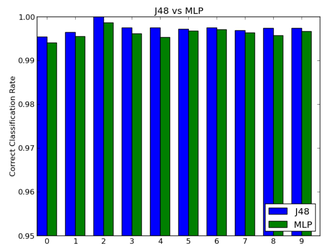
Results & Analysis

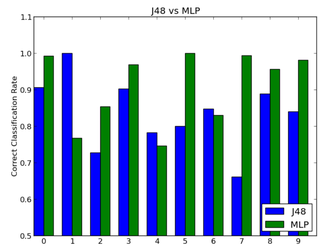




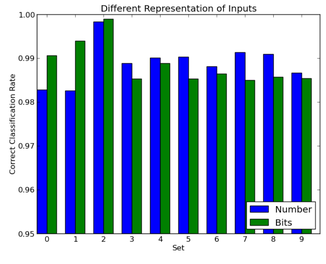


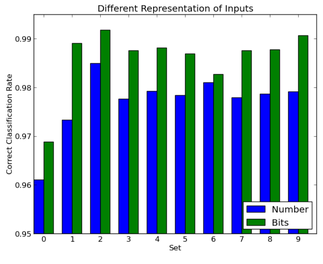
- J48 Decision Tree is slightly better. Building a model using Multilayer Perceptron takes much longer.
- Instruction Window Size of 2 overperforms others.





- The left figure shows the results of using 10-fold cross validation on the training set and validating using the testing set.
- Multilayer perceptron algorithm has less overfitting problem in our case.





- The left figure has 10% error rate in the test program, while the right one has 20%.
- This proves that when the rates change from 10% to 20%, a large portion of errors are caused by the values used in the program.

References

[1] J. Xin and R. Joseph. Identifying and predicting timing-critical instructions to boost timing speculation. In MICRO-44 '11, pages 128–139, New York, NY, USA, 2011. ACM.

Conclusions & Future Direction

Both J48 Decision Tree and Multilayer Perceptron algorithm are good to use to build models to predict timing errors. Each algorithm has its own advantages and disadvanges.

This timing error prediction model can be used either to improve the overall performance of a program or to study interesting characteristics of timing errors.

Website

All resources are available at
`http://users.eecs.northwestern.edu/~yfw492/eecs349/`